System BIOS Interrupts

INT 10h Video Service

INT 10h, the video interrupt routine, has seventeen functions supported by the system BIOS. The system BIOS only supports two video display adapters: monochrome display adapter (MDA) and color graphics adapter (CGA). The BIOS support for EGA, VGA, and XGA display adapters is provided on the video adapter. If EGA is used, INT 42h points to the BIOS Video Service Routine. Both the EGA and VGA video BIOS reside at C000h.

INT 10h Functions

Function	Title
00h	Set Video Mode
01h	Set Cursor Type
02h	Set Cursor Position
03h	Return Cursor Position
04h	Return Light Pen Position
05h	Set Current Video Page
06h	Scroll Text Upward
07h	Scroll Text Downward
08h	Return Character or Attribute
09h	Write Character or Attribute
0Ah	Write Character
0Bh	Set Color Palette Subfunction BH = 00h Set Palette Subfunction BH = 01h Set Color Palette
0Ch	Write Graphic Pixel
0Dh	Read Graphic Pixel
0Eh	Write Character
0Fh	Return Video Display Mode
13h	Write Character String

Note that the IBM BIOS destroys the contents of AX, BX, SI, DI, and BP after all INT 10h function calls, but AMIBIOS does not.

Cont

Function 00h Set Video Mode

Input: AH = 00h
AL = Video Mode
00h 40 x 25 text mode, monochrome with CGA card
01h 40 x 25 text mode, color with CGA card.
02h 80 x 25 text mode, monochrome with CGA card
03h 80 x 25 text mode, color with CGA card
04h 320 x 200 four-color graphics, with CGA card
05h 320 x 200 monochrome, with CGA card
06h 640 x 200 monochrome, with CGA card
07h 80 x 25 monochrome, with monochrome card

Output: No registers set.

Description Function 00h sets the video mode. Only the video modes supported in the MDA and CGA video standards are supported by the system BIOS. This function programs the CRTC, selects a default color palette, and clears the video buffer if the proper flag is set in the save area.

CGA Video Modes

Mode	Adapter	Resolution	Type	Colors	Lines and Rows	Array	Max. Pages	Buffer
0, 1	CGA	320x200	Text	16/256K	40x25	8x8	8	B800h
2, 3	CGA	642x200	Text	16/256K	80x25	8x8	4	B800h
4, 5	CGA	320x200	Graphics	4/256K	40x25	8x8	1	B800h
6	CGA	640x200	Graphics	2/256K	80x25	8x8	1	B800h
7	MDA	720x350	Text	None	80x25	9x14	1	B000h

Cont

Function 01h Set Cursor Type

Input:
AH = 01h
CH = Starting cursor line (bits 4–0). If set to 20h, the cursor is disabled.
CL = Ending Cursor Line (bits 4 – 0)

Output: No registers set. 40:60h is updated.

Description Function 01h sets the cursor type. If using MDA, valid values are 0–13. Using CGA, valid values are 0–7. If CH is 20h, the cursor is disabled. This function programs the CRTC to display the text cursor type. Only one cursor type is maintained for each video page. The BIOS default values are:

Video Type	Description	Register	Initial Value
Monochrome (MDA)	Starting Cursor Line	CH	11
	Ending Cursor Line	CL	12
CGA Color	Starting Cursor Line	CH	6
	Ending Cursor Line	CL	7

Function 02h Set Cursor Position

Input:
AH = 02h
BH = Video Page Number
DH = Line on Screen
DL = Column on Screen

Output: No registers set. 40:50h is updated.

Description Function 02h positions the cursor in a video page. Valid values for DH are 0 – 24. Valid values for DL are 0 – 39 in 40-column mode and 0 – 79 in 80-column mode. If the current video page number is in BH, the BIOS programs the CRTC to update the current cursor position on the specified page.

Cont

Function 03h Return Cursor Position

Input:	AH	=	03h
	BH	=	Video Page Number
Output:	CH	=	Beginning Line of the Blinking Cursor
	CL	=	Ending Line of the Blinking Cursor
	DH	=	Line on Screen
	DL	=	Column on Screen

Description Function 03h reads the current cursor position on the specified video page. This function is used only in text mode.

Function 04h Return Light Pen Position

Input:	AH	=	04h
Output:	AH	=	Position on line 00h Position is unreadable 01h Position is readable 04h Light pen disabled or no valid address.
	BX	=	Column on Graphic Screen (Pixel)
	CH	=	Line on Graphic Screen (Raster Line)
	CL	=	Raster line if resolution of mode is less than 200 lines.
	DH	=	Line on text screen
	DL	=	Column on text screen

Description Use this function to find the position of the light pen. This routine is not accurate in graphics mode and is ineffective when used on monochrome monitors with long image-retention phosphors. The raster line value is always a multiple of two. Depending on screen size, the pixel value is a multiple of four (in 320 x 200 mode) or eight (in 640 x 200 mode).

Cont

Function 05h Set Current Video Page

Input:
AH = 05h
AL = Video page number

Output: None

Description Function 05h sets a new video page or selects the portion of the video buffer to be displayed. This function is ignored if CGA is used because CGA uses the entire 16K video buffer. The BIOS programs the CRTC Start Address Registers in video modes 0 – 3. The BIOS maintains the current cursor location in up to eight video pages at 40:50h. When a new video page is selected, the BIOS moves the cursor to the position the cursor was at the last time the requested video page was displayed.

Function 06h Scroll Text Upward

Input:
AH = 06h
AL = Number of scrolling lines
BH = Color or attribute for scrolling lines
CH = Line Number of upper left window corner
CL = Column number of upper left window corner
DH = Line number of lower right window corner
DL = Column number of lower right window corner

Output: None

Description Function 06h creates a window defined by values specified in CH, CL, DH, and DL. It scrolls the number of window lines upward through the window. The number of lines is defined in AL, and the color or attribute of the new lines is in BH. If AL is set to 00h, the window is cleared.

Cont

Function 07h Scroll Text Downward

Input:

AH	=	07h
AL	=	Number of scrolling lines
BH	=	Color or attribute for scrolling lines
CH	=	Line Number of upper left window corner
CL	=	Column number of upper left window corner
DH	=	Line number of lower right window corner
DL	=	Column number of lower right window corner

Output: None

Description Function 07h creates a window (defined by values in CH, CL, DH, and DL) and scrolls a number of window lines downward through the window. The number of lines to be scrolled is in AL, and the color or attribute of the new lines is in BH. If AL is set to 00h, the window is cleared.

Function 08h Return Character or Attribute

Input:

AH	=	08h
BH	=	Video page number

Output:

AH	=	Color or attribute of character
AL	=	ASCII Code of character

Description Function 08h gets the ASCII code of the character at the current cursor location on the video page specified in BH. The function returns the character attribute or color in AH.

Cont

Function 09h Write Character or Attribute

Input:	AH	=	09h
	AL	=	ASCII code of character to be written
	BH		Video page number (or background pixel value if in 320 x 200 x 256 color mode)
	BL		Attribute or color of character (or background pixel value in graphics mode)
	CX		Number of repetitions

Output: None

Description INT 10h Function 09h writes a character(s) to the current cursor position on the video page specified in BH. You can also specify the character attribute or color and the number of times the character is to be written. The new cursor position is not changed.

Function 0Ah Write Character

Input:	AH	=	0Ah
	AL	=	ASCII code of character to be written
	BH		Video page number (or background pixel value if in 320 x 200 x 256 color mode)
	BL		Foreground pixel value (in graphics mode only)
	CX		Number of repetitions

Output: None

Description INT 10h Function 0Ah writes a character(s) to the current cursor position on the video page specified in BH. You can also specify the number of times the character is to be written. The new cursor position is not changed.

Cont

Function 0Bh Subfunction 00h Set Palette

Input: AH = 0Bh
BH 00h
BL Screen border and background color

Output: No registers set. 40:66h is updated

Description INT 10h Function 0Bh subfunction 00h sets the screen background and border color. If the computer is running in text mode, only the screen border color is defined. If the computer is running in graphics mode, both the background and the screen border colors are defined. Use INT 10h Function 10h instead of this function if the computer is using EGA or VGA.

Function 0Bh Subfunction 01h Set Color Palette

Input: AH = 0Bh
BH 01h
BL Number of color palette

Output: No registers set. 40:66h is updated

Description Function 0Bh subfunction 01h is valid only in 320x200 graphics mode. It also sets the screen color palette. The two palettes in 320x200 mode are:

Palette	Colors
Palette 0	Green, Red, and Yellow
Palette 1	Cyan, Magenta, and White

Cont

Function 0Ch Write Graphic Pixel

Input:

AH	=	0Ch
AL	=	Pixel color number
BH	=	Video page number. This function can only be issued in video modes that permit multiple pages.
CX	=	Screen column number
DX	=	Screen line number

Output: None

Description Function 0Ch draws a color graphic pixel at the specified coordinates in CX and DX. Specify the video page in BH and the pixel color number in AL. The BH value is ignored in 320x200x256 color mode. If VGA or EGA is used, the BH value is ignored in 320x200x4 color mode.

Function 0Dh Read Graphic Pixel

Input:

AH	=	0Dh
BH	=	Video page number. This function can only be issued in video modes that permit multiple pages.
CX	=	Screen column number
DX	=	Screen line number

Output:

AL	=	Pixel color number

Description Function 0Dh reads the color of the pixel specified in CX and DX. The current video page is specified in BH.

Cont

Function 0Eh Write Character

Input:
AH = 0Eh
AL = ASCII Code of the character
BH = Active video page number.
BL = Foreground color (graphics modes)

Output:
AL = No registers set. 40:50h is updated

Description Function 0Eh writes a character to the current video page at the current cursor position. The cursor column position is incremented after writing the character. If the end of a line is reached, the cursor row position is also incremented and the column position is set to 0. The ASCII control characters are: 07h = beep, 08h = backspace, 0Ah = line feed, and 0Dh = carriage return.

Function 0Fh Return Video Display Mode

Input:
AH = 0Fh

Output:
AH = Number of display columns
AL = Video mode
- 00h 40x25 text mode monochrome in CGA
- 01h 40x25 text mode color in CGA
- 02h 80x25 text mode monochrome in CGA
- 03h 80x25 text mode color in CGA
- 04h 320x200 four-color graphics in CGA
- 05h 320x200 monochrome in CGA
- 06h 640x200 monochrome in CGA
- 07h 80x25 monochrome in monochrome

BH = Current video page

Description This function returns the current video mode in AL, the current page number in BH, and the number of columns allowed in this video mode in AH.

Cont

Function 13h Write Character String

Input:

AH	=	13h
AL	=	Output mode 00h Attribute in BL, do not update cursor position. 01h Attribute in BL, update cursor position. 02h Attribute in string buffer, do not update cursor position. 03h Attribute in string buffer, update cursor position.
BH	=	Video page number
BL	=	Attribute of all characters in character string
CX	=	Number of characters in buffer
DH	=	Screen line number
DL	=	Screen column number
ES:BP	=	Segment:offset address of string buffer

Output: No registers set. 40:50h is updated

BH =

Description Function 13h writes character strings to the screen and wraps the string to the next line if it is too long for the current text line. Specify the video page number in BH, the screen line number in DH, and the screen column number in DL where the string is to be displayed. The string should be stored in a buffer in RAM. The buffer address segment is in ES and the offset in BP. The number of characters to be displayed from the buffer should be in CX. If output modes 0 or 2 are used, this function does not change the cursor position.

If output modes 1 or 3 are used, this function sets the final cursor position to the next position past the last character displayed. If the output mode is 0 or 1, the attribute for all characters in the string is determined by the value in BL. In modes 2 and 3, the string consists of sets of two bytes. The first byte is the ASCII value of the character and the second byte is the attribute of the character.

INT 11h Return System Configuration

Input: None

Output: AX = Configuration code

Bits 15–14	Number of parallel ports installed
Bits 11–9	Number of serial ports installed
Bits 7–6	Number of floppy drives
	00 One floppy disk drive
	01 Two floppy disk drives
Bits 5–4	00 VGA or EGA
	01 Video mode is 40x25 CGA
	10 Video mode is 80x25 CGA
	11 Video mode is 80x25 MDA
Bit 2	PS/2 mouse present if set
Bit 1	Math coprocessor installed if set
Bit 0	1 One or more floppy drives.

Description INT 11h reads the system configuration code. The video mode reported by INT 11h is the mode used when the computer was booted. Use INT 10h Function 0Fh to find the current video mode.

INT 12h Return Total Memory Size

Input: None

Output: AX = Number of kilobytes of contiguous memory beginning at absolute address 00000h

Description INT 12h returns the amount of real mode memory available. Real mode memory is memory below the first megabyte address. Use INT 15h Function 88h to find the amount of system memory beyond the first megabyte up to 64 MB. Use INT 15h Function E2 to find system memory above 64 MB.

This interrupt is dependent on the CPU model. For current Pentium processors, you can read the reason for the machine check exception from model-specific registers 00h and 01h. This exception is enabled by setting bit 4 of CR4.

INT 13h Hard Disk Service

Functions The INT 13h functions discussed in this chapter are:

Function	Title
00h	Reset Hard Disk Drive
01h	Return Hard Disk Drive Status
02h	Read Disk Sectors
03h	Write Disk Sectors
04h	Verify Disk Sectors
05h	Format Disk Cylinder
06h	Format Disk Track and Mark Lead Sectors
07h	Format Entire Disk Starting at Specified Cylinder
08h	Return Disk Parameters
09h	Initialize Hard Disk Controller
0Ah	Read Hard Disk Sectors and Error Correction Codes
0Bh	Write Hard Disk Sectors and Error Correction Codes
0Ch	Seek Hard Disk Cylinder
0Dh	Reset Hard Disk Controller
10h	Test Unit Ready
11h	Recalibrate Hard Disk
14h	Perform Internal Controller Diagnostic
15h	Return Drive Type
41h	Check Extension Present
42h	Extended Read
43h	Extended Write
44h	Verify Sectors
45h	Lock Drive
46h	Eject Media
47h	Extended Seek
48h	Get Drive Parameters
49h	Extended Media Change
4Ah	Initiate Disk Emulation for Bootable CD-ROM
4Bh	AL = 00h Terminate Disk Emulation for Bootable CD-ROM AL = 01h Get Status of Bootable CD-ROM
4Ch	Start Disk Emulation and Boot Bootable CD-ROM Drive
4Dh	Return Boot Catalog for Bootable CD-ROM Drive
4Eh	Set Hardware Configuration

Cont'd

Error Codes For most hard disk drive functions, the following error codes are returned through register AH. All error codes appear in AH.

Code in AH	Description
00h	Successful completion
01h	Invalid function in AH or invalid parameter
02h	Address mark not found
03h	Disk write-protected
04h	Sector not found or read error
05h	Reset failed
06h	Disk Line changed
07h	Drive parameter activity failed
08h	DMA overrun
09h	Data boundary error. Attempt to run DMA across a 64K boundary or >80h sectors.
0Ah	Bad sector detected
0Bh	Bad track detected
0Ch	Unsupported track or invalid media
0Eh	Control data address mark detected
0Fh	DMA arbitration level out of range
10h	Uncorrectable CRC or ECC error on read
11h	Data ECC corrected
20h	Controller failure
31h	No media in drive
40h	Seek failed
80h	Timeout. Drive not ready.
AAh	Drive not ready
B0h	Volume not locked in drive
B1h	Volume locked in drive
B2h	Volume not removable
B3h	Volume in use
B4h	Lock count exceeded
B5h	Valid eject request failed
BBh	Undefined error
CCh	Write fault
E0h	Status register error
FFh	Sense operation failed

Cont

INT 13h Coding Conventions For most INT 13h functions, the sector number is placed in CL and the cylinder number in CH.

On a hard disk drive, the cylinder number consists of 10 bits. The lower 8 bits are placed in CH (cylinder number), and the upper 2 bits are placed in CL. The lower 6 bits of CL contain the beginning sector number.

INT 40h Revector for Floppy Functions INT 13h handles both floppy disk and hard disk drive BIOS functions. If the computer has a hard disk drive, the floppy disk device service routine actually resides at INT 40h. All BIOS floppy functions are actually revectored to INT 40h and then executed.

Function 00h Reset Disk Drive

Input:	AH	=	00h
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description INT 13h Function 00h should be used when an error follows a disk operation. Function 00h resets the disk controller and recalibrates the hard drives attached to the controller.

If Function 00h is called for a hard disk drive, the floppy controller is reset and then the hard disk drive controller is reset.

Cont

Function 01h Return Disk Drive Status

Input:	AH	=	01h
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description INT 13h Function 01h can be used to read the status of the last operation.

Function 02h Read Disk Sectors

Input:	AH	=	02h
	AL	=	Number of sectors to read
	CH	=	Cylinder number (low 8 bits)
	CL	=	High two bits of cylinder number in bits 7–6
	DH	=	Head number
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
	ES:BX	=	Buffer segment:offset address
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description Function 02h reads the specified number of sectors from a specified track on one side of a disk. The sector(s) are read from the disk and then stored in a buffer at address ES:BX.

Cont

Function 03h Write Disk Sectors

Input:	AH	=	03h
	AL	=	Number of sectors to write (must not be zero)
	CH	=	Cylinder number (low 8 bits)
	CL	=	High two bits of cylinder number in bits 7–6
	DH	=	Head number
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
	ES:BX	=	Buffer segment:offset address
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description Function 03h writes the number of sectors in AL to the cylinder number in CH using the head specified in DH. The first sector number is in CL. The data written is in the buffer at address ES:BX.

Function 04h Verify Disk Sectors

Input:	AH	=	04h
	AL	=	Number of sectors to verify
	CH	=	Cylinder number (low 8 bits)
	CL	=	High two bits of cylinder number in bits 7–6
	DH	=	Head number
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
	ES:BX	=	Buffer segment:offset address
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description Function 04h verifies that the ECC code after each sector is correct for the data in that sector.

Cont

Function 05h Format Disk Track

Input:	AH	=	05h
	AL	=	Interleave factor
	CH	=	Cylinder number (low 8 bits)
	CL	=	High two bits of cylinder number in bits 7–6
	DH	=	Head number
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
	ES:BX	=	Buffer segment:offset address
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description Function 05h formats an entire track or cylinder on a disk. A buffer with sector data is in ES:BX. The buffer contains a two-byte record:

Byte	Contents
0	00h Good sector 80h Bad sector
1	Sector number

Function 06h Format Track and Mark Lead Sectors

Input:	AH	=	06h
	AL	=	Interleave factor
	CH	=	Cylinder number (low 8 bits)
	CL	=	High two bits of cylinder number in bits 7–6
	DH	=	Head number
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description Function 06h formats an entire track or cylinder of a hard disk and marks the bad sectors that it finds so they cannot be used. See INT 13h Function 05h for additional information about formatting.

Cont

Function 07h Format Entire Disk Starting at Specified Cylinder

Input:	AH	=	07h
	AL	=	Interleave factor
	CH	=	Cylinder number (low 8 bits)
	CL	=	High two bits of cylinder number in bits 7–6
	DH	=	Head number
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description This function formats an entire hard disk, starting at the cylinder number specified in CH and CL. Function 06h also marks bad sectors so these sectors cannot be used. See Function 05h for additional information about formatting.

Function 08h Return Disk Parameters

Input:	AH	=	08h
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
Output:	AH	=	00h No error
		=	Other values are error codes
	AL	=	00h
	CF	=	0 No error
		=	1 Error
	CH	=	Lower 8 bits of last cylinder number
	CL	=	High two bits of last cylinder number and six bits for last sector number
	DH	=	Last head number
	DL	=	Number of disk drives
	ES:DI	=	Address of disk parameter table from BIOS.

Description INT 13h Function 08h retrieves the parameters for a hard disk drive from the ROM BIOS.

Cont

Function 09h Initialize Hard Disk Controller

Input: AH = 09h
DL = 80h Hard Disk Drive C:
= 81h – FFh are valid. 81h = D:, 82h = E:, etc.

Output: AH = 00h No error
= Other values are error codes
CF = 0 No error
= 1 Error

Description INT 13h Function 09h initializes the hard disk controller with the values in the BIOS hard disk parameter table. The vector address for INT 41h points to the drive C: disk parameters and the vector for INT 46h points to the drive D: parameters. On an ISA computer, the blocks are 16 bytes, in the following format:

Offset	Description
00h – 01h	Number of cylinders. Byte 01h is the most significant byte.
02h	Number of heads.
03h – 04h	Reserved
05h – 06h	Starting write precompensation cylinder. Byte 06h is the MSB.
07h	ECC burst length
08h	Control Byte Bits 7–6 Enable or Disable Retries 00h Enable retries. All other values disable retries. Bit 5 1 Defect map is at last cylinder plus one. Bit 4 Reserved. Always set to 0. Bit 3 Set if more than 8 heads. Bits 2–0 Reserved. Always set to 0.
09h – 0Bh	Reserved
0Ch – 0Dh	Landing Zone
0Eh	Number of Sectors per Track
0Fh	Reserved

Cont

Function 0Ah Read Hard Disk Sectors and Error Correction Codes

Input:	AH	=	0Ah
	AL	=	Number of sectors to read
	CH	=	Lower eight bits of last cylinder number
	CL	=	Highest two bits of last cylinder number and six bits for beginning sector number.
	DH	=	Head number
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
	ES:BX	=	Buffer segment:offset address
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description

Function 0Ah reads the number of sectors in AL from the hard disk specified in DL and the location specified in CH and CL using the head number specified in DH and stores it to memory. It also reads a four-byte ECC code for each sector.

INT 13h Function 02h also reads sectors from the hard disk, but terminates the operation when a read error occurs. Function 0Ah does not terminate on error.

Cont

Function 0Bh Write Hard Disk Sectors and Error Correction Codes

Input:	AH	=	0Bh
	AL	=	Number of sectors to write
	CH	=	Lower eight bits of last cylinder number
	CL	=	Highest two bits of last cylinder number and six bits for beginning sector number.
	DH	=	Head number
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
	ES:BX	=	Buffer segment:offset address
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description Function 0Bh writes the number of sectors specified in AL to the hard disk specified in DL using the head number specified in DH. It also writes a four-byte Error Correction Code (ECC) for each sector. The four-byte ECC must follow the data to be written to each sector.

The data to be written to the drive is stored at the location pointed to in ES:BP. The buffer must contain 512 bytes of data followed by a four-byte ECC, then another 512 bytes of data and another four-byte ECC, and so on.

Cont

Function 0Ch Seek Hard Disk Cylinder

Input:	AH	=	0Ch
	CH	=	Lower eight bits of last cylinder number
	CL	=	Highest two bits of last cylinder number and six bits for beginning sector number.
	DH	=	Head number
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
	ES:BX	=	Buffer segment:offset address
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description Function 0Ch moves the hard disk heads to the specified cylinder but does not transfer data. You do not have to call this function before calling Functions 0Ah Read or 0Bh Write, since functions 0Ah and 0Bh perform a Seek command.

Function 0Dh Reset Hard Disk Controller

Input:	AH	=	0Dh
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description Function 0Dh resets the specified hard disk drive. Unlike Function 00h, this function does not reset the floppy controller.

Cont

Function 10h Test Unit Ready

Input:	AH	=	10h
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description Function 10h determines if the hard disk drive specified in DL is ready.

Function 11h Recalibrate Hard Disk

Input:	AH	=	11h
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description Function 11h recalibrates the specified hard disk drive, places the read/write head at cylinder 0, and returns the drive status in AH.

Cont

Function 14h Perform Internal Controller Diagnostic

Input:	AH	=	14h
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description Function 14h executes a diagnostic self-test routine built into ISA hard disk controllers. This diagnostic routine returns the status and results in AH.

Function 15h Return Drive Type

Input:	AH	=	15h
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
Output:	AH	=	00h No error
		=	03h Drive is a hard disk
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error
	CX:DX	=	Number of 512 byte sectors

Description If AH contains 03h, the drive is a hard disk and CX:DX contains the number of 512-byte sectors.

Cont

Function 41h Check Extension Present

Input:
- AH = 41h
- BX = 55AAh
- DL = 80h Hard Disk Drive C:
- = 81h – FFh are valid. 81h = D:, 82h = E:, etc.

Output:
- AH = 00h No error
- = Extension major version if successful.
- = 01h Invalid function
- = Other values are error codes
- AL = Extension minor version number
- BX = AA55h Drive is installed
- CF = 0 No error
- = 1 Error
- CX = Bit-mapped API information
  - Bits 15-3 Reserved (set to 0)
  - Bit 2 1 Enhanced Disk Drive (EDD) functions 48h and 4Eh supported
  - 0 EDD functions not supported
  - Bit 1 1 Removable media control functions 45h, 46h, 48h, 49h, and INT 15h Function 52h supported
  - 0 Removable media functions not supported
  - Bit 0 1 Extended disk functions 42h, 44h, 47h, and 48h supported
  - 0 Extended disk functions not supported
- DH = Extension version

Description This function returns information about the enhanced IDE API.

Cont

Function 42h Extended Read

Input:	AH	=	42h
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
	DS:SI	=	Disk address packet (see below for format)
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Disk Address Packet Table Format

Offset	Size	Description
00h	Byte	10h (Size of packet)
01h	Byte	Reserved. Must be zero.
02h	Word	Number of blocks to transfer.
04h	Dword	The address of the transfer buffer.
08h	Qword	The absolute address of the starting block number. For non-LBA devices, you must compute (Cylinder * Heads + Selected Head) * Sectors Per Track + Selected Sector -1

Description The disk address packet block count field is set to the number of blocks successfully transferred on return.

Cont

Function 43h Extended Write

Input:	AH	=	43h
	AL	=	Write flags
		=	In version 2.0
		=	Bits 7-1 Reserved
		=	Bit 0 Verify write
		=	In version 2.1
		=	00h Write without verify
			01h Write without verify
			02h Write with verify
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
	DS:SI	=	Disk address packet (see below for format)
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Disk Address Packet Table Format

Offset	Size	Description
00h	Byte	10h (Size of packet)
01h	Byte	Reserved. Must be zero.
02h	Word	Number of blocks to transfer.
04h	Dword	The address of the transfer buffer.
08h	Qword	The absolute address of the starting block number. For non-LBA devices, you must compute (Cylinder * Heads + Selected Head) * Sectors Per Track + Selected Sector -1

Description The disk address packet block count field is set to the number of blocks successfully transferred on return. CF is set and AH contains 01h if a verify is requested but not supported.

Cont

Function 44h Verify Sectors

Input:	AH	=	44h
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
	DS:SI	=	Disk address packet (see below for format)
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Disk Address Packet Table Format

Offset	Size	Description
00h	Byte	10h (Size of packet)
01h	Byte	Reserved. Must be zero.
02h	Word	Number of blocks to transfer.
04h	Dword	The address of the transfer buffer.
08h	Qword	The absolute address of the starting block number. For non-LBA devices, you must compute (Cylinder *Heads + Selected Head) * Sectors Per Track + Selected Sector -1

Description The disk address packet block count field is set to the number of blocks successfully verified on return.

Cont

Function 45h Lock Drive

Input:	AH	=	45h
	AL	=	Operation
		=	In version 2.0
		=	00h Lock media in drive
			01h Unlock media
			02h Check lock status
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
Output:	AH	=	00h No error
		=	Other values are error codes
	AL	=	Lock status
		=	00h Unlocked
	CF	=	0 No error
		=	1 Error

Description This function must be supported for all removable drives number above 80h. Each drive can have up to 255 locks. The media is not physically unlocked until all locks have been removed..

Function 46h Eject Media

Input:	AH	=	46h
	AL	=	00h Reserved
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description This function ejects the removable media in the drive specified by the contents of DL.

Cont

Function 47h Extended Seek

Input:	AH	=	47h
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
	DS:SI	=	Disk address packet (see below for format)
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description This function performs and extended Seek operation on the drive specified by the contents of DL.

Cont

Function 48h Get Drive Parameters This function returns the drive parameters to a buffer specified in DS:SI for the drive specified in DL.

Input:	AH	=	48h
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
	DS:SI	=	Buffer that will contain the drive parameters
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error
	DS:SI	=	Filled drive parameter buffer

Drive Parameters

Offset	Size	Description
00h	Word	Call size of buffer. The size in bytes of the buffer passed as input as well as the buffer returned, including the size of this field (2 bytes). 001Ah Version 1.x 001Eh Version 2.x
02h	Word	Information flags Bits 15-7 Reserved. Must be zero. Bit 6 CHS information set to maximum supported values, not to the current media. Bit 5 Drive can be locked (required for removable drives greater than or equal to 80h) Bit 4 Drive support change line (required for removable drives greater than or equal to 80h) Bit 3 Write with verify supported Bit 2 Removable drive Bit 1 Valid cylinder, head, and sectors per track information Bit 0 DMA boundary errors are handled transparently
04h	Dword	Number of physical cylinders on the drive
08h	Dword	Number of physical heads on the drive
0Ch	Dword	Number of physical sectors per track on the drive
10h	Qword	Total number of sectors on the drive
18h	Word	Bytes per sector
1Ah	Dword	Pointer to extended drive parameter table, in segment:offset form. If the content is FFFF:FFFFh, this field is invalid. AMIBIOS points INT 41h for drive 80h and INT 46h for drive 81h. AMIBIOS maintains the pointers to the parameter tables for drives greater than 81h internally. Issue INT 13h function 48h to retrieve parameters for the specified drive

Cont

Function 49h Extended Media Change

Input:	AH	=	49h
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
Output:	AH	=	00h No error
		=	06h Removable disk cartridge may have changed
		=	Other values are error codes
	CF	=	0 Removable disk cartridge has not changed
		=	1 Removable disk cartridge may have changed

Description This function returns CF set and 06h in AH if the removable disk cartridge has changed.

Cont

Function 4Ah AH = 00h Initiate Disk Emulation for Bootable CD-ROM

Input:	AH	=	4Ah
	AL	=	00h
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
	DS:SI	=	Specification packet (see below for format)
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 Successful
		=	1 Error (drive is not in emulation mode).

Bootable CD-ROM Drive Specification Packet

Offset	Size	Description
00h	Byte	Size of packet in bytes (13h)
01h	Byte	Bit 7 The image contains SCSI drivers Bit 6 The image contains an ATAPI driver Bits 5-4 Reserved. Must be zero. Bits 3-0 Media type 0000 No emulation 0001 1.2 MB diskette 0010 1.44 MB diskette 0011 2.88 MB diskette 0100 Hard disk drive C:
02h	Byte	Drive number 00h Floppy image 80h Bootable hard disk drive 81h FhNon-bootable drive
03h	Byte	CD-ROM controller number
04h	Dword	Logical block address of disk image to emulate
08h	Word	Device specification for IDE drive: Bit 0 Drive is slave For SCSI drive: Bits 15-8 Bus number Bits 7-0 LUN and PUN
0Ah	Word	Segment of 3 KB buffer for caching CD-ROM reads
0Ch	Word	Load segment for initial boot image 0000h Load at segment 07C0h
0Eh	Word	# of 512 byte virtual sectors to be loaded (only valid if AH = 4Ch).
10h	Byte	Low byte of cylinder count for INT 13h Function 08h
11h	Byte	Sector count. High bits of cylinder count for INT 13h Function 08h.
12h	Byte	Head count for INT 13h Function 08h.

Cont

Function 4Bh AL = 00h Terminate Disk Emulation for Bootable CD-ROM

Input:	AH	=	4Bh
	AL	=	00h
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
		=	7Fh Terminate all drive emulation
	DS:SI	=	Empty specification packet
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 Successful
		=	1 Error (drive is not in emulation mode).
	DS:SI	=	Specification packet filled (see the specification packet format on the previous page).

Description This function terminates disk emulation for a bootable CD-ROM drive.

Function 4Bh AL = 01h Get Status of Bootable CD-ROM

Input:	AH	=	4Bh
	AL	=	01h
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
		=	7Fh Terminate all drive emulation
	DS:SI	=	Empty specification packet
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 Successful
		=	1 Error (drive is not in emulation mode).
	DS:SI	=	Specification packet filled (see the specification packet format on the previous page).

Description This function returns the status of the bootable CD-ROM drive in DS:SI.

Cont

Function 4Ch Start Disk Emulation and Boot a Bootable CD-ROM Drive

Input:	AH	=	4Ch
	AL	=	00h
	DS:SI	=	Empty specification packet
Output:	AH	=	Nothing returned if successful
		=	Other values are error codes
	CF	=	0 Successful
		=	1 Error (drive is not in emulation mode).

Description This function begins the disk emulation process and boots from the bootable CD-ROM drive.

Function 4Dh Return Boot Catalog for Bootable CD-ROM Drive

Input:	AH	=	4Dh
	DS:SI	=	Command Packet. See below for format.
Output:	AH	=	Nothing returned if successful
		=	Other values are error codes
	CF	=	0 Successful
		=	1 Error (drive is not in emulation mode).

Bootable CD-ROM Drive Boot Catalog Format

Offset	Size	Description
00h	Byte	Size of packet in bytes (08h).
01h	Byte	Number of sectors in the boot catalog to read
02h	Dword	Buffer for boot catalog
06h	Word	First sector in the boot catalog to be transferred

Description This function returns the boot catalog for the bootable CD-ROM drive.

Cont

Function 4Eh Set Hardware Configuration

Input:	AH	=	4Eh
	AL	=	Function
		=	00h Enable prefetch
		=	01h Disable prefetch
		=	02h Set maximum PIO transfer mode
		=	03h Set PIO mode 0
		=	04h Set the default PIO transfer mode
		=	05h Enable INT 13h DMA maximum mode
		=	06h Disable INT 13h DMA
	DL	=	80h Hard Disk Drive C:
		=	81h – FFh are valid. 81h = D:, 82h = E:, etc.
Output:	AH	=	00h Successful
		=	Other values are error codes
	AL	=	Status
			00h The command affected only the specified drive
			01h Other devices were affected
	CF	=	0 Successful
		=	1 Error

Description

This function sets the hardware configuration specified in AL for the drive specified in DL. If AL = 06h, PIO modes may be disabled for the specified device or all devices on the specified IDE controller. If AL = 02h, 03h, or 04h INT 13h DMA is disabled.

Because DMA and PIO modes are mutually exclusive, selecting DMA disables PIO (for either the specified device or all devices on that controller), and selecting PIO disables DMA.

INT 13h Floppy Disk Service

Functions The INT 13h Floppy Disk functions discussed in this chapter are:

Function	Title
00h	Reset Floppy Disk Drive
01h	Return Disk Drive Status
02h	Read Floppy Disk Sectors
03h	Write Disk Sectors
04h	Verify Disk Sectors
05h	Format Disk Track
08h	Return Disk Parameters
15h	Return Drive Type
16h	Disk Media Change Status
17h	Set Floppy Disk Type
18h	Set Floppy Disk Type Before Format

Cont

Error Codes For most floppy and hard disk drive functions, the following error codes are returned through register AH. All error codes appear in AH.

Code	Description	Code	Description
00h	No error	0Dh	Invalid number of sectors for format on hard disk drive
01h	Function invalid	0Eh	Control data address mark found on hard disk drive
02h	Address mark not found	0Fh	DMA arbitration level out of range
03h	Write attempted on write protected floppy disk	10h	Read error (uncorrectable CRC or ECC)
04h	Sector not found	11h	ECC data error corrected on hard disk drive
05h	Hard disk drive reset failed	20h	Error in floppy disk controller
06h	Floppy disk replaced	40h	Track not found on seek
07h	Hard disk drive parameter is corrupt	80h	Timeout, drive not responding
08h	DMA overflow occurred	AAh	Hard disk drive not ready
09h	DMA crossed 64 KB segment boundary	BBh	Unknown error on hard disk drive
0Ah	Hard disk drive bad sector flag	CCh	Hard disk drive write error occurred
0Bh	Hard disk drive bad track flag	E0h	Hard disk drive status register error
0Ch	Floppy disk media type not found	FFh	Hard disk drive sense operation failed

Cont

INT 13h Coding Conventions For most INT 13h functions, the sector number is placed in CL and the cylinder number in CH.

INT 40h Revector for Floppy Functions INT 13h handles both floppy disk and hard disk drive BIOS functions. If the computer has a hard disk drive, the floppy disk service routine actually resides at INT 40h. All BIOS floppy functions are actually revectored to INT 40h and then executed.

Function 00h Reset Disk Drive

Input:	AH	=	00h
	DL	=	00h Floppy Drive A:
		=	01h Floppy Drive B:
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description INT 13h Function 00h should be used when an error follows a disk operation. Function 00h resets the disk controller and recalibrates the floppy drives attached to the floppy controller. If Function 00h is issued for a hard disk drive, the floppy controller is reset, then the hard disk drive controller is reset.

Cont

Function 01h Return Disk Drive Status

Input:	AH	=	01h
	DL	=	00h Floppy Drive A:
		=	01h Floppy Drive B:
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description INT 13h Function 01h can be used to read the status of the last disk operation.

Function 02h Read Disk Sectors

Input:	AH	=	02h
	AL	=	Number of sectors to read
	CH	=	Track number
	CL	=	Starting sector number
	DL	=	00h Floppy Drive A:
		=	01h Floppy Drive B:
	ES:BX	=	Buffer segment:offset address
Output:	AH	=	00h No error
		=	Other values are error codes
	AL	=	Number of sectors actually read
	CF	=	0 No error
		=	1 Error

Description Function 02h reads the specified number of sectors from a specified track on one side of a disk. The sector(s) are read from the disk and then stored in a buffer at address ES:BX.

Cont

Function 03h Write Disk Sectors

Input:	AH	=	03h
	AL	=	Number of sectors to write
	CH	=	Track number
	CL	=	Starting sector number
	DH	=	00h Side 0
		=	01h Side 1
	DL	=	00h Floppy Drive A:
		=	01h Floppy Drive B:
	ES:BX	=	Buffer segment:offset address
Output:	AH	=	00h No error
		=	Other values are error codes
	AL	=	Number of sectors actually written
	CF	=	0 No error
		=	1 Error

Description Function 03h writes the number of sectors in AL to the track in CH on one side (specified in DH) of a floppy disk. The beginning sector number is in CL. The data written to the sectors comes from the buffer at address ES:BX.

Function 04h Verify Disk Sectors

Input:	AH	=	04h
	AL	=	Number of sectors to verify
	CH	=	Track number
	CL	=	Starting sector number
	DL	=	00h Floppy Drive A:
		=	01h Floppy Drive B:
Output:	AH	=	00h No error
		=	Other values are error codes
	AL	=	Number of sectors actually verified
	CF	=	0 No error
		=	1 Error

Description Function 04h verifies that the ECC code at the end of each sector is correct for the data contained in that sector.

Cont

Function 05h Format Disk Cylinder

Input:	AH	=	05h
	AL	=	Number of sectors to format
	CH	=	Track number
	DH	=	00h Side 0
			01h Side 1
	DL	=	00h Floppy Drive A:
		=	01h Floppy Drive B:
	ES:BX	=	Buffer segment:offset address
Output:	AH	=	00h No error
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description Function 05h formats an entire track or cylinder on a disk. A buffer containing sector information is passed in ES:BX.

The buffer contains a four-byte record for each sector in the track, in the following format:

Byte	Description
0	Track number
1	Head number
2	Logical sector number
3	Number of bytes per sector: 00h 128 bytes per sector 01h 256 bytes per sector 02h 512 bytes per sector (ISA and EISA Standard) 03h 1024 bytes per sector

Call INT 13h function 17h or 18h to set the floppy disk media type before invoking this function.

Cont

Function 08h Return Disk Parameters

Input:	AH	=	08h
	DL	=	00h Floppy Drive A:
		=	01h Floppy Drive B:
	ES:BX	=	Buffer segment:offset address
Output:	AH	=	00h No error
		=	Other values are error codes
	BH	=	Floppy drive type
		=	01h 360 KB, 40 track 5¼"
		=	02h 1.2 MB, 80 track 5¼"
		=	03h 720 KB, 80 track 3½"
		=	04h 1.44 MB, 80 track 3½"
		=	05h 2.88 MB, 80 track 3½"
			06h 2.88 MB, 80 track 3½"
	CF	=	0 No error
		=	1 Error
	CH	=	Lower 8 bits of last cylinder number
	CL	=	High two bits of last cylinder number and six bits for last sector number
	DH	=	Last head number
	DL	=	Number of disk drives
	ES:DI	=	Address of disk parameter table from BIOS

Description INT 13h Function 08h retrieves the parameters for a floppy disk drive from the ROM BIOS.

00h is returned in BL when:

- the drive type is known but CMOS RAM data is invalid or not present,
- the CMOS RAM battery is low, or
- the CMOS RAM checksum value is corrupt.

If the specified drive is not installed, all returned values are 00h.

The value for AX, ES, BX, CX, DH, DI is 0, and DL is the number of drives present if any of the following is true:

- the specified drive number is invalid,
- the specified drive type is unknown and CMOS RAM is not present,
- the CMOS RAM battery is low or the CMOS RAM checksum is invalid, or
- the drive type is unknown and the drive type stored in CMOS RAM is invalid.

Cont

Function 15h Return Drive Type

Input:	AH	=	15h
	DL	=	00h Floppy Drive A:
		=	01h Floppy Drive B:
Output:	AH	=	00h No drive present
		=	01h Drive does not support change line
		=	02h Drive supports change line.
		=	Other values are error codes
	BH	=	Floppy drive type
		=	01h 360 KB, 40 track 5¼"
		=	02h 1.2 MB, 80 track 5¼"
		=	03h 720 KB, 80 track 3½"
		=	04h for 1.44 MB, 80 track 3½"
		=	05h 2.88 MB, 80 track 3½"
		=	06h 2.88 MB, 80 track 3½"
	CF	=	0 No error
		=	1 Error

Description Function 15h determines if floppy disk change line information is available.

Function 16h Disk Media Change Status

Input:	AH	=	16h
	DL	=	00h Floppy Drive A:
		=	01h Floppy Drive B:
Output:	AH	=	00h No floppy disk (media) change
		=	01h Invalid floppy disk parameter
		=	06h Floppy disk was changed since last access
		=	80h Floppy disk drive not ready
		=	Other values are error codes
	BH	=	Floppy drive type
		=	01h 360 KB, 40 track 5¼"
		=	02h 1.2 MB, 80 track 5¼"
		=	03h 720 KB, 80 track 3½"
		=	04h 1.44 MB, 80 track 3½"
		=	05h 2.88 MB, 80 track 3½"
		=	06h 2.88 MB, 80 track 3½"
	CF	=	0 No error
		=	1 Error

Description Function 16h determines if a media change was made since the last floppy disk access.

Cont

Function 17h Set Floppy Disk Type

Input:	AH	=	17h
	AL	=	Floppy disk format
		=	01h 320/360 KB floppy in 320/360 KB drive
		=	02h 360 KB floppy in 1.2 MB drive
		=	03h 1.2 MB floppy in 1.2 MB drive
		=	04h 720 KB floppy in 720 KB drive
			05h 1.44 MB floppy in 1.44 MB drive
			06h 2.88 MB floppy in 2.88 MB drive
	DL	=	00h Floppy Drive A:
		=	01h Floppy Drive B:
Output:	AH	=	00h No floppy disk (media) change
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error

Description Function 17h sets the format of a disk in a floppy drive and sets the data rate and media type if the drive supports the disk change line.

Function 18h Set Floppy Disk Type before Format

Input:	AH	=	18h
	CH	=	Maximum number of tracks
	CL	=	Sectors per track
	DL	=	00h Floppy Drive A:
		=	01h Floppy Drive B:
Output:	AH	=	00h Specified track and sector data supported
		=	Other values are error codes
	CF	=	0 No error
		=	1 Error
	ES:DI	=	Pointer to parameter table if AH is 00h

Description This function sets the media type before formatting a floppy disk. Call INT 13h Function 18h before INT 13h Function 05h.

INT 14h Serial Communications Service

INT 14h accesses and controls the serial ports. AMIBIOS permits up to four serial ports to be configured. These serial ports are initialized to the following starting I/O port addresses: 3F8h, 2F8h, 3E8h, and 2E8h.

The default values for the serial I/O port addresses used in an AMIBIOS can be modified via AMIBCP.

INT 14h Functions INT 14h Functions 00h through 03h are defined in ISA standards. Functions 04h and 05h are defined in PS/2 standards and are only available in an AMIBIOS dated 080891 (August 8, 1991) or later.

Function	Title
00h	Initialize Serial Port
01h	Send Character to Serial Port
02h	Receive Character from Serial Port
03h	Read Serial Port Status
04h	Extended Initialize Serial Port
05h	Extended Serial Port Control

Serial Service I/O Ports The Serial port I/O consists of eight contiguous I/O ports, in the following format

I/O Port	R/W	Description
Base	Write	Transmitter Holding Register (contains the character to be sent). Bit 0, the least significant bit, is sent first. Bits 7–0 Contains data bits 7–0 when the Divisor Latch Access Bit (DLAB) is 0.
Base	Read	Receiver Buffer Register (contains the received character). Bit 0, the least significant bit, is received first. Bits 7–0 Contains data bits 7–0 when the Divisor Latch Access Bit (DLAB) is 0.
Base	Read/ Write	Divisor Latch, low byte. Both divisor latch registers store the data transmission rate divisor. Bits 7–0 Bits 7–0 of divisor when DLAB is 1.
Base + 1	Read/ Write	Divisor Latch, high byte. Bits 7–0 Bits 15–8 of data transmission rate divisor when DLAB is 1.

I/O Port	R/W	Description
Base + 1	Read/ Write	Interrupt Enable Register. Permits the serial port controller interrupts to enable the chip interrupt output signal. Bit 3 1 Modem status interrupt enable Bit 2 1 Receiver line status interrupt enable Bit 1 1 Transmitter holding register empty interrupt enable Bit 0 1 Received data available interrupt enable when DLAB is 0.
Base + 2	Read	Interrupt ID Register. Information about a pending interrupt is stored here. When the ID register is addressed, the highest priority interrupt is held and no other interrupts are acknowledged until the CPU services the interrupt with the highest priority. Bits 2–1 The pending interrupt with the highest priority. 11 Receiver Line Status Interrupt, priority is highest. 10 Received Data Available, second in priority. 01 Transmitter Holding Register Empty, third priority. 00 Modem Status Interrupt, fourth in priority. Bit 0 0 Interrupt pending 1 No interrupt is pending.
Base + 3	Read/ Write	Line Control Register Bit 7 Divisor Latch Access Bit (DLAB) 0 Access receiver buffer, transmitter holding register, and interrupt enable register. 1 Access Divisor Latch of baud rate generator. Bit 6 1 Set Break Control. Serial output is forced to spacing state and remains there. Bit 5 1 Stick Parity. Bit 4 1 Even Parity Select. Bit 3 1 Parity Enable. Bit 2 Number of Stop Bits per Character. 0 One stop bit. 1 1½ stop bits if 5-bit word length is selected (2 stop bits if 6, 7, or 8-bit word length is selected). Bits 1–0 Number of Lines per character 00 5-Bit word length. 01 6-Bit word length. 10 7-Bit word length. 11 8-Bit word length.

I/O Port	R/W	Description
Base + 4	Read/ Write	Modem Control Register Bit 4 1 Loopback mode for diagnostic testing of serial port. The output from the transmitter shift register is looped back to the receiver shift register input Transmitted data is immediately received so the CPU can verify the transmit and receive data serial port paths. Bit 3 1 Force OUT2 interrupt Bit 2 1 Force OUT1 active. Bit 1 1 Force Request To Send active Bit 0 1 Force Data Terminal Ready active.
Base + 5	Read Only	Line Status Register Bit 6 1 Transmitter shift and holding registers empty. Bit 5 1 Transmitter holding register empty. The controller is ready to accept a new character to send. Bit 4 1 Break interrupt. The received data input is held in the zero bit state longer than the transmission time of the start bit + data bits + parity bits + stop bits. Bit 3 1 Framing error. The stop bit that follows the last parity or data bit is zero. Bit 2 1 Parity error. The character has incorrect parity. Bit 1 1 Overrun error. A character was sent to the receiver buffer before the previous character in the buffer could be read, which destroys the previous character. Bit 0 1 Data Ready. A complete incoming character has been received and sent to the receiver buffer register.
Base + 6	Read Only	Modem Status Register Bit 7 1 Data Carrier Detect Bit 6 1 Ring Indicator Bit 5 1 Data Set Ready Bit 4 1 Clear To Send Bit 3 1 Delta Data Carrier Detect Bit 2 1 Trailing Edge Ring Indicator Bit 1 1 Delta Data Set Ready Bit 0 1 Delta Clear To Send
Base + 7	R/W	Reserved

Function 00h Initialize Serial Port

Input: AH = 00h

AL = Parameter byte

= Bits 7–5 Data transmission rate

000	110	001	150
010	300	011	600
100	1200	101	2400
110	4800	111	9600

Bits 4–3 Parity

00 No parity
01 Odd parity
10 No parity
11 Even parity

Bit 2 Number of stop bits

0 One bit
1 Two bits

Bits 1–0 Data length

00 Five bits
01 Six bits
10 Seven bits
11 Eight bits

DX = Serial port number. Index to serial port base table at 40:00h.

= 00h COM1 01h COM2

= 02h COM3 03h COM4

Output: AH = Line status

Bit 7 1 Timeout
Bit 6 1 Transmit Shift Register is empty.
Bit 5 1 Transmit Holding Register is empty.
Bit 4 1 Break signal detected.
Bit 3 1 Framing error detected.
Bit 2 1 Parity error detected.
Bit 1 1 Data overrun error detected.
Bit 0 1 Receive data ready.

AL = Modem status

= Bit 7 1 Receive line signal detected.
Bit 6 1 Ring indicator.
Bit 5 1 Data set ready.
Bit 4 1 Clear to send.
Bit 3 1 Delta receive line signal detect.
Bit 2 1 Trailing edge ring indicator.
Bit 1 1 Delta data set ready.
Bit 0 1 Delta clear to send.

Description Function 00h initializes the specified serial port with the parameters in the parameter byte (AL). It returns the line status and the modem status.

Cont

Function 01h Send Character to Serial Port

Input:	AH	=	01h
	AL	=	Character to be sent
	DX	=	Serial Port Number. Index to serial port base table at 40:00h.
		=	00h COM 1
		=	01h COM 2
		=	02h COM 3
		=	03h COM 4
Output:	AH	=	Line status Bit 7 1 Timeout Bit 6 1 Transmit Shift Register is empty. Bit 5 1 Transmit Holding Register is empty. Bit 4 1 Break signal detected. Bit 3 1 Framing error detected. Bit 2 1 Parity error detected. Bit 1 1 Data overrun error detected. Bit 0 1 Receive data ready.
	AL	=	Character sent

Description Function 01h sends a character to the serial port. It returns the line status.

Cont

Function 02h Receive Character from Serial Port

Input:	AH	=	02h
	DX	=	Serial Port Number. Index to serial port base table at 40:00h.
		=	00h COM 1
		=	01h COM 2
		=	02h COM 3
		=	03h COM 4
Output:	AH	=	Line status Bit 7 1 Timeout Bit 6 1 Transmit Shift Register is empty. Bit 5 1 Transmit Holding Register is empty. Bit 4 1 Break signal detected. Bit 3 1 Framing error detected. Bit 2 1 Parity error detected. Bit 1 1 Data overrun error detected. Bit 0 1 Receive data ready.
	AL	=	Character received

Description Function 02h receives a character in AL from the serial port. Function 02h also returns the port status in AH.

Cont

Function 03h Return Serial Port Status

Input: AH = 03h
DX = Serial Port Number. Index to serial port base table at 40:00h.
= 00h COM 1 01hCOM2
= 02h COM 03hCOM4

Output: AH = Line status
Bit 7 1 Timeout
Bit 6 1 Transmit Shift Register is empty.
Bit 5 1 Transmit Holding Register is empty.
Bit 4 1 Break signal detected.
Bit 3 1 Framing error detected.
Bit 2 1 Parity error detected.
Bit 1 1 Data overrun error detected.
Bit 0 1 Receive data ready.
AL = Modem status
Bit 7 1 Receive line signal detected.
Bit 6 1 Ring indicator.
Bit 5 1 Data set ready.
Bit 4 1 Clear to send.
Bit 3 1 Delta receive line signal detect.
Bit 2 1 Trailing edge ring indicator.
Bit 1 1 Delta data set ready.
Bit 0 1 Delta clear to send.

Description Function 03h returns the status of the specified serial port. Function 03h differs from function 00h. Function 03h has no initialization process, but Function 00h does.

Cont

Function 04h Extended Initialize Serial Port

Input:

	AH	=	04h
	AL	=	00h No break signal
		=	01h Break signal
	BH	=	00h No parity 01h Odd parity
		=	02h Even parity 03h Stick parity odd
		=	04h Stick parity even
	BL	=	00h 1 Stop bit
		=	01h 2 Stop bits if data length is 6, 7, or 8 bits
		=	10h 1½ Stop bits if data length is 5 bits
	CH	=	00h Data length is 5 bits 01h Data length is 6 bits
		=	02h Data length is 7 bits 03h Data length is 8 bits
	CL	=	00h 110 bps 01h 150 bps 02h 300 bps
		=	03h 600 bps 04h 1200 bps 05h 2400 bps
		=	06h 4800 bps 07h 9600 bps 08h 19200 bps
		=	09h 28800 bps 0Ah 57600 bps 0Bh 115200
	DX	=	Serial Port Number. Index to base port at 40:00h.
		=	00h COM 1 01h COM2
		=	02h COM3 03h COM4

Output:

AH	=	Line status
		Bit 7 1 Timeout
		Bit 6 1 Transmit Shift Register is empty.
		Bit 5 1 Transmit Holding Register is empty.
		Bit 4 1 Break signal detected.
		Bit 3 1 Framing error detected.
		Bit 2 1 Parity error detected.
		Bit 1 1 Data overrun error detected.
		Bit 0 1 Receive data ready.
AL	=	Modem status
		Bit 7 1 Receive line signal detected.
		Bit 6 1 Ring indicator.
		Bit 5 1 Data set ready.
		Bit 4 1 Clear to send.
		Bit 3 1 Delta receive line signal detect.
		Bit 2 1 Trailing edge ring indicator.
		Bit 1 1 Delta data set ready.
		Bit 0 1 Delta clear to send.

Description Function 04h initializes the specified serial port with the parameters in the parameter byte (AL). Function 04h returns the line and modem status if a modem is attached. Function 04h differs from Function 00h because the input parameters are different.

Cont

Function 05h Extended Serial Port Control Subfunction AL = 00h Read from Modem Control Register

Input:

AH	=	05h	
AL	=	00h Read from modem control register	
DX	=	Serial Port Number. Index to serial port base table at 40:00h.	
	=	00h	COM 1
	=	01h	COM 2
	=	02h	COM 3
	=	03h	COM 4

Output:

AH	=	Line status	
		Bit 7	1 Timeout
		Bit 6	1 Transmit Shift Register is empty.
		Bit 5	1 Transmit Holding Register is empty.
		Bit 4	1 Break signal detected.
		Bit 3	1 Framing error detected.
		Bit 2	1 Parity error detected.
		Bit 1	1 Data overrun error detected.
		Bit 0	1 Receive data ready.
AL	=	Modem status	
		Bit 7	1 Receive line signal detected.
		Bit 6	1 Ring indicator.
		Bit 5	1 Data set ready.
		Bit 4	1 Clear to send.
		Bit 3	1 Delta receive line signal detect.
		Bit 2	1 Trailing edge ring indicator.
		Bit 1	1 Delta data set ready.
		Bit 0	1 Delta clear to send.
BL	=	Modem control register	
		Bits 7–5	Reserved
		Bit 4	1 Loop for testing.
		Bit 3	1 OUT2.
		Bit 2	1 OUT1.
		Bit 1	1 Request to send.
		Bit 0	1 Data terminal ready.

Description Function 05h reads or sets the modem control register for the specified serial port.

Cont

Function 05h Extended Serial Port Control Subfunction AL = 01h Write to Modem Control Register

Input:	AH	=	05h
	AL	=	01h Write to modem control register
	DX	=	Serial Port Number. Index to serial port base table at 40:00h.
		=	00h COM 1
		=	01h COM 2
		=	02h COM 3
		=	03h COM 4
Output:	AH	=	Line status
			Bit 7 1 Timeout
			Bit 6 1 Transmit Shift Register is empty.
			Bit 5 1 Transmit Holding Register is empty.
			Bit 4 1 Break signal detected.
			Bit 3 1 Framing error detected.
			Bit 2 1 Parity error detected.
			Bit 1 1 Data overrun error detected.
			Bit 0 1 Receive data ready.
	AL	=	Modem status
			Bit 7 1 Receive line signal detected.
			Bit 6 1 Ring indicator.
			Bit 5 1 Data set ready.
			Bit 4 1 Clear to send.
			Bit 3 1 Delta receive line signal detect.
			Bit 2 1 Trailing edge ring indicator.
			Bit 1 1 Delta data set ready.
			Bit 0 1 Delta clear to send.
	BL	=	Modem control register
			Bits 7–5 Reserved
			Bit 4 1 Loop for testing.
			Bit 3 1 OUT2.
			Bit 2 1 OUT1.
			Bit 1 1 Request to send.
			Bit 0 1 Data terminal ready.

Description Function 05h reads or sets the modem control register for the specified serial port.

INT 15h System Services

Category	Description and INT 15h Functions
EISA Support	INT 15h Function D8h, subfunctions 00h through 04h, are defined only in the EISA specifications and are supported in the EISA BIOS.
Multitasking Services	The BIOS provides six hooks that can be used by programmers: INT 15h Functions 80h, 81h, 82h, 85h, 90h, and 91h are defined in the ISA standard and are available in the BIOS but do not perform any service. Software developers can trap or redirect the vectors of these interrupt functions to point to programmer-supplied service routines. No routines for these functions are provided in the BIOS.
Protected Mode Services	Function 87h Move Block provides a way to move large blocks of information from conventional to extended memory. Function 89h switches to protected mode.
Wait Routines	Functions 83h and 86h provide wait control. Function 86h does not return control to the calling program until a specified interval completes. Function 83h returns control to the caller immediately but sets a bit when a predetermined wait period is finished.
System Information	Function C1h returns the extended BIOS data area address. Function C0h returns system configuration data. Functions 88h and E2h return the extended memory size.
Advanced Power Management	Function 53h provides power management functions that conform to the APM specification.
PS/2 Mouse Support	Functions 4Fh, C1h, and C2h are defined in the PS/2 specification. AMIBIOS supports some PS/2-defined operations, including all PS/2 mouse operations. The programmer can invoke these mouse functions if the computer includes the necessary hardware as well as the appropriate American Megatrends Keyboard Controller BIOS (version KF, KH, Megakey, or later). Function C2h PS/2 Mouse Support is supported in all AMIBIOS dated August 8, 1991 (080891) or later.
Tape Cassette Services	The only INT 15h function on the original PC was cassette tape I/O. In AMIBIOS, these functions (00h, 01h, 02h, and 03h) are not supported. If called, the BIOS sets the Carry Flag in the FLAGS register and returns AH = 86h (no cassette present). You can trap Functions 00h – 03h and substitute your own code.
Joystick support	Function 84h provides joystick support for up to two joysticks.

INT 15h Systems Services Functions

Function	Title
24h	Enable/Disable/Query Gate A20
4Fh	Keyboard Intercept
53h	Advanced Power Management AL 00h APM Installation Check AL 01h APM Real Mode Interface Connect AL 02h APM 16-Bit Protected Mode Interface Connect AL 03h APM 32-Bit Protected Mode Interface Connect AL 04h APM Interface Disconnect AL 05h CPU Idle AL 06h CPU Busy AL 07h Set Power State AL 08h Enable Power Management AL 09h Restore BIOS Power-On Defaults AL 0Ah Get Power Status AL 0Bh Get PM Event AL 0Ch Get Power State AL 0Dh Enable Device Power Management AL 80h OEM-Defined APM Functions BH 7FhAPM Installation Check BH 00h-7Eh OEM-Defined Function BH 80h-FFh OEM-Defined Function
80h	Device Open (replaced by BIOS user routine)
81h	Device Close (replaced by BIOS user routine)
82h	Program Termination (replaced by BIOS user routine)
83h	Set Event Wait Interval
84h	Joystick Support DX 001h Read Current Switch Settings DX 01h Read Resistive Inputs
85h	System Request Key (replaced by BIOS user routine)
86h	Wait
87h	Move Block
88h	Return Extended Memory Size (up to 64 MB).
89h	Switch to Protected Mode
90h	Device Busy Loop (replaced by BIOS user routine)
91h	Interrupt Complete (replaced by BIOS user routine)
C0h	Return System Configuration Parameters
C1h	Return Address of Extended BIOS Data Area
C2h	PS/2 Mouse Support
C3h	Fail-Safe Timer
D8h	EISA Support
E2h	Return Extended Memory Size (over 64 MB).
E8h	ACPI Access

Function 24h Disable Gate A20

Mode: Real Mode

Input:	AH	=	24h	
	AL	=	00h	
Output:	AH	=	00h	Successful
		=	01h	Keyboard controller is in secure mode
			86h	Function not supported
	CF	=	0	Successful
		=	1	Unsuccessful

Description This function disables the Gate A20 address line.

Function 24h Enable Gate A20

Mode: Real Mode

Input:	AH	=	24h	
	AL	=	01h	
Output:	AH	=	00h	Successful
		=	01h	Keyboard controller is in secure mode
			86h	Function not supported.
	CF	=	0	Successful
		=	1	Unsuccessful

Description This function enables the Gate A20 address line.

Cont

Function 24h Get Gate A20 Status

Mode: Real Mode

Input:	AH	=	24h	
	AL	=	02h	
Output:	AH	=	00h	Successful
		=	01h	Keyboard controller is in secure mode
			86h	Function not supported.
			FFh	Keyboard controller did not become ready within C000h read attempts.
	AL	=	Current Gate A209 state	
		=	00h	Disabled
			01h	Enabled
			FFh	Keyboard controller did not become ready within C000h read attempts.
	CF	=	0	Successful
		=	1	Unsuccessful

Description This function retrieves the Gate A20 address line status.

Function 24h Query Gate A20 Support

Mode: Real Mode

Input:	AH	=	24h	
	AL	=	03h	
Output:	AH	=	00h	Successful
		=	01h	Keyboard controller is in secure mode
			86h	Function not supported.
			FFh	Keyboard controller did not become ready within C000h read attempts.
	BX	=	Current Gate A209 status	
		=	0000h	Supported on keyboard controller
		=	0001h	Supported with bit 1 of I/O port 0092h
		=	000Fh	Additional data is available. The location of this data is not yet defined.
			FFh	Keyboard controller did not become ready within C000h read attempts.
	CF	=	0	Successful
		=	1	Unsuccessful

Description This function queries the Gate A20 address line status and reports the results in BX.

Cont

Function 4Fh PS/2 Keyboard Intercept

Mode: Real Mode

Input:	AH	=	4Fh
	AL	=	Scan code
Output:	AL	=	Scan code
	CF	=	0 Scan code processed but should not go to keyboard buffer.
		=	1 Scan Code processed or modified and should go to keyboard buffer

Description INT 09h calls this function each time a key is pressed. Function 4Fh can be used to search the data from a keyboard. If the specified scan code is found, the routine provided by the programmer is executed. This routine can modify the scan code.

Function 52h Media Eject Intercept

Mode: Real Mode

Input:	AH	=	52h
	DL	=	80h Hard Disk Drive C:
			81h – FFh are valid. 81h = D:, 82h = E:, etc.
Output:	AL	=	Error Code
		=	B1h
		=	B3h
	CF	=	0 OK to eject media.
		=	1 Not OK to eject media.

Description This function is part of the INT 13h Extended IDE functions. You can call this function before calling INT 13h Function 46h Eject Media to make sure that the media is in a state where it can be ejected.

Cont

APM Functions INT 15h Function 53h provides subfunctions that support the Advanced Power Management specification.

APM Error Codes The error codes that can be returned in AH upon completion of an APM subfunction are:

Code in AH	Description
01h	Power management functionality disabled
02h	Interface connection already in effect
03h	Interface not connected
04h	Real mode interface not connected
05h	16-bit protected-mode interface already connected
06h	16-bit protected-mode interface not supported
07h	32-bit protected-mode interface already connected
08h	32-bit protected-mode interface not supported
09h	Unrecognized device ID
0Ah	Invalid parameter value in CX
0Bh	(APM v1.1) interface not engaged
0Ch	(APM v1.2) function not supported
0Dh	(APM v1.2) Resume timer disabled
0Eh – 1Fh	Reserved for other interface and general errors
20h –3Fh	Reserved for CPU errors
40h – 5Fh	Reserved for device errors
60h	Cannot enter requested state
61h-7Fh	Reserved for other system errors
80h	No power management events pending
81h – 85h	Reserved for other power management event errors
86h	APM not present
87h – 9Fh	Reserved for other power management event errors
A0h – FEh	Reserved
FFh	Undefined

Cont

Function 53h Subfunction AL = 00h APM Installation Check

Mode: Real Mode

Input:	AH	=	53h
	AL	=	00h
	BX	=	Power Device ID
		=	0000h BIOS

Output:	AH	=	1 APM major version number (in BCD)
	AL	=	1 APM minor version number (in BCD)
	BH	=	P (in ASCII)
	BL	=	M (in ASCII)
	CF	=	0 APM is supported by the BIOS.
		=	1 APM is not supported by the BIOS.
	ECX	=	APM Flags

Bit 31	BIOS Power Management is disabled (v1.2).
Bits 30-21	Reserved
Bit 20	A *CPU Idle* call does not slow the processor clock speed or stop the clock.
Bits 19–5	Reserved
Bit 4	BIOS power management disengaged (v1.1)
Bit 3	BIOS power management is disabled
Bit 2	CPU idle call reduces processor speed
Bit 1	32-bit protected mode interface supported
Bit 0	16-bit protected mode interface supported

Description This subfunction allows the APM driver (the calling program) to find the supported APM specification. It also specifies if the system BIOS supports APM.

Cont

Function 53h Subfunction AL = 01h APM Real Mode Interface Connect

Mode: Real Mode

Input:	AH	=	53h
	AL	=	01h
	BX	=	Power Device ID
		=	0000h BIOS
Output:	AH	=	Error code if not successful
		=	00h Successful
		=	02h A real mode interface connection is already established.
		=	05h A 16-bit protected mode interface connection is already established.
		=	07h A 32-bit protected mode interface connection is already established.
		=	09h Device ID unrecognized.
	CF	=	0 Successful
		=	1 Not successful
	CX	=	APM 16-bit data segment (real mode segment base address)

Description This subfunction initializes the interface between the APM Driver (the calling program) and the BIOS. Before the interface is established, the BIOS provides OEM-defined power management. Once the interface is defined, the APM driver and the BIOS coordinate power management activities.

Cont

Function 53h Subfunction AL = 02h APM 16-Bit Protected Mode Interface Connect

Mode: Real Mode

Input:	AH	=	53h
	AL	=	02h
	BX	=	Power Device ID
		=	0000h BIOS
Output:	AH	=	Error code if not successful
		=	00h Successful
		=	02h A real mode interface connection is already established.
		=	05h A 16-bit protected mode interface connection is already established.
		=	07h A 32-bit protected mode interface connection is already established.
		=	09h Device ID unrecognized.
	AX	=	APM 16-bit code segment or the real mode segment base address
	BX	=	Offset of the entry point into the BIOS
	CF	=	0 Successful
		=	1 Not successful
	CX	=	APM 16-bit data segment (real mode segment base address)
	DI	=	BIOS code segment length
	SI	=	BIOS data segment length

Description This subfunction initializes the 16-bit protected mode interface between the APM Driver (the calling program) and the BIOS. This function must be invoked from real mode. This interface allows a routine making a call in protected mode to invoke BIOS functions without switching into real or virtual 8086 mode.

Cont

Function 53h Subfunction AL = 02h APM 16-Bit Protected Mode Interface Connect, cont

Initializing Descriptors The APM 16-bit protected mode interface uses two consecutive segment/selector descriptors as a 16-bit code and data segment.

The calling program must initialize these descriptors with the segment base and length information returned by this call. The selectors can be in the GDT or LDT and must be valid when the BIOS is called in protected mode.

The code segment descriptor must specify protection level 0. The BIOS function must be invoked with CPL = 0 so the BIOS can execute privileged instructions.

The calling program invokes the BIOS using the 16-bit interface by making a FAR CALL to the code segment selector that the calling program initialized and the offset returned in BX from this call.

The calling program must supply a stack that can handle both the BIOS and potential interrupt handlers.

The calling program's stack becomes active when interrupts are enabled in the BIOS functions. The BIOS does not switch stacks when interrupts are enabled, including the NMI.

The BIOS 16-bit protected mode interface must be called with a 16-bit stack.

When a BIOS function is called in protected mode, the current I/O permission bitmap must permit access to the I/O ports that the BIOS uses.

Cont

Function 53h Subfunction AL = 03h APM 32-Bit Protected Mode Interface Connect

Mode: Real Mode

Input:	AH	=	53h
	AL	=	03h
	BX	=	Power Device ID
		=	0000h BIOS
Output:	AH	=	Error code if not successful
		=	00h Successful
		=	02h A real mode interface connection is already established.
		=	05h A 16-bit protected mode interface connection is already established.
		=	07h A 32-bit protected mode interface connection is already established.
		=	08h The 32-bit protected mode interface is not supported.
		=	09h Device ID unrecognized.
	AX	=	APM 16-bit code segment or the real mode segment base address
	BX	=	Offset of the entry point into the BIOS
	CF	=	0 Successful
		=	1 Not successful
	CX	=	APM 16-bit data segment (real mode segment base address)
	DI	=	BIOS code segment length
	DX	=	APM data segment (real mode segment base address)
	EBX	=	Offset of the entry point into the BIOS
	SI	=	BIOS data segment length

Description This real mode subfunction initializes the 32-bit protected mode interface between the APM Driver (the calling program) and the BIOS. This interface allows a protected mode routine to invoke BIOS functions without switching to real or Virtual 8086 mode.

Cont

Function 53h Subfunction AL = 03h APM 32-Bit Protected Mode Interface Connect, cont

Initializing Descriptors The APM 32-bit protected mode interface uses three consecutive segment/selector descriptors as 32-bit code, 16-bit code, and data segment. Both the 32-bit and 16-bit code segment descriptors are needed because the BIOS 32-bit interface can call other BIOS routines.

The calling program must initialize these descriptors with the segment base and length information returned by this call. The selectors can be in the GDT or LDT and must be valid when the BIOS is called in protected mode.

The code segment descriptor must specify protection level 0. The BIOS function must be invoked with CPL = 0 so the BIOS can execute privileged instructions.

The calling program invokes the BIOS using the 32-bit interface by making a FAR CALL to the 32-bit code segment selector that the calling program initialized and the offset returned in EBX from this call.

The calling program must supply a stack that can handle both the BIOS and potential interrupt handlers.

The calling program's stack becomes active when interrupts are enabled in the BIOS functions. The BIOS does not switch stacks when interrupts are enabled, including the NMI.

The BIOS 32-bit protected mode interface must be called with a 32-bit stack.

When a BIOS function is called in protected mode, the current I/O permission bitmap must permit access to the I/O ports that the BIOS uses.

Cont

Function 53h Subfunction AL = 04h APM Interface Disconnect

Mode: Real Mode, 16-Bit Protected Mode, 32-Bit Protected Mode

Input:	AH	=	53h
	AL	=	04h
	BX	=	Power Device ID
		=	0000h BIOS
Output:	AH	=	Error code if not successful
		=	00h Successful
		=	03h Interface disconnected
		=	09h Device ID unrecognized.
	CF	=	0 Successful
		=	1 Not successful

Description This subfunction:

- disconnects the BIOS and the APM driver,
- restores the BIOS default functions, and
- returns control of power management to the BIOS.

All power management parameters in effect when APM is disconnected will remain in effect.

Cont

Function 53h Subfunction AL = 05h CPU Idle

Mode: Real Mode, 16-bit Protected Mode, 32-bit Protected Mode

Input:	AH	=	53h
	AL	=	05h
	BX	=	Power Device ID
		=	0000h BIOS
Output:	AH	=	Error code if not successful
		=	00h Successful
		=	03h Interface disconnected
		=	0Bh (APM v1.1) interface not engaged
	CF	=	0 Successful
		=	1 Not successful

Description Call this function to inform the BIOS that the computer is idle. The BIOS will suspend the computer until the next system event, which is usually an interrupt. This function permits the BIOS to implement power-saving actions, such as a CPU HLT instruction or slowing the CPU clock.

Cont

Function 53h Subfunction AL = 06h CPU Busy

Mode: Real Mode, 16-bit Protected Mode, 32-bit Protected Mode

Input:	AH	=	53h
	AL	=	06h
	BX	=	Power Device ID
		=	0000h BIOS
Output:	AH	=	Error code if not successful
		=	00h Successful
		=	03h Interface disconnected
		=	0Bh (APM v1.1) interface not engaged
	CF	=	0 Successful
		=	1 Not successful

Description You need to invoke this subfunction only if *INT 15h AH = 53h Subfunction AL = 05h CPU Idle* was previously invoked. Check bit 2 in CX after invoking *Function 53h Subfunction AL = 00h APM Installation Check* to determine if the BIOS will slow the clock during an *INT 15h AH = 53h Subfunction AL = 05h CPU Idle* call.

This subfunction tells the BIOS that the computer is busy. The BIOS restores the CPU clock speed to full speed.

Do not call this function when the CPU is already operating at full speed. While it is not illegal to do so, it adds overhead.

Cont

Function 53h Subfunction AL = 07h Set Power State

Mode: Real Mode, 16-bit Protected Mode, 32-bit Protected Mode

Input:	AH	=	53h
	AL	=	07h
	BX	=	Power Device ID
		=	0000h BIOS
		=	0001h All devices under APM
		=	01*xx*h Display (*xx* = unit number). Use *xx* = FF to specify all devices in a class.
		=	02*xx*h Secondary storage
		=	03*xx*h Parallel ports
		=	04*xx*h Serial ports
		=	E000h – EFFFh OEM-defined device IDs
	CX	=	Power state
		=	0000h APM enabled (not supported for Device ID 0001h)
		=	0001h Standby
		=	0002h Suspend
		=	0003h Off
		=	0004h – 001Fh Reserved system states
		=	0020h – 003Fh OEM-defined system states
		=	0040h – 007Fh OEM-defined device states
		=	0080h – FFFFh Reserved device states
Output:	AH	=	Error code if not successful
		=	00h Successful
		=	01h Power management disabled
		=	03h Interface disconnected
		=	09h Device ID unrecognized
		=	0Ah Parameter value out of range
		=	0Bh (APM v1.1) interface not engaged
		=	60h Unable to enter requested state
	CF	=	0 Successful
		=	1 Not successful

Description Sets the specified power state for the specified device.

Example The following parameters enter Standby mode. The calling program invokes this function in response to a *System Standby Request Notification* from the BIOS and can also invoke this function any time the computer is in Standby mode. When any interrupt occurs, full on mode is entered.

```
BX = 0001h All devices under APM
CX = 0001h System standby
```

Cont

Function 53h Subfunction AL = 08h Enable Power Management

Mode: Real Mode, 16-bit Protected Mode, 32-bit Protected Mode

Input:	AH	=	53h
	AL	=	08h
	BX	=	Power Device ID
		=	0001h All devices under APM
		=	FFFFh All devices under APM as specified in the APM 1.0 specification
	CX	=	Function code
		=	0000h Disable power management
		=	0001h Enable power management
Output:	AH	=	Error code if not successful
		=	00h Successful
		=	01h Power management disabled
		=	03h Interface disconnected
		=	09h Device ID unrecognized
		=	0Ah Parameter value out of range
		=	0Bh (APM v1.1) interface not engaged
	CF	=	0 Successful
		=	1 Not successful

Description This subfunction enables (or disables) automatic power down. When disabled, the BIOS does not automatically power devices down, enter Suspend State, enter the Standby State, or perform any power-saving steps in response to Function 53h Subfunction AL = 05h CPU Idle calls.

Cont

Function 53h Subfunction AL = 09h Restore BIOS Power-On Defaults

Mode: Real Mode, 16-bit Protected Mode, 32-bit Protected Mode

Input:	AH	=	53h
	AL	=	09h
	BX	=	Power Device ID
		=	0001h All devices under APM
		=	FFFFh All devices under APM as specified in the APM 1.0 specification.
	CX	=	Function code
		=	0000h Disable power management
		=	0001h Enable power management
Output:	AH	=	Error code if not successful
		=	00h Successful
		=	03h Interface disconnected
		=	09h Device ID unrecognized
		=	0Bh (APM v1.1) interface not engaged
	CF	=	0 Successful
		=	1 Not successful

Description This subfunction reinitializes the BIOS power-on default values.

Cont

Function 53h Subfunction AL = 0Ah Get Power Status

Mode: Real Mode, 16-bit Protected Mode, 32-bit Protected Mode

Input:
AH = 53h
AL = 0Ah
BX = Power Device ID
= 0001h All devices under APM
= FFFFh All devices under APM as specified in the APM 1.0 specification.

Output:
AH = Error code if not successful
= 00h Successful
= 09h Device ID unrecognized
= 0Ah Invalid parameter value in CX
BH = Line status
= 00h Offline
= 01h Online
= 02h On backup power
= FFh Unknown
BL = Battery status
= 00h High
= 01h Low
= 02h Critical
= 03h Charging
= FFh Unknown
CF = 0 Successful
= 1 Not successful
CL = Remaining battery life (percentage of charge)
= 0 through 100 % of full charge
= 255 Unknown
DX = Remaining battery life (time units)
Bit 15 0 Time unit is seconds
1 Time unit is minutes
Bits 14-0 Number of seconds or minutes of battery life left
0000h–7FFFh Valid number
FFFFh Unknown

Description This subfunction returns the current system power status.

Cont

Function 53h Subfunction AL = 0Bh Get PM Event

Mode: Real Mode, 16-bit Protected Mode, 32-bit Protected Mode

Input:
AH = 53h
AL = 0Bh

Output:
AH = Error code if not successful
= 00h Successful
= 03h Interface disconnected
= 0Bh (APM v1.1) interface not engaged
= 80h No power management events pending

BX = Event Code
0001h System standby request (v1.0)
0002h System suspend request (v1.0)
0003h Normal resume system notification (v1.0)
0004h Battery low notification (v1.0)
0006h Power status change notification (v1.1)
0007h Update time notification (v1.1)
0008h Critical system suspend notification (v1.1)
0009h User system standby request notification (v1.1)
000Bh System standby resume notification (v1.1)
000Ch Capabilities change notification (v1.2)
00Dh-00FFh Reserved system events (v1.2)
0100h-01FFh Reserved device events (v1.2)
0200-02FFh OEM-defined APM events
0300-FFFFh Reserved

CF = 0 Successful
= 1 Unsuccessful

Description This subfunction returns the next power management event or indicates that no power management events are pending. Power management events can apply to a device or to the APM system.

This subfunction should be invoked until no power management events are pending or an error occurs.

Cont

Function 53h Subfunction AL = 0Ch Get Power State

Mode: Real Mode, 16-bit and 32-bit Protected Mode

Input:	AH	=	53h
	AL	=	0Ch
	BX	=	Power Device ID
		=	0001h All devices under APM
		=	01*xx*h Display (*xx* is the unit number). *xx* = FFh includes all devices in a class.
		=	02*xx*h Secondary storage (*xx* is unit number).
		=	03*xx*h Parallel ports (*xx* is unit number).
		=	04*xx*h Serial ports (*xx* is unit number).
		=	04*xx*h Serial ports (*xx* is unit number).
		=	E00h – EFFFh OEM-defined power device IDs
Output:	AH	=	Error code if not successful
		=	00h Successful
		=	01h Power management disabled
		=	09h Device ID unrecognized
	CF	=	0 Successful
		=	1 Not successful
	CX	=	0000h APM enabled
		=	0001h Standby
		=	0001h Suspend
		=	0003h Off
		=	0004h – 001Fh Reserved system states
		=	0020h – 003Fh OEM-defined system states
		=	0040h – 007Fh OEM-defined device states
		=	0080h – FFFFh Reserved device states

Description This subfunction returns the device power state for a specific Device ID. *0001h All devices under APM* or *all devices in a class* (*x*FF*x*h) is returned for the specified Power Device ID when that device has been used in an *AL = 07h Set Power State* call. When the power device ID has not been used in an *AL = 07h Set Power State* call, this function is unsuccessful and returns AH = 09h Device ID unrecognized. Use this subfunction to find out if BIOS power management is enabled for a device. This subfunction returns AH = 01h if BIOS power management is disabled.

Cont

Function 53h Subfunction AL = 0Dh Enable Device Power Management

Mode: Real Mode, 16-bit Protected Mode, 32-bit Protected Mode

Input:

AH	=	53h	
AL	=	0Dh	
BX	=	Power Device ID	
	=	0001h	All devices under APM
	=	01*xx*h	Display (*xx* is the unit number). *xx* = FFh includes all devices in a class.
	=	02*xx*h	Secondary storage (*xx* is unit number).
	=	03*xx*h	Parallel ports (*xx* is unit number).
	=	04*xx*h	Serial ports (*xx* is unit number).
	=	04*xx*h	Serial ports (*xx* is unit number).
	=	E00h – EFFFh	OEM-defined power device IDs.
CX	=	Function code	
	=	0000h	Disable power management
	=	0001h	Enable power management

Output:

AH	=	Error code if not successful	
	=	00h	Successful
	=	01h	Power management disabled
	=	03h	Interface disconnected
	=	09h	Device ID unrecognized
	=	0Ah	Parameter value out of range
	=	0Bh	(APM v1.1) interface not engaged
CF	=	0	Successful
	=	1	Not successful

Description This subfunction enables (or disables) automatic power down for the specified device. When disabled, the BIOS does not automatically power the device down.

Cont

Function 53h Subfunction AL = 80h BH = 7Fh APM Installation Check (OEM-Defined APM Functions)

Mode: Real Mode, 16-bit Protected Mode, 32-bit Protected Mode

Input:	AH	=	53h
	AL	=	80h
	BH	=	7Fh OEM APM installation check
Output:	AH	=	Error code if not successful
		=	03h Interface disconnected
	BX	=	OEM ID
	CF	=	0 Successful
		=	1 Not successful
	CX	=	Optional OEM-Specific information
	DX	=	Optional OEM-Specific information

Description Call this subfunction to find out if the BIOS supports OEM hardware-dependent functions.

Function 53h Subfunction AL = 80h BH = OEM-Defined Function Code

Mode: Real Mode, 16-bit Protected Mode, 32-bit Protected Mode

Input:	AH	=	53h
	AL	=	80h
	BH	=	7Eh OEM-Defined function code
		=	80h – FFh OEM-Defined function code
Output:	AH	=	Error code if not successful
		=	03h Interface disconnected
	BX	=	OEM ID
	CF	=	0 Successful
		=	1 Not successful
	CX	=	Optional OEM-Specific information
	DX	=	Optional OEM-Specific information

Description: Call this subfunction to access OEM product-specific APM functions.

Cont

Power Management Error Codes

AH	Description	Generated by Value in AL
01h	Power management disabled	07h Set Power State 08h Enable Power Management 0Ah Get Power Status 0Dh Enable Device Power Management
02h	Real mode interface connection already established	01h APM Real Mode Interface Connect 02h APM 16-Bit Protected Mode Interface Connect 03h APM 32-Bit Protected Mode Interface Connect
03h	Interface disconnected	04h APM Interface Disconnect 05h CPU Idle 06h CPU Busy 07h Set Power State 08h Enable Power Management 09h Restore BIOS Power-On Defaults 0Bh Get PM Event 0Dh Enable Device Power Management 80h OEM APM Function
05h	16-bit protected mode interface already established	01h APM Real Mode Interface Connect 02h APM 16-Bit Protected Mode Interface Connect 03h APM 32-Bit Protected Mode Interface Connect
06h	16-bit protected mode interface unsupported	02h APM 16-Bit Protected Mode Interface Connect
07h	32-bit protected mode interface already established	01h APM Real Mode Interface Connect 02h APM 16-Bit Protected Mode Interface Connect 03h APM 32-Bit Protected Mode Interface Connect
08h	32-bit protected mode interface unsupported	03h APM 32-Bit Protected Mode Interface Connect
09h	Device ID Unrecognized	01h APM Real Mode Interface Connect 02h APM 16-Bit Protected Mode Interface Connect 03h APM 32-Bit Protected Mode Interface Connect 04h APM Interface Disconnect 07h Set Power State 08h Enable Power Management 09h Restore BIOS Power-On Defaults 0Ah Get Power Status 0Ch Get Power State 0Dh Enable Device Power Management
0Ah	Parameter values out of range	07h Set Power State 08h Enable Power Management 0Dh Enable Device Power Management
60h	Unable to enter requested state	07h Set Power State
80h	Power management events not pending	0Bh Get PM Event
86h	No APM present.	

Cont

Function 80h Device Open

Input: AH = 80h
BX = Device ID
CX = Process ID

Output: Programmer-defined

Description Functions 80h, 81h, and 82h can be used for multitasking operating systems. The system program manager can trap these interrupt functions and provide individual service routines for these operations. The routine provided for Function 81h should detach a logical device from a specified process.

Function 81h Device Close

Input: AH = 81h
BX = Device ID
CX = Process ID

Output: Programmer-defined

Description Functions 80h, 81h, and 82h can be used to handle multitasking operating systems. The system program manager can trap these interrupt functions and provide individual service routines for these operations. The routine provided by the programmer for Function 81h should detach a logical device from a specified process.

Function 82h Process Termination

Input: AH = 82h
BX = Process ID

Output: Programmer-defined

Description Functions 80h, 81h and 82h can be used to handle multitasking operating systems. The system program manager can trap these interrupt functions and provide individual service routines for these operations. The routine provided by the programmer for Function 82h should terminate a process.

Cont

Function 83h Event Wait

Input:	AH	=	83h
	AL	=	00h Request Wait
		=	01h Cancel Wait
	CX:DX	=	Number of microseconds to wait
	ES:BX	=	Pointer to a flag. The high bit is to be set at the end of the interval specified in CX:DX.
Output:	AH	=	00h Successful
	AL	=	Value written to CMOS RAM Register B if successful.
		=	00h Function is busy
	CF	=	0 No error
		=	1 Function is busy

Description Function 83h sets a flag after a specified number of 盜econds has elapsed. Bit 7 of the first byte at ES:BX is set after the wait has expired. The 盜econds to delay must be a multiple of 976

Function 84h Joystick Support

Input:	AH	=	84h
	DX	=	0000h Read Current Switch Settings
		=	0001h Read Resistive Inputs
Output:			*If DX is set to 0000h:*
	AL	=	Bits 7–4 Switch Settings
		=	Bits 3–0 Reserved
			If DX is set to 0001h:
	AX	=	Joystick A x coordinate
	BX	=	Joystick A y coordinate
	CX	=	Joystick B x coordinate
	DX	=	Joystick B y coordinate
	CF	=	0 No error
		=	1 Value in DX is incorrect

Description Function 84h reads the switches and inputs of a joystick attached via a game adapter. 00h is returned if a joystick is not installed.

Cont

Function 85h SysReq Key Handler

Input:	AH	=	85h	
	AL	=	00h	Key Make (Depressed)
		=	01h	Key Break (Released)
Output:		=	Programmer-defined	

Description A multitasking operating system can use Function 85h to see when the SysReq key is pressed or released. The programmer can trap this function and provide another service routine. The BIOS returns AH = 00h and the Carry Flag is set to 0.

Function 86h Wait Function

Input:	AH	=	86h	
	CX:DX	=	Number of microseconds to wait	
Output:	CF	=	0	No error
		=	1	Error

Description Function 86h delays the computer for a specified number of microseconds.

Cont

Function 87h Move Extended Memory Block

Input:

AH	=	87h	
CX	=	Number of words to move	
ES:SI	=	Address of descriptor table	

Output:

AH	=	00h	No error
	=	01h	RAM Parity Error (Parity Error Cleared)
	=	02h	Exception INT Error
	=	03h	Gate Address 20 (GA20) Failed
CF	=	0	No error
	=	1	Error

Description Function 88h moves data between conventional (DOS) memory and extended memory. It uses a Global Descriptor Table (GDT) in the following format (all offsets are with respect to ES:SI):

Offset	Entry Description
00h – 07h	Dummy entry, should be all zeros.
08h – 0Fh	GDT entry (ES:SI)
10h – 17h	Source GDT entry
18h – 1Fh	Destination GDT entry
20h – 27h	Temporary BIOS CS entry
28h – 2Fh	Temporary SS area

Initialize the source GDT and destination GDT entries. All other entries should be initialized to zero. Interrupts are disabled while this function is performed.

Function 88h Return Size of Extended Memory

Input:

AH	=	88h

Output:

AX	=	Number of contiguous 1 KB blocks of extended memory beginning at absolute address 100000h

Description Function 88h returns the size of extended memory (memory above 1 MB) installed in the computer (up to 64 MB). The number of 1 KB blocks is specified in AX.

Cont

Function 89h Switch to Protected Mode

Input:	AH	=	89h
	BH	=	Offset to Interrupt Descriptor Table that points to the beginning of the first eight hardware interrupts (IRQ 0 – 7).
	BL	=	Offset to Interrupt Descriptor Table that points to the beginning of the last eight hardware interrupts (IRQ 8 – 15).
	ES:SI	=	Address of descriptor table
Output:	AH	=	00h No error
			FFh Error enabling address line 20
	CF	=	0 No error
		=	1 Error

Description Function 89h switches the CPU to protected mode from real mode. In the *IBM PC/AT Technical Reference Manual*, protected mode was called virtual mode.

Global Descriptor Table Initialize a Global Descriptor Table (GDT) as follows. All offsets are with respect to ES:SI.

Offset	Table Entry
00h – 07h	Dummy entry, should be all zeros.
08h – 0Fh	Pointer to GDT.
10h – 17h	Interrupt Descriptor Table (IDT) entry.
18h – 1Fh	Programmer-defined DS entry.
20h – 27h	Programmer-defined ES entry.
28h – 2Fh	Programmer-defined SS entry.
30h – 37h	Programmer-defined CS entry.
38h – 3Fh	Temporary BIOS CS entry.

Initialize the GDT, IDT, DS, ES, SS, and CS entries. The temporary BIOS CS entry should be zero. The dummy entry should be all zeros.

The entry at offset 08h is actually a pointer to the GDT table. Its value consists of the physical address derived from ES:SI (pointer to GDT = ((ES * 10) + SI)) and the segment limit (length of the GDT). For additional information on Global Descriptor Tables, see the *Intel Pentium or i486 Programmers Reference Manual*.

Cont

Function 90h Device Busy Loop

Input: AH = 90h
AL = Device type code
= 00h Hard disk drive
= 01h Floppy disk drive
= 02h Keyboard
= 03h PS/2-type mouse
= 80h Network
= FCh Hard disk reset
= FDh Floppy disk drive motor
= FEh Printer
ES:BX = Pointer to a request block if AL = 80h Fh (a reentrant device).

Output: AH = Programmer-defined

Description Function 90h is provided for system-level device drivers to perform a wait for I/O completion. The service routine is provided by the drivers. Serially reusable devices must be given device types from 00h – 7Fh. Reentrant devices must have a type between 80h and BFh. Wait-only calls that have no corresponding INT 15h Function 91h Interrupt Complete call must have device types C0h – FFh.

Function 91h Interrupt Complete

Input: AH = 91h
AL = Device type code
= 00h Hard disk drive
= 01h Floppy disk drive
= 02h Keyboard
= 03h PS/2-type mouse
= 80h Network
= FCh Hard disk reset
= FDh Floppy disk drive motor
= FEh Printer
ES:BX = Pointer to a request block if AL = 80h Fh (a reentrant device).

Output: AH = Programmer-defined

Description Function 91h is provided for system-level device drivers to signal that I/O has been completed. The service routine is provided by the drivers.

Cont

Function C0h Return System Configuration Parameter

Input: AH = C0h

Output: AH = 00h No error
AL = 86h
CF = 0 No error
= 1 Error
ES:BX = Address of configuration parameter table

Description Function C0h returns a pointer to the System Configuration Table. The format of this table is:

Offset	Initial Value	Description
00h – 01h		Number of Bytes in this table (must be at least 8)
02h	FCh	Model Byte (always FCh).
03h	01h	Submodel Byte (always 01h).
04h		BIOS Revision Level
05h		Feature Information Byte Bit 7 1 DMA channel 3 used. Bit 6 1 Interrupt controllers cascaded. Bit 5 1 Real time clock available. Bit 4 1 Keyboard intercept (INT 15h Function 4Fh) is available. Bits 3–0 Reserved. Should be zeros.
06h – 09h		Reserved

Since this book deals only with AMIBIOS for ISA and EISA computers, the value for byte 02h is FCh and for 03h is 01h. These values are the same for all ISA and EISA computers.

Function C1h Return Address of Extended BIOS Data Area

Input: AH = C1h

Output: CF = 0 No error
AL = 1 Error
ES = Segment part of Extended BIOS Data Area address

Description This function returns the segment part of the extended BIOS data area address.

Cont

Function C2h PS/2 Mouse Support Function C2h, originally defined in the IBM PS/2 specification, controls a PS/2-type mouse or pointing device. Support for a PS/2-type mouse is provided by AMIBIOS if the computer has the proper hardware and an American Megatrends Keyboard Controller BIOS version F (KF), KH, Megakey, or later.

Function C2h Subfunction 00h Enable or Disable Mouse

Input:	AH	=	C2h	
	AL	=	00h	
	BH	=	00h	Disable
		=	01h	Enable
Output:	AH	=	00h	No error
		=	01h	Invalid subfunction number
		=	02h	Invalid input values
		=	03h	Mouse interface error
		=	04h	Resend required
		=	05h	FAR CALL is not installed
	CF	=	0	No error
		=	1	Error

Description INT 15h Function C2h Subfunction 00h enables or disables the mouse.

Cont

Function C2h Subfunction 01h Reset Mouse

Input:	AH	=	C2h	
	AL	=	01h	
Output:	AH	=	00h	No error
		=	01h	Invalid subfunction number
		=	02h	Invalid input values
		=	03h	Mouse interface error
		=	04h	Resend required
		=	05h	FAR CALL is not installed
	CF	=	0	No error
		=	1	Error

Description INT 15h Function C2h Subfunction 01h resets the mouse and sets the sample rate, resolution, and other attributes to the default values. The mouse is also disabled by default. The default settings are:

Parameter	Disabled State
Mouse	Disabled
Sample Rate	100 samples per second
Resolution	4 counts per millimeter
Data package size	unchanged
Scaling	1:1

Cont

Function C2h Subfunction 02h Set Sample Rate

Input:	AH	=	C2h	
	AL	=	02h	
	BH	=	00h	10 samples per second
		=	01h	20 samples per second
		=	02h	40 samples per second
		=	03h	60 samples per second
		=	04h	80 samples per second
		=	05h	100 samples per second (default)
		=	06h	200 samples per second
Output:	AH	=	00h	No error
		=	01h	Invalid subfunction number
		=	02h	Invalid input values
		=	03h	Mouse interface error
		=	04h	Resend required
		=	05h	FAR CALL is not installed
	CF	=	0	No error
		=	1	Error

Description INT 15h Function C2h Subfunction 02h sets the mouse sample rate. The default sample rate is 100 samples per second.

Cont

Function C2h Subfunction 03h Set Resolution

Input:	AH	=	C2h	
	AL	=	03h	
	BH	=	00h	1 count per millimeter
		=	01h	2 counts per millimeter
		=	02h	4 counts per millimeter (default)
		=	03h	8 counts per millimeter
Output:	AH	=	00h	No error
		=	01h	Invalid subfunction number
		=	02h	Invalid input values
		=	03h	Mouse interface error
		=	04h	Resend required
		=	05h	FAR CALL is not installed
	CF	=	0	No error
		=	1	Error

Description INT 15h Function C2h Subfunction 03h sets the mouse resolution rate. The default is 4 counts per millimeter.

Function C2h Subfunction 04h Return Mouse Type

Input:	AH	=	C2h	
	AL	=	04h	
Output:	AH	=	00h	No error
		=	01h	Invalid subfunction number
		=	02h	Invalid input values
		=	03h	Mouse interface error
		=	04h	Resend required
		=	05h	FAR CALL is not installed
	BH	=	Device ID	
	CF	=	0	No error
		=	1	Error

Description This subfunction 04h returns the mouse device ID number.

Cont

Function C2h Subfunction 05h Initialize Mouse Interface

Input:	AH	=	C2h
	AL	=	05h
	BH	=	Data Packet Size (1 to 8, representing 1 – 8 bytes)
Output:	AH	=	00h No error
		=	01h Invalid subfunction number
		=	02h Invalid input values
		=	03h Mouse interface error
		=	04h Resend required
		=	05h FAR CALL is not installed
	CF	=	0 No error
		=	1 Error

Description INT 15h Function C2h Subfunction 05h performs the same operations as Subfunction 01h, but it also sets the data packet size of the mouse interface. The same default values specified in subfunction 01h are used and the packet size must be in BH. The default settings are:

Parameter	Disabled State
Mouse	Disabled
Sample Rate	100 samples per second
Resolution	4 counts per millimeter
Data package size	unchanged
Scaling	1:1

Cont

Function C2h Subfunction 06h Mouse Status or Set Scaling Factor

Input:	AH	=	C2h
	AL	=	06h
	BH	=	00h Return mouse status
		=	01h Set 1:1 scaling factor
		=	02h Set 2:1 scaling factor
Output:	AH	=	00h No error
		=	01h Invalid subfunction number
		=	02h Invalid input values
		=	03h Mouse interface error
		=	04h Resend required
		=	05h FAR CALL is not installed
	BL	=	Status Byte (If BH was 00h, BL is the status byte)
			Bit 7 Reserved
			Bit 6 0 Stream mode is used
			1 Remote mode is used
			Bit 5 0 Mouse disabled
			1 Mouse enabled
			Bit 4 0 1:1 scaling is used
			1 2:1 scaling is used
			Bit 3 Reserved
			Bit 2 1 Left button pressed
			Bit 1 Reserved
			Bit 0 1 Right button pressed
	CF	=	0 No error
		=	1 Error
	CL	=	Resolution rate
		=	00h 1 count per millimeter
		=	01h 2 counts per millimeter
		=	02h 4 counts per millimeter
		=	03h 8 counts per millimeter
	DL	=	Sample rate
		=	0Ah 10 samples per second
		=	14h 20 samples per second
		=	28h 40 samples per second
		=	3Ch 60 samples per second
		=	50h 80 samples per second
			64h 100 samples per second
		=	C8h 200 samples per second

Description This function can be used to ascertain the mouse status or to set the mouse scaling factor.

Cont

Function C2h Subfunction 07h Set Mouse Handler Address

Input:

AH	=	C2h	
AL	=	07h	
ES:BX	=	Address of programmer routine	

Output:

AH	=	00h	No error
	=	01h	Invalid subfunction number
	=	02h	Invalid input values
	=	03h	Mouse interface error
	=	04h	Resend required
	=	05h	FAR CALL is not installed
CF	=	0	No error
	=	1	Error

Description This subfunction attaches a programmer-supplied mouse routine to the BIOS mouse service routine. When the BIOS routine receives data from the mouse, the programmer-supplied routine is called by the BIOS. Place the following four parameters on the stack before calling this function:

Address	Description
SS:SP + 0Ah	Status word Bits 15–8 Reserved Bit 7 Y coordinate has overflowed if set to 1 Bit 6 X coordinate has overflowed if set to 1 Bit 5 Y coordinate is negative if set to 1 Bit 4 X coordinate is negative if set to 1 Bits 3–2 Reserved. Bit 3 should be 1 and Bit 2 should be 0. Bit 1 Right button pressed if set to 1 Bit 0 Left button pressed if set to 1
SS:SP + 08h	x coordinate
SS:SP + 06h	y coordinate
SS:SP + 04h	z coordinate (should be 00h)

The programmer-supplied routine should exit via a far return and must not remove the parameters from the stack.

Cont

Function C3h Fail-Safe Timer Control

Input:	AH	=	C3h	
	AL	=	00h	Disable fail-safe timer
		=	01h	Enable fail-safe timer
	BL	=	Fail-safe timer counter value (01h – FFh)	
Output:	CF	=	0	No error
		=	1	Error

Description INT 15h Function C3h enables or disables the EISA fail-safe timer. When enabled, the value in BX becomes the timer count value. The fail-safe timer is placed in mode 0 operation, the fail-safe timer NMI is enabled, and the value in BL is copied to the BIOS extended data area. CF is set if there is an invalid input.

When disabled, the fail-safe timer value in the BIOS extended data area is cleared.

Function D0h P6 Microcode Update

Input:	AH	=	D0h	
	AL	=	42h	
Output:	CF	=	0	No error
		=	1	Error

Description Issue INT 15h Function D0h subfunction 42h to update the Intel Pentium Pro CPU microcode.

Cont

Function D8h EISA Support Function D8h configures EISA controllers and stores values in EISA Extended CMOS RAM. This function is the only way in which EISA Extended CMOS RAM should be accessed.

This function has four subfunctions primarily used by the EISA Configuration Utility (ECU) with the Configuration (CFG) files supplied by EISA product manufacturers with EISA adapter cards and motherboards.

All EISA subfunctions (00h/80h through 04h/84h) are described in this section. Functions 00 – 04h are used for 16-bit cards. Functions 80h – 84h are used for 32-bit cards. Improper use of these subfunctions could cause an EISA computer to operate erratically.

EISA Extended CMOS RAM EISA-specific configuration data is stored in I/O-mapped EISA Extended CMOS RAM. There must be at least 4 KB of EISA Extended CMOS RAM, in addition to the required 64 bytes of ISA CMOS RAM.

EISA Devices Any controller in an EISA computer can be called an EISA device. There can be up to 64 devices in an EISA computer: 16 physical devices and 48 virtual (logical) devices.

EISA Devices and Slots EISA controllers and EISA devices are the same. EISA slots are used as addresses in EISA computers and are the actual physical expansion slots on the EISA motherboard. EISA devices are addressed by their physical or logical slot number. The EISA motherboard is always Slot 0. The physical slots are 1 through 15.

Cont

EISA Device Number A physical device resides in an actual expansion slot on the EISA motherboard and is numbered 1 through 15. This number is the EISA device number.

EISA Embedded Devices The motherboard can have one or more embedded EISA devices. Embedded device numbers begin after the last physical device number. If the last physical device is 7, the first embedded device is 8.

EISA Virtual Devices A virtual device is often a software device driver that uses system resources but does not physically exist. ISA devices on the motherboard can be virtual devices. Virtual devices are numbered sequentially after the last physical or embedded device. If the last physical or embedded device is 6, the first virtual device is 7.

Device Functions A device can have more than one function. Some standard functions are: memory, serial port, parallel port, and disks.

Cont

Function D8h Subfunction 00h/80h Read Slot Configuration

Input:	AH	=	D8h
	AL	=	00h Use 16-bit addressing
		=	80h Use 32-bit addressing
	CL	=	Slot Number (virtual and embedded devices included)
		=	00h Motherboard
		=	01h Slot 1 through 0Fh Slot 15
Output:	AH	=	00h No error
		=	80h Invalid slot number
		=	81h Invalid function number
		=	82h EISA Extended CMOS RAM is corrupt
		=	83h Slot is empty
		=	86h Invalid BIOS call
		=	87h Invalid system configuration
	CL	=	CFG and Slot Status
			Bit 7 0 Duplicate CFG ID not found.
			Bit 6 0 Product ID was readable.
			Bits 5–4 00 Slot is an expansion slot.
			01 Slot is an embedded device.
			10 Slot is a virtual device.
			Bits 3–0 0000 No duplicate CFG ID found.
			0001 First duplicate CFG ID used.
			0010 Second duplicate CFG ID used.
			… ...
			1111 Fifteenth duplicate CFG ID used.
	BH	=	Major Revision Level of ECU
	BL	=	Minor Revision Level of ECU
	CF	=	No error
		=	1 Error
	CH	=	MSB of CFG checksum
	CL	=	LSB of CFG checksum
	DH	=	Number of device function
	DL	=	Combined Function Information Byte
			Bit 5 1 Slot has one or more port initialization entries.
			Bit 4 1 Slot has one or more port range entries.
			Bit 3 1 Slot has one or more DMA entries.
			Bit 2 1 Slot has one or more IRQ entries.
			Bit 1 1 Slot has one or more memory entries.
			Bit 0 1 Slot has one or more function type entries.
	DI (LSB)	=	Byte 0 of compressed ID
	DI (MSB)	=	Byte 1 of compressed ID
	SI (LSB)	=	Byte 2 of compressed ID
	SI (MSB)	=	Byte 3 of compressed ID

Cont

Function D8h Subfunction 00h/80h Read Slot Configuration, cont

Description INT 15h Function D8h Subfunction 00h returns EISA configuration information for a specified slot by reading information directly from EISA Extended CMOS RAM. The slots can be the motherboard, an adapter card, an embedded device, or a virtual device. Each slot has a corresponding CFG file used by the ECU to configure the slot properly.

Duplicate CFG Files If the computer finds more than one CFG file for the specified slot, a duplicate ID condition occurs and bit 8 of AL is set. Bits 3 — 0 of AL indicate the duplicate ID that was used.

Device ID Number DI and SI contain a four-byte compressed ID number pertaining to the device installed in the specified slot. This number identifies the manufacturer of the device, the device product number, and the product revision number.

Register	Description
DI (LSB)	Bit 7 Reserved, should be zero. Bits 6–2 First character of the manufacturer code. Bits 1–0 First two bits of second character of the manufacturer code.
DI (MSB)	Bits 7–5 Remaining 3 bits of second character of the manufacturer code. Bits 4–0 Third character of the manufacturer code.
SI (LSB)	**Adapter card:** Bits 7–4 First hex digit of the manufacturer's product number. Bits 3–0 Second hex digit of the manufacturer's product number. **Motherboard:** Bits 7–0 Reserved for manufacturer.
SI (MSB)	**Adapter card:** Bits 7–4 Third hex digit of the manufacturer's product number. Bits 3–0 Product revision number **Motherboard:** Bits 7–3 Reserved for manufacturer's use. Bits 2–0 EISA bus version number (001 in initial version). 001 is currently the only standard value defined for this field, but, in practice, EISA motherboard and adapter card manufacturers have been using this field for their own purposes.

Cont

Function D8h Subfunction 01h/81h Read Function Configuration

Input:	AH	=	D8h
	AL	=	01h For 16-bit addressing
		=	81h Use 32-bit addressing
	CH	=	Function Number (from 0 through *m* - 1, where *m* = the contents of DH from Subfunction 00h)
	CL	=	Slot Number (virtual and embedded devices included)
		=	00h Motherboard
		=	01h Slot 1
		=	... ...
		=	0Fh Slot 15
	DS:SI	=	Address of data buffer for 16-bit addressing.
	DS:ESI	=	Address of data buffer for 32-bit addressing.
Output:	AH	=	00h No error
		=	80h Invalid slot number
		=	81h Invalid function number
		=	82h EISA Extended CMOS RAM is corrupt
		=	83h Slot is empty
		=	86h Invalid BIOS call
		=	87h Invalid system configuration
	CF	=	No error
		=	1 Error
	DS:SI	=	Return data buffer address if 16-bit call.
	DS:ESI	=	Return data buffer address if 32-bit call.

Description Function D8h Subfunction 01h reads the specified function information directly from CMOS RAM. The calling software can find the number of functions for a specific device using subfunction 00h (80h).

With subfunction 01h (81h), the caller receives information about each specific device function. This subfunction reads a 320-byte table and then writes this table to the memory buffer address specified in DS:SI. Each block of a variable-length data field describes an individual EISA adapter card.

EISA Configuration Table Function

Offset	Description
00h	First Byte of Compressed ID Bit 7 Reserved, should be zero. Bits 6–2 First character of the manufacturer code. Bits 1–0 First two bits of second character of the manufacturer code.
01h	Second Byte of Compressed ID Bits 7–5 Remaining three bits of second character of the manufacturer code. Bits 4–0 Third character of the manufacturer code.
02h	Third Byte of Compressed ID **Adapter card:** Bits 7–4 First hex digit of the manufacturer's product number. Bits 3–0 Second hex digit of the manufacturer's product number. **Motherboard:** Bits 7–0 Reserved for manufacturer's use.
03h	Fourth Byte of Compressed ID **Adapter card:** Bits 7–4 Third hex digit of the manufacturer's product number. Bits 3–0 Product revision number **Motherboard:** Bits 7–3 Reserved for manufacturer's use. Bits 2–0 EISA bus version number (001 is initial version).

Offset	Description
04h – 05h	ID and Slot Information Byte 0 Bit 7 0 No duplicate ID is present. 1 Duplicate ID found. Bit 6 0 ID is readable. 1 ID is unreadable. Bits 5–4 Device Type 00 Expansion device 01 Embedded device 10 Virtual device Bits 3–0 Number of Duplicate CFG filenames 0000 No duplicate CFG 0001 First duplicate CFG ... ... 1110 Fourteenth duplicate CFG 1111 Fifteenth duplicate CFG Byte 1 Bit 7 0 Configuration is successful. 1 Configuration is unsuccessful. Bits 6–2 Reserved, should be zeros. Bit 1 0 EISA IOCHKERR not supported. 1 EISA IOCHKERR supported. Bit 0 0 EISA ENABLE not supported (adapter card cannot be enabled or disabled). 1 EISA ENABLE supported (adapter card can be enabled or disabled). The EISA specification allows EISA adapter cards to be enabled or disabled via software. If bit 0 of byte 1 above is set, external software can disable the adapter card. Similarly, the availability of IOCHKERR allows external software to check expansion slots for pending errors.
06h – 07h	Revision levels of the CFG overlay files used for a specified slot. Both bytes are 0 if no overlay file exists. Byte 0 Minor revision level of the CFG overlay file. Byte 1 Major revision level of the CFG overlay file.
08h – 21h	Selections made by the ECU. The possible choices for the specified slot function are counted here. The actual names of the choices are specified in the CFG file. Byte 0 Selection 1 Byte 1 Selection 2 ... ... Byte 24 Selection 25 Byte 25 Selection 26

Offset	Description
22h	Slot function information Bit 7 0 Slot function is enabled. 1 Slot function is disabled. Bit 6 1 CFG is using free form data. Bit 5 1 Port initialization entry(s) follows. Bit 4 1 Port range entry(s) follows. Bit 3 1 DMA entry(s) follows. Bit 2 1 IRQ entry(s) follows. Bit 1 1 Memory entry(s) follows. Bit 0 1 Type and subtype string follows.
23h – 62h	80-character ASCII string describing the slot device. The string has types and subtypes. The manufacturer determines the type and subtype format, but the following conventions are often used: Type String COM Communications device KEY Keyboard MEM Memory card MFC Multifunction card MSD Mass storage device NET Network card NPX Math coprocessor OSE Operating system or environment OTH Other PAR Parallel port PTR Pointing device SYS Motherboard VID Video adapter card , Delimiter for type string fragments ; End of type string and beginning of subtype string 0 End of subtype strings The unused part of the 80-character string should be zero (not including the subtype delimiter).

Offset	Description
73h – B1h	Memory Configuration Section. Nine seven-byte entries: Byte 0 Memory Configuration Byte Bit 5 0 Memory is not shared 1 Memory is shared Bits 4–3 00 SYS (base/extended memory) 01 EXP (expanded memory) 10 VIR (virtual memory) 11 OTH (other memory) Bit 1 0 Memory is not cached 1 Memory is cached Bit 0 0 Memory is ROM (read only) 1 Memory is RAM (read and write) Byte 1 Memory Data Size Bits 3–2 Decode Size 00 20 address lines 01 24 address lines 10 32 address lines Bits 1–0 Data Access Size 00 Byte 01 Word (16 bits) 10 Doubleword (32 bits) Bytes 2–4 Starting Memory Address divided by 100h Bytes 5–6 Memory Size divided by 400. If 0000h, memory is 64 MB. The size is specified in 1024 byte increments.
B2h – BFh	Hardware Interrupt Configuration Section. Seven two-byte entries: Byte 0 Bit 6 0 Interrupt is not shared 1 Interrupt is shared Bit 5 0 Interrupt is edge-triggered 1 Interrupt is level-triggered Bits 3–0 Interrupt number 0000 IRQ0 0001 IRQ1 ..... 1110 IRQ14 1111 IRQ15 Byte 1 Reserved, should be zero.

Offset	Description
C0h – C7h	DMA Channel Description Section. Four two-byte entries: Byte 0 Bit 6 0 DMA channel is not shared 1 DMA channel is shared Bits 5–3 Reserved, should be zeros. Bits 2–0 DMA Channel Number 000 Channel 0 001 Channel 1 ..... 110 Channel 6 111 Channel 7 Byte 1 Bits 7–6 Reserved, should be zeros. Bits 5–4 DMA Timing 00 ISA-compatible timing 01 Type A timing 10 Type B timing 11 Type C (Burst) timing Bits 3–2 DMA Transfer Size 00 Byte transfers 01 Word transfers (16 bits) 10 Doubleword transfers (32 bits) Bits 1–0 Reserved, should be zeros.
C8h – 103h	I/O Port Information consists of 20 three-byte entries: Byte 0 Bit 6 0 Port is not shared 1 Port is shared Bit 5 Reserved, should be zero. Bits 4–0 Number of Ports (starting at 0) 00000 One port 00001 Two sequential ports 00010 Three sequential ports ... ... 11110 Thirty-one sequential ports 11111 Thirty-two sequential ports Byte 1 LSB of I/O Port Address Byte 2 MSB of I/O Port Address

Offset	Description
104h – 13Fh	I/O Port Initialization Data Section. Entries vary in length. Byte 0 Initialization Type Bits 6–3 Reserved, should be zeros. Bit 2 0 Write value to port 1 Use both mask and value Bits 1–0 Data Access Size *If Byte 0, bit 2 is 0, the following format is used:* 00 Byte 3 is the initialization value. 01 Byte 3 is the LSB of the initialization value. Byte 4 is the MSB of the initialization value. 10 Byte 3 is the LSB of the initialization value. Byte 4 is the second byte of the initialization value. Byte 5 is the third byte of the initialization value. Byte 6 is the MSB of the initialization value. *If Byte 0, bit 2 is 1, the following format is used:* 00 Byte 3 is the initialization value. Byte 4 is mask value. 01 Byte 3 is the LSB of the initialization value. Byte 4 is the MSB of the initialization value. Byte 5 is the LSB of the mask value. Byte 6 is the MSB of the mask value. 10 Byte 3 is the LSB of the initialization value. Byte 4 is the second byte of the initialization value. Byte 5 is the third byte of the initialization value. Byte 6 is the MSB of the initialization value. Byte 7 is the LSB of the mask value. Byte 8 is the second byte of the mask value. Byte 9 is the third byte of the mask value. Byte 10 is the MSB of the mask value. Byte 1 LSB of Port Address Byte 2 MSB of Port Address

Note: If bit 6 of the Function Information Section (22h) is set, the table is not in the table format described above, but uses free-form data. Entries through Type and Subtype (23h) are the same, but starting at 73h, the data in the table is in the board manufacturer's proprietary format.

Cont

Function D8h Subfunction 02h (82h) Clear EISA CMOS RAM

Input:	AH	=	D8h
	AL	=	02h For 16-bit addressing
		=	82h Use 32-bit addressing
	BH	=	Major revision number of ECU
	BL	=	Minor revision number of ECU
Output:	AH	=	00h No error
		=	84h Error while clearing CMOS RAM
		=	86h Invalid BIOS call
		=	88h ECU is not supported
	AL	=	Major revision number of ECU supported by BIOS (if AH = 88h).
	CF	=	No error
		=	1 Error

Description Function D8h Subfunction 02h clears EISA Extended CMOS RAM. This routine does not clear the ISA CMOS RAM, which contains the date, time, hard disk drive type, and basic system configuration.

Cont

Function D8h Subfunction 03h (83h) Write to EISA CMOS RAM

Input:	AH	=	D8h
	AL	=	03h (if CS specifies 16-bit addressing)
		=	83h (if CS specifies 32-bit addressing)
	CX	=	Length of table (if 0, then slot is empty)
	DS:SI	=	Address of data buffer (16-bit addressing)
	DS:ESI	=	Address of data buffer (32-bit addressing)
Output:	AH	=	00h No error
		=	84h Error while clearing CMOS RAM
		=	85h CMOS RAM is full
		=	86h Invalid BIOS call
		=	87h EISA configuration is locked
	AL	=	Major revision number of ECU supported by BIOS (if AH = 88h).
	CF	=	No error
		=	1 Error

Description Function D8h Subfunction 03h writes the configuration data specified in the data buffer pointed to by DS:SI to EISA Extended CMOS RAM. This function does not write to ISA CMOS RAM, which contains the basic system parameters. The data to be written to EISA Extended CMOS RAM should begin at address DS:SI (DS:ESI if using 32-bit addressing) for the length specified in CX. The last two bytes in the table are reserved for the checksum of the CFG file to be used.

EISA Configuration Data Table

The format for the EISA configuration data at DS:SI (DS:ESI) is:

Offset	Description
00h	First Byte of Compressed ID Bit 7 Reserved, should be zero. Bits 6–2 First character of the manufacturer code. Bits 1–0 First two bits of second character of the manufacturer code.
01h	Second Byte of Compressed ID Bits 7–5 Remaining three bits of second character of the manufacturer code. Bits 4–0 Third character of the manufacturer code.
02h	Third Byte of Compressed ID **Adapter card:** Bits 7–4 First hex digit of the manufacturer's product number. Bits 3–0 Second hex digit of the manufacturer's product number. **Motherboard:** Bits 7–0 Reserved for manufacturer's use.
03h	Fourth Byte of Compressed ID **Adapter card:** Bits 7–4 Third hex digit of the manufacturer's product number. Bits 3–0 Product revision number **Motherboard:** Bits 7–3 Reserved for manufacturer's use. Bits 2–0 EISA bus version number (001 is initial version).

Offset	Description
04h – 05h	ID and Slot Information Byte 0 Bit 7 0 No duplicate ID is present. 1 Duplicate ID found. Bit 6 0 ID is readable. 1 ID is unreadable. Bits 5–4 Device Type 00 Expansion device 01 Embedded device 10 Virtual device Bits 3–0 Number of Duplicate CFG filenames 0000 No duplicate CFG 0001 First duplicate CFG ..... 1110 Fourteenth duplicate CFG 1111 Fifteenth duplicate CFG Byte 1 Bit 7 0 Configuration is successful. 1 Configuration is unsuccessful. Bits 6–2 Reserved, should be zeros. Bit 1 0 EISA IOCHKERR not supported. 1 EISA IOCHKERR supported. Bit 0 0 EISA ENABLE not supported (adapter card cannot be enabled or disabled). 1 EISA ENABLE supported (adapter card can be enabled or disabled). The EISA specification allows EISA adapter cards to be enabled or disabled via software. If bit 0 of byte 1 above is set, external software can disable the adapter card. Similarly, the availability of IOCHKERR allows external software to check expansion slots for pending errors.
06h – 07h	Revision levels of the CFG overlay files used for a specified slot. Both bytes are 0 if no overlay file exists. Byte 0 Minor revision level of the CFG overlay file. Byte 1 Major revision level of the CFG overlay file.
The rest of this table is repeated once for every EISA function. There can be *1 through n* EISA functions. Most EISA Adapter Cards have more than one function. The last function is empty and has a length of 0. All functions must fit in 340 bytes.	
2 bytes, but they do not count as part of the function length.	Function Length. The length does not include these two bytes or the checksum at the end of EISA CMOS RAM. The last function must be set to length 0. Byte 0 LSB of the length of the following function entry. Byte 1 MSB of the length of the following function entry.

Offset	Description
2 to 27 bytes for each function.	Selections made by the ECU. The possible choices for the specified slot function are counted here. The actual names of the choices are specified in the CFG file. Byte 0 Selection 1 Byte 1 Selection 2 ... ... Byte 24 Selection 25 Byte 25 Selection 26
1 byte for each function.	Slot function information Bit 7 0 Slot function is enabled. 1 Slot function is disabled. Bit 6 CFG is using free form data if set. Bit 5 Port initialization entry(s) follows if set. Bit 4 Port range entry(s) follows if set. If not set, the port range section is length 0. Bit 3 DMA entry(s) follows if set. If not set, the DMA entry section is length 0. Bit 2 IRQ entry(s) follows if set. If not set, the IRQ entry section is length 0. Bit 1 Memory entry(s) follows if set. If not set, the memory section is length 0. Bit 0 Type and subtype string follow if set.
2 – 81 bytes for each function.	Byte 0 Length of the following field Bytes 1-80 A 1 - 80-character ASCII string describing the slot device. The string has types and subtypes. For example, TYPE=COM, AMI; COM1 would be: 0ChCOM,AMI;COM1 The manufacturer determines the type and subtype format, but the conventions are: **Type String** COM Communications device KEY Keyboard MEM Memory card MFC Multifunction card MSD Mass storage device NET Network card NPX Math coprocessor OSE Operating system or environment OTH Other PAR Parallel port PTR Pointing device SYS Motherboard VID Video adapter card , Delimiter for type string fragments ; End of type string and beginning of subtype string 0 End of subtype strings The unused part of the 80-character string should be zero (not including the subtype delimiter).

Offset	Description
7 to 63 bytes for each function.	Memory Configuration Section. 0 to Nine seven-byte entries: Byte 0 Memory Configuration Byte Bit 7 0 Last entry 1 More entries follow Bit 6 Reserved, should be zero. Bit 5 0 Memory is not shared 1 Memory is shared Bits 4–3 00 SYS (base/extended memory) 01 EXP (expanded memory) 10 VIR (virtual memory) 11 OTH (other memory) Bit 1 0 Memory is not cached 1 Memory is cached Bit 0 0 Memory is ROM (read only) 1 Memory is RAM (read and write) Byte 1 Memory Data Size Bits 7-4 Reserved, should be zeros. Bits 3–2 Decode Size 00 20 address lines 01 24 address lines 10 32 address lines Bits 1–0 Data Access Size 00 Byte 01 Word (16 bits) 10 Doubleword (32 bits) Bytes 2–4 Starting Memory Address divided by 100h Bytes 5–6 Memory Size divided by 400. If 0000h, memory size is 64 MB. The size is specified in 1024 byte increments.
2 - 14 bytes for each function.	IRQ Configuration Section. 1 to 7 two-byte entries. Byte 0 Bit 7 0 Last entry 1 More entries follow Bit 6 0 Interrupt is not shared 1 Interrupt is shared Bit 5 0 Interrupt is edge-triggered 1 Interrupt is level-triggered Bit 4 Reserved (should be 0) Bits 3–0 Interrupt number 0000 IRQ0 0001 IRQ1 ..... 1110 IRQ14 1111 IRQ15 Byte 1 Reserved, should be zero.

Offset	Description
0 - 4 entries for each function. 2 - 8 bytes for each entry.	DMA Channel Description Section. 0 – 4 two-byte entries. Byte 0 Bit 7 0 Last entry 1 More entries follow Bit 6 0 DMA channel is not shared 1 DMA channel is shared Bits 5–3 Reserved, should be zeros. Bits 2–0 DMA Channel Number 000 Channel 0 001 Channel 1 ..... 110 Channel 6 111 Channel 7 Byte 1 Bits 7–6 Reserved, should be zeros. Bits 5–4 DMA Timing 00 ISA-compatible timing 01 Type A timing 10 Type B timing 11 Type C (Burst) timing Bits 3–2 DMA Transfer Size 00 Byte transfers 01 Word transfers (16 bits) 10 Doubleword transfers (32 bits) Bits 1–0 Reserved, should be zeros.
1 to 20 entries for each function. 3 to 60 bytes for each entry.	I/O Port Information consists of 0 to 20 three-byte entries: Byte 0 Bit 7 0 Last entry 1 More entries follow Bit 6 0 Port is not shared 1 Port is shared Bit 5 Reserved, should be zero. Bits 4–0 Number of Ports (starting at 0) 00000 One port 00001 Two sequential ports 00010 Three sequential ports ... ... 11110 Thirty-one sequential ports 11111 Thirty-two sequential ports Byte 1 LSB of I/O Port Address Byte 2 MSB of I/O Port Address

Offset	Description
0 - 60 bytes for each function. 0 - 20 entries for each function.	I/O Port Initialization Data Section. Entries vary in length. Byte 0 Initialization Type Bit 7 0 Last entry 1 More entries follow Bits 6–3 Reserved, should be zeros. Bit 2 0 Write value to port 1 Use both mask and value Bits 1–0 Data Access Size *If Byte 0, bit 2 is 0, the following format is used:* 00 Byte 3 is the initialization value. 01 Byte 3 is the LSB of the initialization value. Byte 4 is the MSB of the initialization value. 10 Byte 3 is the LSB of the initialization value. Byte 4 is the second byte of the initialization value. Byte 5 is the third byte of theinitialization value. Byte 6 is the MSB of the initialization value. *If Byte 0, bit 2 is 1, the following format is used:* 00 Byte 3 is the initialization value. Byte 4 is mask value. 01 Byte 3 is the LSB of the initialization value. Byte 4 is the MSB of the initialization value. Byte 5 is the LSB of the mask value. Byte 6 is the MSB of the mask value. 10 Byte 3 is the LSB of the initialization value. Byte 4 is the second byte of the initialization value. Byte 5 is the third byte of the initialization value. Byte 6 is the MSB of the initialization value. Byte 7 is the LSB of the mask value. Byte 8 is the second byte of the mask value. Byte 9 is the third byte of the mask value. Byte 10 is the MSB of the mask value. Byte 1 LSB of Port Address Byte 2 MSB of Port Address
The following field is not included in the entries for each function. This field only occurs once at the very end of this table.	
2 bytes	Checksum of the CFG file that configured this table Byte 0 LSB of the EISA configuration file checksum. Byte 1 MSB of the EISA configuration file checksum.

Note: If bit 6 of the Function Information Section (22h) is set, the table is not in the table format described above, but uses free-form data. Entries through the Type and Subtype field are the same, but starting with the Memory Configuration field, the motherboard manufacturer's proprietary format is used.

Function D8h Subfunction 04h (84h) Read Slot Device Compressed ID

Input:	AH	=	D8h
	AL	=	04h For 16-bit addressing
		=	84h For 32-bit addressing
	CL	=	Slot Number (virtual and embedded devices included)
			00h Motherboard
			01h Slot 1
			02h Slot 2
			... ...
			... ...
			0Fh Slot 15
Output:	AH	=	00h No error
		=	80h Invalid slot number
		=	83h Slot is empty
		=	86h Invalid BIOS call
		=	87h EISA configuration is locked
	AL	=	Major revision number of ECU supported by BIOS (if AH = 88h).
	CF	=	No error
		=	1 Error
	DI	=	Least Significant Byte Byte 0 of Compressed ID
	DI	=	Most Significant Byte Byte 1 of Compressed ID
	SI	=	Least Significant Byte Byte 2 of Compressed ID
	SI	=	Most Significant Byte Byte 3 of Compressed ID

Description Function D8h Subfunction 04h (84h) returns the compressed ID from the device installed in the specified slot. The slot can be the motherboard, an adapter card, an embedded device, or a virtual device. DI and SI contain a four-byte compressed ID number of the device installed in the specified slot.

Function E8h Get Extended Memory Size

Input:	AH	=	E8h
	AL	=	01h (IBM) Return the total system memory size.
Output:	AH	=	00h No error
	CF	=	No error
		=	1 Error

Description INT 15h Function E8h subfunction 01h is used by OS/2.

Cont

Function E8h Subfunction AL = 20h Query System Address Map

Input:	AH	=	E8h
	AL	=	20h (Intel) Return a complete map of physical memory to the caller.
	EBX	=	Contains the value to retrieve the next unit of physical memory. This value is returned by a previous call to this function. If this is the first time INT 15h Function E820h is issued in a routine, EBX must be zero.
	ES:DI	=	Pointer to an Address Range Descriptor structure. The BIOS enters information in this structure.
	ECX	=	The length (in bytes) of the structure passed to the BIOS. The BIOS enters the number of bytes of information into the structure specified in ECX. The minimum structure size is hat must be supported by the calling program is 20 bytes.
	EDX		The signatures is MAP". The BIOS uses this signature to verify that the calling program is requesting that the BIOS send the system map information to the address pointed to by the contents of ES:DI.
Output:	EAX	=	MAP" signature
	CF	=	0 Successful 1 This is the last descriptor. The calling program should ignore any other information returned by the BIOS if CF =1.
	ES:DI	=	Pointer to an Address Range Descriptor structure (the same as the input value in these registers).
	ECX	=	The number of bytes returned by the BIOS in the address range descriptor. The minimum is 20 bytes.
	EBX	=	The continuation value to retrieve the next address descriptor. The calling program must use the unchanged continuation value as input in the next iteration of the INT 15h Function E820h call to retrieve the next Address Range Descriptor. If EBX = 0 or CF = 1 is returned by the BIOS, this is the last descriptor.

Description This function sends the system address map to the structure constructed by the calling program at the address pointed to by ES:DI. The BIOS does not report memory mapping for PCI devices option ROMS and ISA plug and play devices. Chipset-defined address holes are returned as reserved. Addresses reserved for motherboard memory-mapped I/O devices (such as APICs) are reported as reserved. System BIOS memory is reported as reserved. Standard PC address ranges (such as video memory) are not reported. Address ranges that describe motherboard memory and all contiguous ISA or PCI memory are reported.

Function E8h Subfunction 20h Query System Address Map, cont

Address Range Descriptor Structure

Offset	Field name	Description
0	BaseAddrLow	Low 32 bits of the base address. BaseAddrLow and BaseAddrHigh make up the 64-bit base address of this range. The base address is the physical address of the start of the specified range.
4	BaseAddrHigh	High 32 bits of the base address
8	LengthLow	Low 32 bits of the length of this range in bytes. LenghtLow and LengthHigh make up the 64-bit length of this range. The length is the physical contiguous length in bytes of the specified range.
8	LengthLow	Low 32 bits of the length of this range in bytes. LenghtLow and LengthHigh make up the 64-bit length of this range. The length is the physical contiguous length in bytes of the specified range.
12	LengthHigh	High 32 bits of the length of this range in bytes
16	Type	Address type of this range, as define below.

Type Field Address Ranges

Value	Mnemonic	Description
1	AddressRange Memory	This run is available RAM that can be used by the operating system.
2	AddressRangeReserved	This run of addresses is in use or is reserved and must not be used by the operating system.
3	AddressRange ACPI	ACPI Reclaim Memory. This run is available RAM that can be used by the operating system after it reads the ACPI tables.
4	AddressRange NVS	ACPI NVS Memory. This run of addresses is in use or is reserved and must not be used by the operating system. This range must be saved and restored during an NVS Sleep period.
other	Undefined	Reserved for future use. The operating system must treat this type the same as the AddressRangeReserved type.

INT 16h Keyboard Service

INT 16h controls the keyboard. Functions 00h through 02h are used with XT-compatible keyboards (83 and 84-key) only. Functions 10h through 12h are used with AT enhanced keyboards (101 and 102-key) only. Functions 03h and 05h can be used with either type of keyboard.

INT 16h Functions

Function	Description
00h	Read Character
01h	Return Keyboard Status
02h	Return Keyboard Flags
03h	Set Keyboard Typematic Rate Parameters
05h	Push Character and Scan Code to Buffer
10h	Enhanced Keyboard Read Character
11h	Enhanced Keyboard Write Character
12h	Enhanced Keyboard Return Keyboard Flags
E0h	AL = 00h Get BIOS/Flash ROM Interface Information AL = 01h Get Save and Restore Status Requirement AL = 02h Save Chipset Status and Prepare Chipset AL = 03h Restore Chipset Status AL = 04h Lower Programming Voltage Vpp AL = 05h Raise Programming Voltage Vpp AL = 06h Flash Write Protect AL = 07h Flash Write Enable AL = 08h Flash Select AL = 09h Flash Deselect AL = 0Ah Verify Allocated Memory AL = 0Bh Save Internal Cache Status AL = 0Ch Restore Internal Cache Status AL = 10h Get Flash Details AL = 11h Read ROM Bytes AL = FFh Generate CPU Reset
F0h	Set CPU Speed
F1h	Read CPU Speed
F4h	Cache Controller AL = 00h Read Cache Controller Status AL = 01h Enable Cache Controller AL = 02h Disable Cache Controller

Cont

Function 00h Read Character

Input:	AH	=	00h
Output:	AH	=	Scan code or character ID if special character.
	AL	=	ASCII code

Description INT 16h Function 00h reads a character from the keyboard and returns the scan and ASCII codes for that character.

Function 01h Return Keyboard Status

Input:	AH	=	01h
Output:	AH	=	Scan code of character ID if special character (only if ZF is 0).
	AL	=	ASCII code or character translation
	ZF	=	0 Character waiting
		=	1 No character waiting

Description INT 16h Function 01h determines if a character is waiting for input. If so, it returns the character and its scan code. Function 01h does not remove the character from the keyboard buffer. The character must be read using Function 00h to be removed from the buffer.

Function 02h Return Keyboard Flags

Input:	AH	=	02h
Output:	AL	=	Keyboard Flags Bit 7 1 Insert mode on Bit 6 1 Caps Lock key on Bit 5 1 Num Lock key on Bit 4 1 Scroll Lock key on Bit 3 1 Alt key pressed Bit 2 1 Ctrl key pressed Bit 1 1 Left Shift key pressed Bit 0 1 Right Shift key pressed

Description INT 16h Function 02h returns the Keyboard Flags Byte (40:17h in the BIOS Data Area). The Keyboard Flags Byte describes the state of certain keys.

Cont

Function 03h Set Typematic Rate Parameters

Input:

AH	=	03h	
AL	=	05h	
BH	=	Typematic delay	
		00h 250 ms	
		01h 500 ms	
		02h 750 ms	
		03h 1000 ms	
BL	=	Typematic rate	
		00h 30.0 cps	01h 26.7 cps
		02h 24.0 cps	03h 21.8 cps
		04h 20.0 cps	05h 18.5 cps
		06h 17.1 cps	07h 16.0 cps
		08h 15.0 cps	09h 13.3 cps
		0Ah 12.0 cps	0Bh 10.9 cps
		0Ch 10.0 cps	0Dh 9.2 cps
		0Eh 8.6 cps	0Fh 8.0 cps
		10h 7.5 cps	11h 6.7 cps
		12h 6.0 cps	13h 5.5 cps
		14h 5.0 cps	15h 4.6 cps
		16h 4.3 cps	17h 4.0 cps
		18h 3.7 cps	19h 3.3 cps
		1Ah 3.0 cps	1Bh 2.7 cps
		1Ch 2.5 cps	1Dh 2.3 cps
		1Eh 2.1 cps	1Fh 2.0 cps

Output: None

Description This function sets the keyboard typematic rate parameters. The typematic rate delay is the length of the delay between the first key character printed on the screen and first repeated character. The typematic rate is the number of characters to be repeated per second.

Cont

Function 05h Push Character and Scan Code to Buffer

Input:	AH	=	05h
	CH	=	Scan code to be pushed
	CL	=	Character to be pushed
Output:	AH	=	00h No error
		=	01h Keyboard buffer full
	CF	=	0 No error
		=	1 Keyboard buffer is full

Description INT 16h Function 05h places the specified character and scan code in the keyboard buffer.

Function 10h Enhanced Keyboard Read Character

Input:	AH	=	10h
Output:	AH	=	00h No error
		=	01h Keyboard buffer full
	AL	=	ASCII scan code

Description Function 10h reads a character from the keyboard buffer and returns its ASCII code and scan code.

Function 10h should be used with enhanced keyboards only.

Function 11h Enhanced Keyboard Return Status

Input:	AH	=	11h
Output:	AH	=	Scan Code or character ID if special character.
	AL	=	ASCII code of character
	ZF	=	0 Character waiting
		=	1 No character waiting

Description Function 11h determines if a character is waiting for input. If so, it returns the character and its scan code. Function 11h does not remove the character from the keyboard buffer. The character must be read via Function 10h to be removed from the buffer. Function 11h should be used only with enhanced keyboards.

Cont

Function 12h Return Enhanced Keyboard Flags

Input: AH = 12h

Output: AL = Keyboard flags
00h Right Shift key pressed
01h Left Shift key pressed
02h Ctrl key pressed
03h Alt key pressed
04h Scroll Lock is on
05h Num Lock is on
06h Caps Lock is on
07h Insert mode is on
08h Left Ctrl key is pressed
09h Left Alt key is pressed
0Ah Right Ctrl key is pressed
0Bh Right Alt key is pressed
0Ch Scroll Lock key is pressed
0Dh Num Lock key is pressed
0Eh Caps Lock key is pressed
0Fh SysReq key is pressed

Description INT 16h Function 12h returns the Keyboard Flags at 40:17h and 40:18h and the Extended Keyboard Flags Byte (40:97h). These flags describe the state of various keys on the keyboard. Function 12h should be used only when accessing enhanced keyboards.

Function E0h Flash EPROM Programming There are several types of Flash EPROM devices. The American Megatrends Flash programming utility (AMIFlash) must be generalized to be able to work with all types of Flash ROM hardware. INT 16h Function E0h provides 14 system BIOS subfunctions that facilitate the use of the AMIFlash Flash EPROM programming utility so AMIFlash can be used successfully with all types of Flash ROM hardware.

Cont

Function E0h Subfunction 00h Get AMIBIOS/Flash ROM Interface Information

Input:
AH = E0h
AL = 00h

Output:
AL = FAh Successful
BX = Version number in BCD format (should be 0350h).
CF = 0 Successful
= 1 Error
CX = Attribute
Bits 15-2 Reserved
Bit 1 Boot block programming
0 Protected flash block cannot be programmed.
1 Protected flash block can be programmed.
Bit 0 A16 inversion
0 A16 inversion is available.
1 A16 inversion is not available.

Description

This subfunction returns the version number of BIOS/Flash interface implementation in BCD format in BX. For example, version number 3.50 is returned in BX as 0350h. If CX bit 1 is 0, A16 inversion is available. If CX bit 1 is 1, the AMIFlash utility can program the 8 KB bootblock area.

This subfunction can be used to determine if the BIOS/Flash interface is in the system BIOS. After returning from the subfunction, AX should be checked for FAh even CF is 0 (successful operation).

All registers except the returned registers are saved. The contents of AX, BX, and CX are destroyed if this subfunction is successful (CF = 0). The contents of AL should be unchanged if unsuccessful.

Cont

Function E0h Subfunction 01h Get Chipset Save and Restore Status Requirement

Input:
AH = E0h
AL = 01h

Output:
AL = FAh Successful
BX = Number of bytes needed to save chipset environment.
CF = 0 Successful
= 1 Error
CX = Attribute
Bits 15-2 Reserved
Bit 1 Boot block programming
0 Protected flash block cannot be programmed.
1 Protected flash block can be programmed.
Bit 0 A16 inversion
0 A16 inversion is available.
1 A16 inversion is not available.

Description This subfunction returns the data area space needed to save the current chipset status.

Cont

Function E0h Subfunction 02h Save Chipset Status and Prepare Chipset

Input:	AH	=	E0h
	AL	=	02h
	ES:DI	=	Pointer to start of buffer where chipset status will be saved.
Output:	AL	=	FAh Successful
	BX	=	Number of bytes needed to save chipset environment.
	CF	=	0 Successful
		=	1 Error

Description This function saves the current chipset status in the specified data area and then prepares the chipset to make the Flash EPROM accessible. The current cache memory status, power management status, shadow status, and other status is saved.

This subfunction should be invoked before programming the Flash EPROM so the computer can be restored if a non-fatal error Flash utility error occurs. This function:

- saves chipset features, and
- disables Shadow RAM, cache memory, power management features, and other chipset features.

Disabling cache memory may be necessary to make the target ROM address space non-cacheable. If the target ROM address space is cacheable only when shadowing is enabled (for instance, only shadow RAM is cacheable, but ROM is not cacheable), disabling shadow RAM also makes the target ROM address space non-cacheable and cache memory does not have to be disabled. But if the ROM is cacheable, then cache memory must be disabled.

The contents of AL are destroyed if successful. The contents of AL should be unchanged if unsuccessful.

Cont

Function E0h Subfunction 03h Restore Chipset Status

Input:	AH	=	E0h
	AL	=	03h
	ES:DI	=	Pointer to start of buffer where chipset environment will be restored.
Output:	AL	=	FAh Successful
	BX	=	Number of bytes needed to save chipset environment.
	CF	=	0 Successful
		=	1 Error

Description This function restores the chipset status from the specified data area where the chipset status was saved by INT 16h Function E0h subfunction 02h Chipset Status and Prepare Chipset.

The contents of AL are destroyed if successful. The contents of AL should be unchanged if unsuccessful.

Function E0h Subfunction 04h Lower Programming Voltage Vpp

Input:	AH	=	E0h
	AL	=	04h
Output:	AL	=	FAh Successful
	CF	=	0 Successful
		=	1 Error

Description This subfunction lowers programming voltage Vpp to the normal level. This routine waits 50 ms after execution until the voltage level stabilizes.

The contents of AL are destroyed if this function is successful. The contents of AL should be unchanged if unsuccessful.

Lowering the Vpp programming voltage and write-protecting the Flash EPROM can be done in one operation in some Flash EPROMs. If the hardware supports this combination of functions, the calling program must only invoke INT 16h Function E0h Subfunction 04h and not Subfunction 06h.

Cont

Function E0h Subfunction 05h Raise Programming Voltage Vpp

Input:	AH	=	E0h
	AL	=	05h
Output:	AL	=	FAh Successful
	CF	=	0 Successful
		=	1 Error

Description This subfunction raises the programming voltage to 12.0 Volt. This subfunction waits 50 ms after execution until the voltage level stabilizes.

The contents of AL are destroyed if this function is successful. The contents of AL should be unchanged if unsuccessful.

Raising the Vpp programming voltage and write-protecting the Flash EPROM can be done in one operation in some Flash EPROMs. If the hardware supports this combination of functions, the calling program must only invoke INT 16h Function E0h Subfunction 05h and not Subfunction 06h.

Cont

Function E0h Subfunction 06h Flash Write Protect

Input:	AH	=	E0h
	AL	=	06h
Output:	AL	=	FAh Successful
	CF	=	0 Successful
		=	1 Error

Description This subfunction write-protects the Flash EPROM. The contents of AL are destroyed if successful. The contents of AL should be unchanged if unsuccessful.

Function E0h Subfunction 07h Flash Write Enable

Input:	AH	=	E0h
	AL	=	07h
Output:	AL	=	FAh Successful
	CF	=	0 Successful
		=	1 Error

Description This subfunction enables Flash EPROM programming. The contents of AL are destroyed if successful. The contents of AL should be unchanged if unsuccessful.

Function E0h Subfunction 08h Flash Select

Input:	AH	=	E0h
	AL	=	08h
Output:	AL	=	FAh Successful
	CF	=	0 Successful
		=	1 Error

Description This subfunction selects the Flash EPROM. In normal operation, a call to this subfunction is not necessary. This function should be issued if both a standard EPROM and Flash EPROM reside on the motherboard. If this subfunction call was unnecessary, it returns with CF set to 0. The contents of AX are destroyed if this subfunction is successful. The contents of AX should be unchanged if unsuccessful.

Cont

Function E0h Subfunction 09h Flash Deselect

Input:	AH	=	E0h
	AL	=	09h
Output:	AL	=	FAh Successful
	CF	=	0 Successful
		=	1 Error

Description This subfunction deselects the Flash EPROM. In normal operation, a call to this subfunction is not necessary. This function should be issued if both a standard EPROM and Flash EPROM reside on the motherboard. If this subfunction call was unnecessary, it returns with CF = 0. The contents of AL are destroyed if this subfunction is successful. The contents of AL should be unchanged if unsuccessful.

Function E0h Subfunction 0Ah Verify Allocated Memory

Input:	AH	=	E0h
	AL	=	0Ah
	BX	=	Offset part of specified memory address
	ES	=	Segment part of specified memory address
Output:	AL	=	FAh Successful
	CF	=	0 Successful
		=	1 Error

Description This subfunction indicates if the address specified in ES:BX can be used. Normally, you do not have to call this subfunction. If BX contains 0, this function returns with CF set to indicate an error condition.

If a certain memory region cannot be accessed (for example, 80000h – 9FFFFh is inaccessible when shadowing is disabled) invoke this subfunction to verify the memory that the Flash EPROM programming utility will use. If this subfunction call was unnecessary, it returns with CF = 0.

The contents of AL are destroyed if this subfunction is successful. The contents of AL should be unchanged if unsuccessful.

Cont

Function E0h Subfunction 0Bh Save Internal Cache Status

Input:	AH	=	E0h
	AL	=	0Bh
	ES:DI	=	Pointer to the beginning of a 4 KB buffer where the internal cache memory status is saved.
Output:	AL	=	FAh Successful
	CF	=	0 Successful
		=	1 Error

Description This subfunction saves the current status of internal cache memory to the buffer pointed to be ES:DI. This subfunction returns with CF set if the requisite cache memory hardware is not available or this subfunction was called from protected mode.

The calling program must make sure that the buffer pointed to by ES:DI is at least 16 bytes.

The contents of AL are destroyed if this subfunction is successful. The contents of AL should be unchanged if unsuccessful.

Function E0h Subfunction 0Ch Restore Internal Cache Status

Input:	AH	=	E0h
	AL	=	0Ch
	ES:DI	=	Pointer to the beginning of a buffer where the internal cache memory status is saved.
Output:	AL	=	FAh Successful
	CF	=	0 Successful
		=	1 Error

Description This subfunction restores the current status of the internal cache to the buffer pointed to by the contents of ES:DI. This subfunction returns with CF set if the requisite cache memory hardware is not available or this subfunction was called from protected mode.

The calling program must make sure that the buffer pointed to by ES:DI is at least 16 bytes. The contents of AX are destroyed if this subfunction is successful. The contents of AX should be unchanged if unsuccessful.

Cont

Function E0h Subfunction 10h Get Flash Details

Input:	AH	=	E0h
	AL	=	10h
Output:	AL	=	FAh Successful
	CF	=	0 Successful
		=	1 Error
	BX	=	Version Number in BCD format
	CX	=	Number of Flash parts supported by BIOS (one-based).
	DX	=	Index number (0-based) of flash parts detected by BIOS.

Description

This subfunction returns information about flash ROM support in this AMIBIOS. The contents of AL, BX, CX, and DX are destroyed.

Version 6.24 of the 7/15/95 core AMIBIOS should return 0350h in BX.

The one-based value returned in CX is the total number of flash parts supported by this AMIBIOS. Flash is not supported if BX contains 0000h.

The value returned in DX is the zero-based index number of the flash part detected by AMIBIOS.

Cont

Function E0h Subfunction 11h Read ROM Bytes

Input:	AH	=	E0h
	AL	=	11h
	CX	=	Number of bytes to be read.
	DS:SI	=	Pointer to the beginning ROM address to be read.
	DX	=	The zero-based index number of the flash part whose algorithm is used to read from ROM.
	ES:DI	=	Pointer to a buffer area for returned data.
Output:	AL	=	FAh Successful
	CF	=	0 Successful
		=	1 Error
	BX	=	Version Number in BCD format
	CX	=	Number of bytes read

Description

This subfunction returns the specified number of bytes from ROM. The contents of AL and CX are destroyed.

Place 0000h in CX to specify that 64 KB of ROM is to be read.

The number of flash parts supported by this AMIBIOS is returned by INT 16h Function E0h Subfunction 10h.

This function does not perform any error checking on the input parameters. You must make sure that all input parameters are valid. You must make sure that reads from ROM do not cross a 64 KB boundary.

Function E0h Subfunction FFh Generate CPU Reset

Input:	AH	=	E0h
	AL	=	FFh
Output:	AL		None

Description

This subfunction generates the CPU reset. A CPU reset is necessary to reboot the computer after the Flash EPROM has been programmed successfully.

This subfunction does not return control to the calling program. The contents of all registers are destroyed by this subfunction call, since the computer is rebooted when this subfunction is invoked.

Cont

Function F0h Set CPU Speed

Input:	AH	=	F0h	
	AL	=	00h	Equivalent to 6 MHz 286
		=	01h	Equivalent to 8 MHz 286
		=	02h	Full 16 MHz
		=	03h	Toggles between 8 MHz-equivalent and speed set by system board switch (Auto or High)
		=	08h	Full 16 MHz except 8 MHz-equivalent during floppy disk access.
		=	09h	Specify speed directly
	CX	=	If AL = 09h, CX = speed value, from 1 (slowest) to 50 (fastest.) 3 is equivalent to 8088.	
Output:	AL		None	

Description Function F0h sets the CPU speed to Low or High. This function returns no values and does not destroy the contents of any registers. This function is available only if the BIOS date is 06/06/92 or later.

Function F1h Read CPU Speed

Input:	AH	=	F1h	
Output:	AL	=	00h	Equivalent to 6 MHz 80286 (COMMON)
		=	01h	Equivalent to 8 MHz 80286 (FAST)
		=	02h	Full 16MHz (HIGH)
		=	03h	Toggles between 8HMz-equivalent and speed set by system board switch (Auto or High)
		=	08h	Full 16 MNz except 8 MHz-equivalent during floppy disk access
		=	09h	Specify speed directly
	CX	=	If AL = 09h, CX = speed value, from 1 (slowest) to 50 (fastest.) 3 is equivalent to 8088.	

Description Function F1h reads the current CPU speed. This function destroys the contents of AL, but no other registers. This function is available only if the BIOS date is 06/06/92 or later. The contents of AL are destroyed if successful.

Cont

Function F4h Subfunction 00h Read Cache Controller Status

Input:	AH	=	F1h
	AL	=	00h
Output:	AH	=	00h Cache controller cannot be enabled.
		=	E2h Successful
	AL	=	Cache controller status
		=	00h Cache controller not present
		=	01h Cache memory enabled
		=	02h Cache memory disabled
	CX	=	Cache memory size
			Bit 15 0 Cache size information is valid
			1 Cache size information is invalid
			Bits 14–0 Cache memory size in KB
	DH	=	Cache write technology
			Bit 7 0 Cache write information valid
			1 Cache write information invalid
			Bits 6–1 Reserved (Set to 0)
			Bit 0 0 Write-through caching algorithm
			1 Write-back caching algorithm
	DL	=	Cache type
			Bit 7 0 Cache type information is valid
			1 Cache type information invalid
			Bits 6–1 Reserved (Set to 0)
			Bit 0 Cache Type
			0 Direct-mapped
		=	1 Two-way set-associative

Description Function F4h Subfunction AL = 00h returns cache controller status information. If unsuccessful, no register values are changed. The values in AX, CX, and DX are destroyed if successful. This function is available only if the BIOS date is 06/06/92 or later.

Cont

Function F4h Subfunction 01h Enable Cache Controller

Input: AH = F4h
AL = 01h

Output: AH = 00h Cache controller cannot be enabled.
= E2h Cache controller can be enabled.

Description Function F4h Subfunction AL = 01h enables the cache controller. The register content is not changed if the cache controller cannot be enabled. The contents of AH are destroyed if successful. This function is available only if the BIOS date is 06/06/92 or later.

Function F4h Subfunction 02h Disable Cache Controller

Input: AH = F4h
AL = 02h

Output: AH = 00h Cache controller cannot be enabled.
= E2h Cache controller can be enabled.

Description Function F4h Subfunction AL = 02h disables the cache controller. The register content is not changed if the cache controller cannot be enabled. The contents of AH are destroyed if successful. This function is available only if the BIOS date is 06/06/92 or later.

Cont

INT 17h Parallel Port Service

INT 17h controls the parallel ports. AMIBIOS supports up to three parallel ports, initialized to the following beginning I/O port addresses: 03BCh, 0378h, and 0278h, if present. The default values for the parallel ports in AMIBIOS can be modified via AMIBCP.

INT 17h Parallel Printer Functions The INT 17h parallel printer functions are:

Function	Description
00h	Write Character
01h	Initialize Parallel Port
02h	Return Parallel Port Status

Function 00h Write Character

Input:
AH = 00h
AL = Character
DX = Parallel Port Number. Index to parallel port lead address table at 40:08h.
00h LPT 1
01h LPT 2
02h LPT 3

Output:
AH = Parallel Port Status
Bit 7 Printer not busy if set to 1.
Bit 6 Printer acknowledge if set to 1.
Bit 5 Out of paper if set to 1.
Bit 4 Printer selected if set to 1.
Bit 3 I/O error if set to 1.
Bits 2–1 Reserved
Bit 0 Printer timed-out is set to 1.

Description Function 00h writes a character to the specified parallel port. The status is returned in AH.

Cont

Function 01h Initialize Parallel Port

Input: AH = 01h
DX = Parallel Port Number. Index to parallel port lead address table at 40:08h.
00h LPT 1
01h LPT 2
02h LPT 3

Output: AH = Parallel Port Status

Bit 7	Printer not busy if set to 1.
Bit 6	Printer acknowledge if set to 1.
Bit 5	Out of paper if set to 1.
Bit 4	Printer selected if set to 1.
Bit 3	I/O error if set to 1.
Bits 2–1	Reserved
Bit 0	Printer timed-out if set to 1.

Description This function initializes the specified parallel port. The Parallel Port Status is in AH.

Function 02h Read Parallel Port Status

Input: AH = 02h
DX = Parallel Port Number. Index to parallel port lead address table at 40:08h.
00h LPT 1
01h LPT 2
02h LPT 3

Output: AH = Parallel Port Status

Bit 7	Printer not busy if set to 1.
Bit 6	Printer acknowledge if set to 1.
Bit 5	Out of paper if set to 1.
Bit 4	Printer selected if set to 1.
Bit 3	I/O error if set to 1.
Bits 2–1	Reserved
Bit 0	Printer timed-out if set to 1.

Description This function returns the specified parallel port status in AH.

Cont

Function 02h Read EPP Status

Input:	AH	=	02h
	BX	=	5050h (P")
	CH	=	45h (")
	DX	=	Printer port number 00h LPT1 01h LPT2 02h LPT3
Output:	AH	=	EPP Status 00h EPP is installed. 03h Installed but the specified port is not supported.
	CX:AL	=	Installed BIOS type 5050:45h PE". EPP v3.0+ BIOS is installed 4550:50h PP". EPP V1.0 BIOS is installed.
	DX:BX	=	FAR Entry pointer to Advanced BIOS (EPP v7)
	DX	=	EPP I/O base address
	ES:BX	=	FAR entry pointer to EPP BIOS (EPP v1.0 and 3.0)

Description This function returns the EPP status for the specified parallel port. See the EPP Specification documents for additional information.

INT 18h ROM Basic

Input: None
Output: None

Description On the original IBM PC, INT 18h transferred control to ROM BASIC. ROM BASIC is not supported by IBM anymore. If INT 18h is invoked, the BIOS halts the computer and displays:

```
NO BOOT DEVICE AVAILABLE
```

The only way to regain control is to reboot.

Other Uses of INT 18h Some network cards contain boot ROMs to permit a computer attached to a network to be booted without using a hard disk or floppy disk. These ROMs trap INT 18h to access the computer. For this reason, INT 18h has been included in AMIBIOS.

INT 19h System Boot Control

Input: None

Output: None

Description INT 19h transfers control to the operating system.

The system BIOS reads the boot sector (sector 1, track 0) from the primary boot device (floppy drive A:, a CD-ROM drive, or hard disk C:) and writes the data to 0000:7C00h. The BIOS gives control to the data at that address, which in turn loads (or boots) the operating system.

If the BIOS does not find a boot sector on the primary boot device, it looks for a boot sector on the secondary boot device. The primary and secondary boot devices are floppy drive A:, then hard disk drive C:.

If no boot sector is found on either drive A: or C:, INT 18h is invoked. See the INT 18h description.

System Boot In older AMIBIOS, the *Boot Up Sequence* options in AMIBIOS Advanced Setup permit you to set the boot sequence to *C:, then A: then CDROM* or *A:, then C:, then CDROM.* An option to boot only from the C: drive has been added to some AMIBIOS to prevent the inadvertent entry of viruses to the system via the A: drive.

The **1st Boot Device, 2nd Boot Device, 3rd Boot Device,** and **4th Boot Device** options in newer AMIBIOS Setup utilities allow much greater latitude. You can boot from floppy drives, floptical drives, CD-ROM drives, IDE devices, SCSI drives, or Network drives. See the description of the AMIBIOS Setup for your AMIBIOS for detailed information.

Using AMIBCP to Change Boot Sequence The OEM can also use AMIBCP to change the system boot sequence, setting drive C: as the primary boot device, and drive A: as the secondary boot device.

Cont

If...	and...	then...
The Advanced Setup option *System Boot Up Sequence* is set to A:, C:,	a bootable floppy disk is in drive A:,	INT 19h reads the boot sector on the floppy disk and places its contents at 7C00h.
The Advanced Setup option *System Boot Up Sequence* is set to A:, C:,	Drive A: has no bootable disk: or the floppy disk in drive A: is not bootable,	INT 19h invokes INT 18h. INT 18h displays: NO BOOT DEVICE AVAILABLE
The Advanced Setup option *System Boot Up Sequence* is set to C:, A:,	the boot sector is found on drive C:	INT 19h reads the boot sector on the floppy disk and places its contents at 7C00h.
The Advanced Setup option *System Boot Up Sequence* is set to C:, A:,	Hard Disk Drive C: has no boot sector (most likely the drive type is not properly configured)	INT 19h invokes INT 18h. INT 18h displays: NO BOOT DEVICE AVAILABLE

INT 1Ah Real Time Clock Service

INT 1Ah Features The INT 1Ah features include:

- set or read the system Real Time Clock,
- perform PCMCIA Socket Service,
- perform PCMCIA Card Services,
- perform PCI services, and
- perform Plug and Play and DMI (Desktop Management Interface) services.

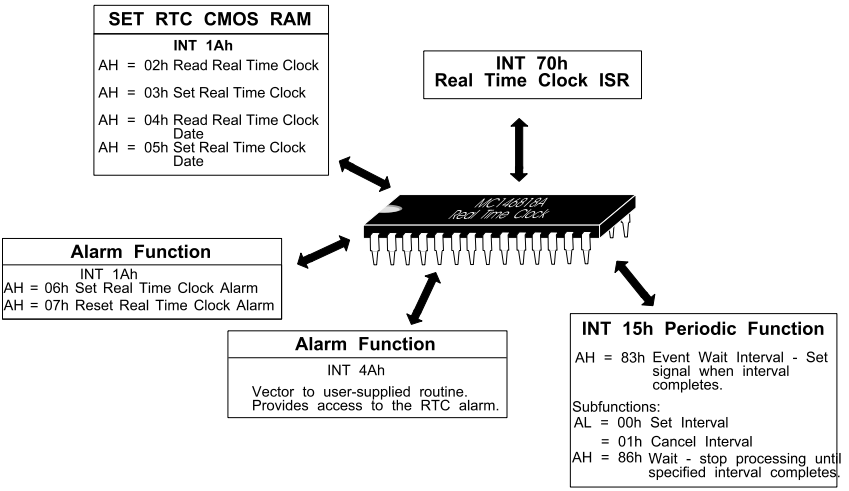
INT 1Ah Functions

Function	Service	Title
00h	Real Time Clock	Return Clock Tick Count
01h	Real Time Clock	Set Clock Tick Count
02h	Real Time Clock	Return Current Time
03h	Real Time Clock	Set Current Time
04h	Real Time Clock	Return Current Date
05h	Real Time Clock	Return Current Date
06h	Real Time Clock	Set Alarm
07h	Real Time Clock	Reset Alarm
50h	DMI	Get Number of DMI Structures
51h	DMI	Get DMI Structure
53h	DMI	Get DMI Event Information
55h	DMI	Get General Purpose NVRAM Data
56h	DMI	Read General Purpose NVRAM Data
57h	DMI	Write General Purpose NVRAM Data
80h	Socket Services	Set Adapter Count
83h	Socket Services	Get SS Information
84h	Socket Services	Inquire Adapter
85h	Socket Services	Get Adapter
86h	Socket Services	Set Adapter
87h	Socket Services	Inquire Window
88h	Socket Services	Get Window
89h	Socket Services	Set Window
8Ah	Socket Services	Get Page
8Bh	Socket Services	Set Page
8Ch	Socket Services	Inquire Socket
8Dh	Socket Services	Get Socket
8Eh	Socket Services	Set Socket
8Fh	Socket Services	Get Status
90h	Socket Services	Reset Card
95h	Socket Services	Inquire EDC (Error Detection Code)
96h	Socket Services	Get EDC
97h	Socket Services	Set EDC
98h	Socket Services	Start EDC
99h	Socket Services	Pause EDC

Function	Service	Title
9Ah	Socket Services	Resume EDC
9Bh	Socket Services	Stop EDC
9Ch	Socket Services	Read EDC
9Dh	Socket Services	Get Vendor Info
9Eh	Socket Services	Acknowledge Interrupt
9Fh	Socket Services	Get and Set Prior Handler
A0h	Socket Services	Get SS Address
A1h	Socket Services	Get and Set Access Offsets
AEh	Socket Services	Vendor-Specific
	Card Services	Card Services Functions
B1h	PCI	AL = 01h/81h PCI BIOS Present
B1h	PCI	AL = 02h/82h Find PCI Device
B1h	PCI	AL = 03h/83h Find PCI Class Code
B1h	PCI	AL = 06h/86h Generate Special Cycle
B1h	PCI	AL = 08h/88h Read Configuration Byte
B1h	PCI	AL = 09h/89h Read Configuration Word
B1h	PCI	AL = 0Ah/8Ah Read Configuration DWord
B1h	PCI	AL = 0Bh/8Bh Write Configuration Byte
B1h	PCI	AL = 0Ch/8Ch Write Configuration Word
B1h	PCI	AL = 0Dh/8Dh Write Configuration DWord
N/A	Plug and Play	00 GetDeviceNode
N/A	Plug and Play	01 GetSystemDeviceNode
N/A	Plug and Play	02 SetSystemDeviceNode
N/A	Plug and Play	40 GetISAConfigurationStructure
N/A	Plug and Play	03 GetEvent
N/A	Plug and Play	04 SendMessage
N/A	Plug and Play	05 GetDockingStationIdentifier
N/A	Plug and Play	07 SelectPrimaryBootDevices
N/A	Plug and Play	08 GetPrimaryBootDevices

Cont

Real Time Clock Functions The Real Time Clock ISR is INT 70h. See the INT 08h description for a discussion of timers. The following graphic illustrates how the real time clock is used with the BIOS.



INT 1Ah Socket Services Socket Services is an extension to system BIOS software interrupt 1Ah Real Time Clock Service. All Socket Services are function calls to INT 1Ah.

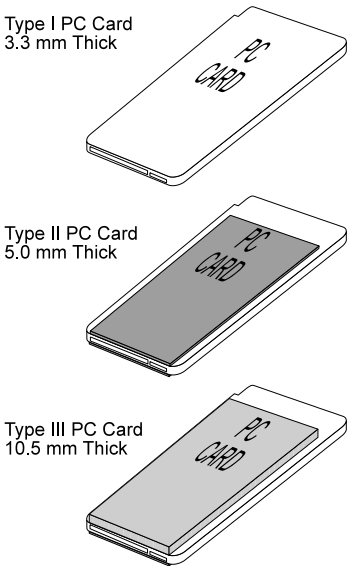
Socket Services provides the software interface to the hardware controlling PCMCIA-compatible cards (memory and I/O) in sockets. Socket Services provides the lowest level access to PCMCIA cards but does not interpret the content of the cards.

PCMCIA The PC Card specification has been defined by PCMCIA. The Personal Computer Memory Card International Association (PCMCIA) consists of over 300 international manufacturers of computer hardware, software, semiconductors, connectors, peripherals and system BIOS manufacturers (including American Megatrends, Inc.). PCMCIA develops methods or systems of data interchange suitable for the needs of portable computing. PCMCIA has developed a standard for PC Cards.

Cont

PC Card Size A PC Card is a small form factor electronic device a little thicker than a credit card. PC Cards provide functions such as added memory for data interchange between computers. Additionally, these cards are used to expand the I/O capabilities of a computer by adding such functions as serial or parallel ports, SCSI ports, network ports, and Fax/modems.

PC Card Types The PCMCIA specifications describe the physical, electrical, and software requirements for three card types: Types I, II, and III. All types use the same 68-pin edge connector to connect to the computer, but differ in width. The differences in the PC Card types is shown below.



Type I Cards Type I PC Cards are used primarily for various types of memory upgrades such as RAM, FLASH, One Time Programmable (OTP), or electrically.

Type II Cards Type II PC Cards can be used for memory enhancements as described in Type I above or for I/O functions such as FAX/Modems, LAN connections, or other host communications.

Cont

Type III Cards Type III PC Cards are twice the thickness of Type II cards and can be used for memory enhancements and/or I/O functions requiring additional room on the card such as rotating media devices and radio communication devices.

Form Follows Function Since all three cards adhere to the same electrical interface, the type of card chosen by the card designer depends totally on the function being implemented. The functionality of the card depends on the components inside the card and the software residing inside the computer.

Where Can PC Cards be Used? PC Cards can be used in laptop computers, palmtop computers, pen computers, desktop computers, or any other type of computing device that adheres to the specifications. PC Cards make communication between portable computers and desktop computers or peripherals easy and affordable.

Advantages Unlike ISA adapter cards, which require actual physical configuration jumpers and switches, PC Cards are configured through software. PCMCIA hardware and software provide an easy-to-install, self-configuring Plug and Play solution for portable and desktop computers. Once PCMCIA software has been installed and initialized, PC Cards can be inserted and removed with no concern for system performance.

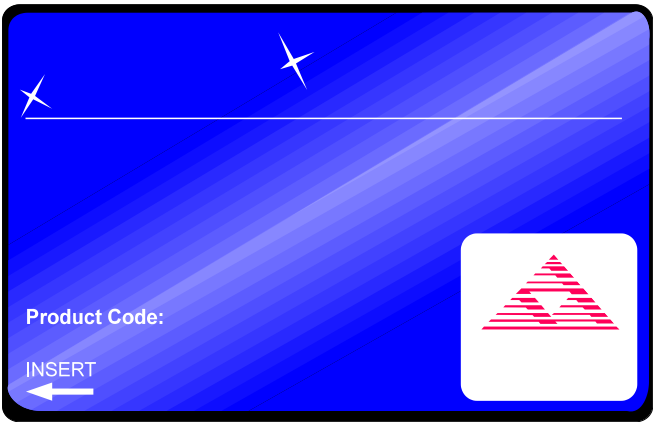
PCMCIA Software The primary architectural software building blocks required by PCMCIA PC Cards are Socket Services and Card Services.

Socket Services Function Socket Services are BIOS-resident and provide a means of access to the functions of the sockets themselves. Socket Services are used by the computer to identify how many sockets are in the computer and whether a card has been removed (hot extraction) or inserted into (hot insertion) the computer while it is powered on. Socket Service functions are called via INT 1Ah.

Cont

Card Services Card Services is a software/system management layer that allocates and manages computer resources to PC Cards. These activities can occur only after Socket Services has determined that there is a PC Card in one of the system sockets. Card Services also releases system resources for use by other system software if Socket Services determines that a specific PC Card has been removed from one of the system sockets. Card Services is the system software level interface for the operating systems for PC Card and Socket Services.

PCMCIA Hardware Standards The following illustration is approximately the actual size of a PCMCIA PC Card.



Function 00h Return Clock Tick Count

Input:	AH	=	00h
Output:	AL	=	00h Midnight has not passed since last call.
	CX:DX	=	Clock Tick Count (CX is the MSB)

Description Function 00h returns the value of the timer tick counter from 40:6Ch through 40:6Fh. The value is the number of ticks counted since midnight. Approximately 18.2 timer ticks occur every second.

The contents of 40:70h Timer Overflow are returned in AL. This value is zero if the timer has not overflowed past 24 hours since the last call.

Cont

Function 01h Set Clock Tick Count

Input:	AH	=	01h
	CX:DX	=	Clock Tick Count (CX is the MSB)
Output:			None

Description Function 01h sets the clock tick counter in 40:6Ch – 6Fh to the value specified in CX and DX. Approximately 18.2 ticks occur a second. The Timer Overflow flag at 40:70h is reset to 0 by this function.

Function 02h Return Current Time

Input:	AH	=	02h
Output:	CF	=	0 Successful
		=	1 Clock has stopped running
	CH	=	Number of Hours in Binary Coded Decimal (BCD)
	CL	=	Number of Minutes (in BCD)
	DH	=	Number of Seconds (in BCD)
	DL	=	00h Standard time
		=	01h Daylight savings time

Description Function 02h reads the current time from Real Time Clock CMOS RAM.

Function 03h Set Current Time

Input:	AH	=	03h
	CH	=	Number of Hours in Binary Coded Decimal (BCD)
	CL	=	Number of Minutes (in BCD)
	DH	=	Number of Seconds (in BCD)
	DL	=	00h Standard time
		=	01h Daylight savings time
Output:	AL	=	Value written to CMOS RAM Register B

Description Function 03h writes a specified time to Real Time Clock CMOS RAM.

Cont

Function 04h Return Current Date

Input:	AH	=	04h
Output:	CF	=	0 Successful
		=	1 Clock has stopped running
	CH	=	Number of Hours in Binary Coded Decimal (BCD)
	CL	=	Number of Minutes (in BCD)
	DH	=	Number of Seconds (in BCD)
	DL	=	00h Standard time
		=	01h Daylight savings time

Description Function 04h reads the current date from Real Time Clock CMOS RAM.

Function 05h Set Current Date

Input:	AH	=	05h
	CH	=	Century (in BCD)
	CL	=	Year (in BCD)
	DH	=	Month (in BCD)
	DL	=	Day (in BCD)
Output:	AL	=	Value written to Register B of CMOS RAM

Description This function writes the specified date to RTC CMOS RAM.

Cont

Function 06h Set Alarm

Input:	AH	=	06h
	CH	=	Hours (in BCD)
	CL	=	Minutes (in BCD)
	DH	=	Seconds (in BCD)
Output:	CF	=	0 No error
		=	1 The alarm is already set.

Description Function 06h sets an alarm for the specified time in RTC CMOS RAM and enables the clock interrupt request line (IRQ8). Trap the INT 4Ah vector (0:128h) and replace it with the address of your own alarm service routine.

Function 07h Reset Alarm

Input:	AH	=	07h
Output:	AL	=	Value written to Register B in CMOS RAM

Description This function resets all alarms in Real Time Clock CMOS RAM. This function does not disable the clock interrupt request line (IRQ8).

Socket Services Function Summary

Function	Name
80h	Get Adapter Count
83h	Get SS Information
84h	Inquire Adapter
85h	Get Adapter
86h	Set Adapter
87h	Inquire Window
88h	Get Window
89h	Set Window
8Ah	Get Page
8Bh	Set Page
8Ch	Inquire Socket
8Dh	Get Socket
8Eh	Set Socket
8Fh	Get Status
90h	Reset Card
95h	Inquire EDC (Error Detection Code)
96h	Get EDC
97h	Set EDC
98h	Start EDC
99h	Pause EDC
9Ah	Resume EDC
9Bh	Stop EDC
9Ch	Read EDC
9Dh	Get Vendor Information
9Eh	Acknowledge Interrupt
9Fh	Get and Set Prior Handler
A0h	Get SS Address
A1h	Get and Set Access Offsets
AEh	Vendor-Specific

Cont

Socket Services Calling Conventions Socket Services functions are invoked through software interrupt 1Ah. The general convention for invoking the socket services functions is:

Input:	AH	=	Function number
	AL	=	Adapter number
	BH	=	Window number
	BL	=	Socket number or page number
			Other input parameters may be added, depending on the specific function.
Output:	AH	=	Error code
	CF	=	0 Successful
		=	1 Error code

Function 80h Get Adapter Count

Input:	AH	=	80h
Output:	AL	=	Number of adapters (one-based)
	CF	=	0 Successful. Socket Services handler present.
			1 Error. Socket Services handler not present.
	CX	=	the string *SS*

Description This function returns the number of adapters supported by Socket Services and can be used to determine the presence of the Socket Services handler.

Even if the Socket Services handler is present, there may not be any adapter installed. In this case, this function should return with CF set, *SS* in CX, and 00h in AL. The caller of this function must handle this situation properly.

Cont

Function 83h Get SS Information

Input:	AH	=	83h
	AL	=	Adapter number (zero-based)
Output:	AH	=	Error code
		=	00h Successful
		=	01h Bad adapter
	AL	=	PCMCIA Socket Services Version Number
			00h Ensures compatibility with Release 1.01.
	BX	=	Socket Services Interface Specification
			Compliance Level (0201h for PCMCIA V2.01)
	CF	=	0 Successful
			1 Error.
	CH	=	Number of adapters supported by this handler.
	CL	=	First adapter supported by this handler.

Description

This function returns the version of both Implementer and PCMCIA Socket Services compliance levels. Version numbers are returned as binary coded decimals (BCD) values.

If more than one type of adapter is present, there may be more than one Socket Services handlers present. This function determines the support level of Socket Services for the specified adapter.

Cont

Function 84h Inquire Adapter

Input:	AH	=	84h
	AL	=	Adapter number (zero-based)
	ES:EDI	=	Pointer to a buffer supplied by the calling program that will be filled with information about the adapter by Socket Services.
Output:	AH	=	Error code
		=	00h Successful
		=	01h Bad adapter
	BH	=	Number of windows (one-based)
	BL	=	Number of sockets (one-based)
	CF	=	0 Successful
			1 Error.
	CX	=	Number of EDCs (Error Detection Code) (can be 0 – the total number of sockets)
	ES:EDI	=	Pointer to buffer containing adapter characteristics and power management tables.

Description This function returns information about the specified adapter.

The buffer pointed to by the contents of ES:EDI is supplied by the calling program and must have the following format:

```
typedef struct tagAISTRUCT {
    WORD wBufferLength;
    WORD wDataLength;
    ACHARTBL CharTable
    WORD wNumPwrEntries = NUM_ENTRIES;
    PWRENTRY PwrEntry[NUM_ENTRIES];
} AISTRUCT;
```

The CharTbl structure is defined below. wBufferLength must be set by the calling program to the size of AISTRUCT minus four bytes. wDataLength is set by Socket Services to the size of the information block returned. If the wDataLength value is greater than the wBufferLength value, the information is truncated.

Cont

Function 84h Inquire Adapter, Cont

PWRENTRY PWRENTRY is a two-member structure. The first member is a binary value representing a DC voltage level in tenths of a volt with a maximum of 25.5 VDC. The second member specifies the power signals that may be set to the specified voltage level (either Vcc, Vpp1, or Vpp2). All sockets on an adapter should use the same power levels. Make one PWRENTRY for each supported voltage. PWRENTRY only indicates that it is possible to set power pins to a certain power level. It is up to the calling program to determine if the specified combination of power levels is valid for the PC Card in the socket. The PWRENTRY structure is shown below:

```
typedef struct tagPWRENTRY {
    BYTE PowerLevel;
    BYTE ValidSignals;
} PWRENTRY
```

where:

PowerLevel	the DC voltage level in tenths of a volt. Power levels from 0 (N/C) through 25.5 VDC are valid.
ValidSignals	flags that indicate if voltage is valid for specific signals. A combination of the following can be used:

Vcc Voltage level valid for the Vcc signal
Vpp1 Voltage level valid for the Vpp1 signal
Vpp2 Voltage level valid for the Vpp2 signal

Sample AISTRUCT

```
AISTRUCT AdapterInfo = {
    24,             //Size of calling program-supplied buffer is 24 //bytes
    24,             //Size of data returned is 24 bytes
    {0,             //Indicators, power, and data bus width are controlled
                    //at the socket
    0xDEB8          //Status changes may be routed to IRQ levels
                    //3, 4, 5, 7, 9, 10, 11, 12, 14, and 15
                    //as an active high signal
    0},             //Status changes are not available on
                    //any level as an active low signal
    3,              //Number of PWRENTRY elements
    ((VCC | VPP1 | VPP2) << 8) | 0
                    //Vcc, Vpp1, and Vpp2 - No Connect
    ((VCC | VPP1 | VPP2) << 8) | 50
                    //Vcc, Vpp1, and Vpp2 - 5.0 VDC
    ((VPP | VPP2 | << 8) | 120
                    //Vpp1 and Vpp2 - 12.0 VDC
```

Cont

Function 84h Inquire Adapter, Cont

ACHATBL Structure The following code describes the ACHATBL structure. The parameters are described in the table that follows.

```
typedef struct tagACHATBL {    //Same format as Socket
                               //characteristics except
   WORD AdpCaps;               //CHARTBL has different values
   DWORD ActiveHigh;
   DWORD ActiveLow;
} ACHATBL;
```

Indicators	Description
AdpCaps	AdpCaps (Adapter capabilities) is structured as follows: Indicators 0 There are individual indicators for each socket. 1 Indicators for write protect, card lock, battery status, busy status, and XIP status are shared by all sockets on the adapter. Power Level 0 Power levels can be individually set for each socket. 1 The adapter requires all sockets to be set to the same power level controls. Data bus width 0 Data bus width set individually for each window. 1 All windows on the adapter must use the same data bus width.
ActiveHigh	A doubleword bitmap of the status change interrupt levels that can be routed active high.
ActiveLow	A doubleword bitmap of the status change interrupt levels that can be routed active low.

Cont

Function 85h Get Adapter This function returns the current configuration of the specified adapter.

Input: AH = 85h

Output: AH = Error code
= 00h Successful
= 01h Bad adapter

CF = 0 Successful
1 Error.

DH = Adapter attributes
Bits 7-2 Reserved
Bit 1 0 Preserve state information in power down
1 True
Bit 0 0 Reduce power consumption
1 True

DI = Status change interrupt routing
Bit 7 IRQ enabled
1 Status change is enabled.
Bit 6 IRQ high
1 Status change interrupt is active high.
Bits 4-0 IRQ level

Description

Bit 0 of DH (Reduce power consumption) indicates if the adapter hardware is attempting to conserve power. Before using the adapter, full power must be restored via INT 1Ah AH = 86h Set Adapter.

If Bit 1 of DH (Preserve State Information) is set to 1, all adapter and socket status are retained in reduced-power mode. If this bit is set to 0, the software that placed the adapter in reduced-power mode must save all adapter and socket status.

Not all adapters can reduce power consumption. Reduced power settings may not result in any power savings. The Inquire Adapter function (AH = 84h) indicates if it is possible to share the status change interrupt. This function returns the form of interrupt sharing (if any) currently being performed.

Cont

Function 86h Set Adapter This function sets the configuration of the specified adapter. The card status change interrupt is enabled through this function.

Input:

AH = 86h

AL = Adapter number (zero-based)

DH = Adapter attributes

- Bits 7-2 Reserved
- Bit 1 0 Status information in power down
  1 Preserve status information
- Bit 0 0 Power consumption
  1 Reduce

DI = Status change interrupt routing

- Bit 7 0 IRQ enabled
  1 Status change is enabled.
- Bit 6 0 IRQ high
  1 Status change interrupt is active high. If the adapter status change level is not programmable, this setting must match the actual hardware signal level.
- Bits 4-0 IRQ level

Output:

AH = Error code

= 00h Successful

= 01h Bad adapter

06h Bad IRQ

CF = 0 Successful

1 Error.

Bit 0 of DH (Reduce power consumption) indicates the adapter hardware is attempting to conserve power. Reduced power settings may not actually reduce power consumption because power management features are vendor-specific.

Before using the adapter, full power must be restored using this function. If Bit 1 of DH (Preserve state information) is set to 1, all adapter and socket status are retained in reduced-power mode.

If this bit is set to 0, the software that placed the adapter in reduced-power mode must save all adapter and socket status.

Cont

Function 87h Inquire Window This function returns information about the specified window on the specified adapter.

Input:	AH	=	87h
	AL	=	Adapter number (zero-based)
	BH	=	Window number (zero-based)
	ES:EDI	=	Pointer to a buffer provided by the calling program that holds window information.
Output:	AH	=	Error code
		=	00h Successful
		=	01h Bad adapter
			11h Bad window
	BL	=	Capabilities
			Bit 7 0 Use PC Card -WAIT signal
			1 Windows use the -WAIT signal from a PC Card to generate additional wait states.
			Bits 6-3 Reserved (set to 0)
			Bit 2 0 I/O space
			1 The window can be used to map I/O ports on a PC Card to the host system I/O space.
			Bit 1 0 Attribute memory
			1 The window can be used to map PC Card attribute memory to the host computer system memory.
			Bit 0 0 Common memory
			1 The window can be used to map PC Card common memory to host computer system memory.
	CF	=	0 Successful
			1 Error
	CX		Bitmap of assignable sockets
	ES:EDI		Pointer to either the memory window characteristics table or the I/O window characteristics table.

Cont

Function 87h Inquire Window, cont

Memory Window Characteristics Table

```
typedef struct tagMEMWINTBL {
   WORD  MemWndCaps;
   WORD  FirstByte;
   WORD  LastByte;
   WORD  MinSize;
   WORD  MaxSize;
   WORD  ReqGran;
   WORD  ReqBase;
   WORD  ReqOffset;
   BYTE  Slowest;
   BYTE  Fastest;
} MEMWINTBL;
```

MemWndCaps is a set of memory window characteristic flags:

Flag	Description
Base	If set, the base address of the window is programmable within the range specified by FirstByte and LastByte. If set to 0, the window base address is fixed in memory at the location specified in FirstByte and LastByte is undefined.
Size	If set, the window size is programmable within the range specified by MinSize and MaxSize.
Enable	If set, the windows can be disabled without reprogramming its characteristics. If 0, the calling program must preserve window state information before disabling the window.
8bit	If set, the window can be programmed for an 8-bit data bus width.
16bit	If set, the window can be programmed for a 16-bit data bus width.
Balign	If set, the window base address must be a multiple of the windows size. If 0, the base address can be any valid address.
Pow2	If set, a fixed-length window must be equal to a power of two of the ReqGran value. If 0, window size could be any value on a 4 KB boundary between 4 KB and 64 KB.
Calgn	If set, PC Card offsets must be in increments equal to the size of the window.
Pavail	If set, the windows can be divided into multiple pages via hardware. If 0, the window can only be addressed as a single page. If 0, the calling program must preserve page state information before disabling the page.
Pshare	If set, the window paging hardware is sharable with another window. A request to use the paging hardware may fail if another window is using it. This value is only valid if Pavail is set. *The calling program should check **Pshare** when using window paging. If set, the calling program must make sure a subsequent INT 1Ah AH = 89h Set Window request is successful before using the window. To determine if the page is available, assign it to a window by invoking INT 1Ah AH = 89h Set Window and make sure AH = 00h on return from Socket Services.*

Flag	Description
Penbl	If set, the page can be disabled without reprogramming its characteristics.
Wp	If set, the PC Card memory window mapped to the host computer system can be write-protected.
FirstByte	The first byte this window can use in the host memory system. If the window base address is not programmable, this is the same as the window base address.
LastByte	The last byte this window can use in the host memory system. The last byte of the window cannot exceed this value. This value is not used if the window base address is not programmable.
MinSize	The minimum window size. The window must meet all granularity and base requirements and must be between the *MinSize* and *MaxSize* values.
MaxSize	The maximum window size. The window must meet all granularity and base requirements and must be within the *MinSize* and *MaxSize* values. If *MaxSize* is 0, the window size is the largest value that may be represented by the SIZE data type plus one.
ReqGran	The units required for defining the windows size because of hardware constraints. If the window is a fixed size, this value is the same as Min Size and MaxSize.
ReqBase	If Balign is 0, this value specifies the boundary alignment for setting the window base address via INT 1Ah AH = 89h Set Window.
ReqOfst	If *Calign* is 0, this value specifies the boundary alignment for setting the window base address via INT 1Ah AH = 8Bh Set Page. This field is undefined if *Calign* is set.
Slowest	The slowest access speed supported by this window.
Fastest	The fastest access speed supported by this window. *Slowest* and *Fastest* are in the format specified by the PCMCIA Device Speed Code and Extended Device Speed Codes. Bit 7 of *Slowest* and *Fastest* is reserved and is always set to 0.

Function 87h Inquire Window, cont

I/O Window Characteristics Table

```
typedef struct tagIOWINTBL {
   WORD  IOWndCaps;
   WORD  FirstByte;
   WORD  LastByte;
   WORD  MinSize;
   WORD  MaxSize;
   WORD  ReqGran;
   BYTE  AddrLines;
   BYTE  EISASlot;
} IOWINTBL;
```

IOWndCaps is a set of I/O window characteristic flags, as follows:

Flag	Description
Base	If set, the base address of the window is programmable within the range specified by FirstByte and LastByte. If set to 0, the window base address is fixed in system I/O space at the location specified in FirstByte and LastByte is undefined.
Size	If set, the window size is programmable within the range specified by MinSize and MaxSize.
Wenable	If set, the windows can be disabled without reprogramming its characteristics. If 0, the calling program must preserve window state information before disabling the window.
8bit	If set, the window can be programmed for an 8-bit data bus width.
16bit	If set, the window can be programmed for a 16-bit data bus width.
Balign	If set, the window base address must be a multiple of the windows size. If 0, the base address can be any valid address.
Pow2	If set, a fixed-length window must be equal to a power of two of the Reqgran value. If 0, window size could be any value between the *MinSize* and *MaxSize* values.
Inpck	If set, the window supports the -INPACK signal from a PC Card. -INPACK allows windows to overlap in I/O space.
EISA	If set, the window supports EISA-type I/O mapping as would an EISA computer. EISASlot specifies the slot-specific address decodes for this window.
Cenable	If set, EISA-like common addresses can be ignored. If 0 and the window is programmed for EISA-like I/0 mapping, the PC Card receives a Card Enable signal when an access is made to an EISA common address. This value is only valid if *EISA* is set.
FirstByte	The first byte this window can use in the host I/O space. If the window base address is not programmable, this is the same as the window base address.
LastByte	The last byte this window can use in the host I/O space. The last byte of the window cannot exceed this value. This value is not used if the window base address is not programmable.

Flag	Description
MinSize	The minimum window size. The window must meet all granularity and base requirements and must be within the *MinSize* and *MaxSize* values.
MaxSize	The maximum window size. The window must meet all granularity and base requirements and must be within the *MinSize* and *MaxSize* values. If *MaxSize* is 0, the window size is the largest value that may be represented by the SIZE data type plus one.
ReqGran	The units required for defining the windows size because of hardware constraints. If the window is a fixed size, this value is the same as Min Size and MaxSize.
AddrLins	The number of address lines decoded by the window. Usually either 10 or 16. If a window only decodes 10 address lines, accesses to address above 1 KB will drive Card Accesses to a PC Card when the ten least significant address lines fall within the range defined by the base address and the window size.
EISASlot	The upper byte for window-specific EISA I/O decoding. This value specifies the upper four address lines used for EISA slot-specific addresses that drive Card Enables. This filed is not used if *EISA* is 0.

Cont

Function 88h Get Window This function returns the current configuration of the specified window on the specified adapter.

Input:
AH = 88h
AL = Adapter number (zero-based)
BH = Window number (zero-based)

Output:
AH = Error code
= 00h Successful
= 01h Bad adapter
11h Bad window
BL = Socket Number (zero-based)
CF = 0 Successful
1 Error
CX Size of window. In bytes for I/O windows. In 4 KB units for memory windows. If 0, the window is the maximum size that can be represented.
DH = Window state (bit-mapped). *Bits 3 and 4 vary if the function is reporting an I/O or a memory window.*
Bit 4 EISA common I/O. This bit is only valid for I/O windows that have bit 3 set.
0 Access to I/O ports in EISA common I/O areas ignored.
1 Access to I/O ports in EISA common I/O areas enabled.
Bit 3 (If I/O window)
0 ISA I/O mapping
1 EISA I/O mapping
Bit 3 Memory page (if memory window)
0 Single page window
1 Window divided into multiple 16 KB pages with PC Card offset addresses set individually via Function 8Bh Set Page.
Bit 2 16/8-bit data path
0 Window uses an 8-bit data bus width.
1 Window uses a 16-bit data bus width.
Bit 1 Window enabling
0 Window is disabled.
1 Window is enabled and can map a PC Card to the host system memory or I/O space.
Bit 0 I/O Mapping
0 Common or attribute memory is mapped to the host memory space.
1 PC Card registers mapped to the host I/O space.
DL = Access speed. Select only one. Not used for I/O windows. See the PCMCIA PC Card Standards 2.01 specification for the speed codes.
DI = Windows base address. In bytes if an I/O window. In 4 KB units if a memory window.

Cont

Function 89h Set Window This function sets the configuration of the specified window on the specified adapter. The area of the PC Card memory array mapped to the host memory is managed by INT 1Ah AH = 8Ah Get Page and INT A1h AH = 8Bh Set Page for memory-mapped windows.

Input:

AH = 89h

AL = Adapter number (zero-based)

BH = Window number (zero-based)

BL = Socket number (zero-based)

CX = Window size. In 4 KB units for memory windows and in bytes for I/O windows.

DH = Window state (bit-mapped). *Bits 3 and 4 vary if the function is reporting an I/O or a memory window.*

- Bit 4 EISA common I/O. This bit is only valid for I/O windows that have bit 3 set.
  - 0 Access to I/O ports in EISA common I/O areas ignored.
  - 1 Access to I/O ports in EISA common I/O areas enabled.
- Bit 3 I/O mapping type (If I/O window)
  - 0 ISA I/O mapping
  - 1 EISA I/O mapping
- Bit 3 Memory page (if memory window)
  - 0 Single page window
  - 1 Window divided into multiple 16 KB pages with PC Card offset addresses set individually via Function 8Bh Set Page.
- Bit 2 16/8-bit data path
  - 0 Window uses an 8-bit data bus width.
  - 1 Window uses a 16-bit data bus width.
- Bit 1 Window enabling
  - 0 Window is disabled.
  - 1 Window is enabled and can map a PC Card to the host system memory or I/O space.
- Bit 0 I/O Mapping
  - 0 Common or attribute memory is mapped to the host memory space.
  - 1 PC Card registers mapped to the host I/O space.

DL = Access speed. Select only one. Not used for I/O windows. See the PCMCIA PC Card Standards 2.01 specification for speed codes.

DI = Windows base address. In bytes if an I/O window. In 4 KB units if a memory window.

Output:

AH = Error code

= 00h Successful

= 01h Bad adapter

02h Bad attribute

03h Bad base

0Ah Bad size

0Bh Bad socket

0Ch Bad type

17h Bad speed

11h Bad window

CF = 0 Successful

1 Error

Cont

Function 8Ah Get Page This function returns the current configuration for the specified page in the specified window on the specified adapter.

Input:
- AH = 8Ah
- AL = Adapter number (zero-based)
- BH = Window number (zero-based)
- BL = Socket number (zero-based)

Output:
- AH =
  - 00h Successful
  - 01h Bad adapter
  - 08h Bad page number
  - 11h Bad window
- CF =
  - 0 Successful
  - 1 Error
- CX = Window size. In 4 KB units for memory windows and in bytes for I/O windows.
- DL = Page attributes
  - Bits 7-3 Reserved (set to 0)
  - Bit 2
    - 0 Write-protection
    - 1 Page is write-protected by page mapping hardware in the socket.
  - Bit 1 Page enable
    - 1 PC Card attribute memory is mapped to system memory or I/O space (if page is also enabled).
  - Bit 0 Type of mapping
    - 0 PC Card common memory is mapped to system memory (if page is also enabled).
    - 1 PC Card attribute memory is mapped to system memory (if page is also enabled).
- DI = Memory card offset (in 4 KB units)

Largest Page Number The maximum page number is the window size in bytes divided by 16 KB - 1. The associated socket number is implied by the prior INT 1Ah AH = 89h Set Window function call. Page attributes indicate if the page is currently enabled.

Bit 1 of DL returned by Function 8Ah Get Page and Bit 1 of DH as returned by Function 88h Get Window must be set before you can map PC Card memory into system memory.

Cont

Function 8Ah Get Page, cont

Description

For windows with Bit 3 of DH set to 0 as returned by Function 88h Get Window, Bit 1 of DL as returned by Function 8Ah Get Page is ignored. The windows is enabled and disabled by Bit 1 of DH as returned by Function 89h Set Window. Function 8Ah for windows with Bit 3 of DH set to 0 as returned by Function 88h Get Window supply the same value for Bit 1 of DH and Bit 1 of DL.

For windows with Bit 3 of DH set, Bit 1 of DH as returned by Function 88 Get Window globally enables or disables all pages in the window. After Bit 1 of DH has been set via Function 89h Set Windows, individual pages can be enabled and disabled via Function 8Bh Set Page and setting bit 1 of DL.

If the Wenable bit in the I/O window characteristics table is set as reported by Function 87h Inquire Window, Socket Services preserves the current state of DL bit 1 for every page in the window when Bit 1 of DH is changed by Function 89h Set Window. If Bit 1 of DH is 0 as returned by Function 87h Inquire Window, the calling program must:

- invoke Function 89h Set Window and set Bit 1 of DH, and then must
- invoke Function 8Bh Set Page to set Bit 1 of DL for each page in the window.

The memory card offset is the absolute memory card address (in 4 KB units) mapped to host system memory space for that page.

Cont

Function 8Bh Set Page This function sets the configuration for the specified page in the specified window on the specified adapter.

Input:

AH = 8Bh
AL = Adapter number (zero-based)
BH = Window number (zero-based)
BL = Socket number (zero-based)
DI = Memory card offset (4 KB units)
DL = Page attributes
- Bits 7-3 Reserved (set to 0)
- Bit 2 0 Write-protection
  1 Page is write-protected by page mapping hardware in the socket.
- Bit 1 Page enable
  1 PC Card attribute memory is mapped to system memory or I/O space (if page is also enabled).
- Bit 0 Type of mapping
  0 PC Card common memory is mapped to system memory (if page is also enabled).
  1 PC Card attribute memory is mapped to system memory (if page is also enabled).

Output:

AH = 00h Successful
01h Bad adapter
02h Bad attribute
07h Bad offset
08h Bad page number
11h Bad window
CF = 0 Successful
= 1 Error

Memory Windows Only Use this function for memory windows only. The maximum page number is equal to the window size in bytes divided by 16 KB - 1. The associated socket number is implied by the prior Set Window function call.

Cont

Function 8Bh Set Page, Cont

Description

If the hardware does not allow individual pages to be enabled (only the entire window can be disabled or enabled), this function returns an error on an attempt to disable a page.

The memory card offset is the absolute memory card address (in 4 KB units) mapped to host system memory space for that page. Bit 1 of DL as returned by Function 8Ah Get Page and Bit 1 of DH as returned by Function 88h Get Window must be set before you can map PC Card memory to system memory.

For windows with Bit 3 of DH set to 0 (as returned by Function 88h Get Window), Bit 1 of DL (returned by Function 8Ah Get Page) is ignored. The window is enabled by Bit 1 of DH (returned by Function 89h Set Window).

Issue Function 8Ah for windows with Bit 3 of DH set to 0 (returned by Function 88h Get Window) to supply the same value for Bit 1 of DH and DL.

For windows with Bit 3 of DH set, Bit 1 of DH (returned by Function 88 Get Window) globally enables all pages in the window. After Bit 1 of DH has been set via Function 89h Set Windows, individual pages can be enabled by issuing Function 8Bh Set Page and setting bit 1 of DL.

If the Wenable bit in the I/O window characteristics table is set as reported by Function 87h Inquire Window, Socket Services preserves the current state of DL bit 1 for every page in the window when Bit 1 of DH was changed by Function 89h Set Window. If Bit 1 of DH is 0 as returned by Function 87h Inquire Window, the calling program must:

- invoke Function 89h Set Window and set Bit 1 of DH, and then must
- invoke Function 8Bh Set Page to set Bit 1 of DL for each page in the window.

Cont

Function 8Ch Inquire Socket This function returns information about the specified socket on the specified adapter.

Input:

AH	=	8Ch
AL	=	Adapter number (zero-based)
BL	=	Socket number (zero-based)
ES:EDI	=	Pointer to buffer supplied by the calling program for information about the socket.

Output:

AH	=	00h Successful 01h Bad adapter 0Bh Bad socket
BL	=	Status change interrupt flags. Before an event can trigger a status change interrupt on a socket, the corresponding value in the Status Change Interrupt Mask parameter in INT 1Ah AH = 8Dh Set Socket must be set and status change interrupts must be enabled. Bit 7 Card Detect signal (set to 1 if enabled) Bit 6 PC Card RDY/BSY signal (set to 1 if enabled) Bit 5 PC Card BVD2 (battery weak) signal (set to 1 if enabled) Bit 4 PC Card BVD1 (dead battery) signal (set to 1 if enabled) Bit 3 Externally-generated signal to insert a PC Card in the socket (set to 1 if enabled) Bit 2 Externally-generated signal to eject PC Card (set enabled) Bit 1 Externally-generated signal from a mechanical or electric card lock (set to 1 if enabled) Bit 0 PC Card Write-Protect signal (set to 1 if enabled)
CF	=	0 Successful
	=	1 Error
DH	=	Status change events that the socket can report on. If an event is not reportable by INT 1Ah AH = 8Fh Get Status, it is set to 0. The bit settings are exactly the same as for BL above.
DL	=	Hardware indicators Bit 7 XIP status (set to 1 if enabled) Bit 6 Card busy status (set to 1 if enabled) Bit 5 Battery status (set to 1 if enabled) Bit 4 Card lock status (set to 1 if enabled) Bit 3 Externally-generated signal to insert a PC Card in the socket (set to 1 if enabled) Bit 2 Externally-generated signal to eject a PC Card from the socket (set to 1 if enabled) Bit 1 Externally-generated signal from a mechanical or electric card lock (set to 1 if enabled) Bit 0 PC Card Write-Protect signal (set to 1 if enabled)
ES:EDI	=	Pointer to buffer supplied by the calling program to hold the information about the socket. The required table structure is shown below.

Cont

Function 8Ch Inquire Socket, cont

Socket Information Table Structure

```
typedef SISTRUCT {
   WORD   WBufferLength    //Size of buffer provided by calling program
   WORD   wDataLength      //Size of data returned is 10 bytes
   SCHARTBL CharTable;
} SISSTRUCT
```

Socket Information Table Structure Example

```
   SISTRUCT SocketInfo = {
      10,           //Size of buffer provided by calling
                    //program is 10 bytes
      10,           //Size of data returned is 10 bytes
   IF_MEMORY\IF_IO  //Socket support memory-only and
                    //I/O and memory interfaces
      0xDEB8,       //PC Card IRQ signal can be routed to IRQs
                    // 3, 4, 5, 7, 9, 10, 11, 12, 14, and 15
                    //as an active high signal.
      0},           //PC Card IREQ routing not available on
                    //any level as an active low signal.
   };
```

Socket Characteristics Structure

```
   typedef struct tagSCHARTBL {   //same as adapter
      WORD SktCaps;               //except for this member
      DWORD ActiveHigh;
      DWORD ActiveLow;
   } SCHARTBL;
```

SktCaps SktCaps are flags that specify socket characteristics:

Flags	Description
IF_MEMRY	The socket supports memory-only interfaces as per Release 2.01.
IF_IO	The socket supports I/O port and memory interfaces as per Release 2.01.

ActvHgh A bitmap of the IRQ levels available for routing an inverted PC Card IREQ signal when an unmasked event occurs.

ActvLw A bitmap of the IRQ levels available for routing the normal PC Card IREQ signal when an unmasked event occurs. Normal PC Card IREQ signals can be shared in a host system.

Cont

Function 8Dh Get Socket This function returns the current configuration of the specified socket. Voltage levels Vcc, Vpp1, Vpp2 are indexes to the power management table.

Input: AH = 8Dh
AL = Adapter number (zero-based)
BL = Socket number (zero-based)

Output: AH = 00h Successful
= 01h Bad adapter
= 0Bh Bad socket

BH = Status change interrupt enable mask
Bit 7 Card detect change (1 is Enabled)
Bit 6 Ready change (1 is Enabled)
Bit 5 Battery warning change (1 is Enabled)
Bit 4 Battery dead change (1 is Enabled)
Bit 3 Insertion request (1 is Enabled)
Bit 2 Ejection request (1 is Enabled)
Bit 1 Card lock (1 is Enabled)
Bit 0 Write protect (1 is Enabled)

CF = 0 Successful
= 1 Error

CH = Bits 3-0 Vcc level

CL = Bits 7-4 Vpp1 level
Bits 3-0 Vpp2 level

DH = Bitmapped socket state
= Bit 7 Card detect change (1 is Enabled)
Bit 6 Ready change (1 is Enabled)
Bit 5 Battery warning change (1 is Enabled)
Bit 4 Battery dead change (1 is Enabled)
Bit 3 Insertion request (1 is Enabled)
Bit 2 Ejection request (1 is Enabled)
Bit 1 Card lock (1 is Enabled)
Bit 0 Write protect (1 is Enabled)

DL = Indicators
= Bit 7 XIP status (1 is Enabled)
Bit 6 Card busy status (1 is Enabled)
Bit 5 Battery status (1 is Enabled)
Bit 4 Card lock status (1 is Enabled)
Bit 3 Externally-generated signal to insert PC Card (1 is Enabled)
Bit 2 Externally-generated signal to eject PC Card (1 is Enabled)
Bit 1 Externally-generated signal from card lock (1 is Enabled)
Bit 0 PC Card Write-Protect signal (1 is Enabled)

DI = IRQ level steering (valid I/O cards only)
= Bit 9 I/O and memory interface (1 is Enabled)
Bit 8 Memory interface (1 is Enabled)
Bit 7 IRQ enabled (1 is Enabled)
Bit 6 IRQ high (1 is Enabled)
Bits 4-0 IRQ level
00h-0Fh IRQ 00h-0Fh
10h NMI
11h I/O check
12h Bus error
13h Vendor-unique

Cont

Function 8Eh Set Socket This function sets the current configuration of the specified socket. It waits until the requested Vpp power level becomes valid before it sets the socket parameters.

Input:

AH = 8Eh
AL = Adapter number (zero-based)
BL = Socket number (zero-based)
BH = Status change interrupt enable mask
- Bit 7 Card detect change (1 is Enabled)
- Bit 6 Ready change (1 is Enabled)
- Bit 5 Battery warning change (1 is Enabled)
- Bit 4 Battery dead change (1 is Enabled)
- Bit 3 Insertion request (1 is Enabled)
- Bit 2 Ejection request (1 is Enabled)
- Bit 1 Card lock (1 is Enabled)
- Bit 0 Write protect (1 is Enabled)

CH = Bits 3-0 Vcc level
CL = Bits 7-4 Vpp1 level
Bits 3-0 Vpp2 level
DH = Bitmapped socket attributes
= Bit 7 Card detect change (1 is Enabled)
- Bit 6 Ready change (1 is Enabled)
- Bit 5 Battery warning change (1 is Enabled)
- Bit 4 Battery dead change (1 is Enabled)
- Bit 3 Insertion request (1 is Enabled)
- Bit 2 Ejection request (1 is Enabled)
- Bit 1 Card lock (1 is Enabled)
- Bit 0 Write protect (1 is Enabled)

DL = Indicators
= Bit 7 XIP status (1 is Enabled)
- Bit 6 Card busy status (1 is Enabled)
- Bit 5 Battery status (1 is Enabled)
- Bit 4 Card lock status (1 is Enabled)
- Bit 3 Externally-generated signal to insert a PC Card (1 is Enabled)
- Bit 2 Externally-generated signal to eject a PC Card (1 is Enabled)
- Bit 1 Externally-generated signal from a card lock (1 is Enabled)
- Bit 0 PC Card Write-Protect signal (1 is Enabled)

DI = IRQ level steering (valid I/O cards only)
- Bits 15-10 Reserved
- Bit 9 I/O and memory interface (1 is Enabled)
- Bit 8 Memory interface (1 is Enabled)
- Bit 7 IRQ enabled (1 is Enabled)
- Bit 6 IRQ high (1 is Enabled)
- Bits 4-0 IRQ level
  - 00h-0Fh IRQ 00h-0Fh
  - 10h NMI
  - 11h I/O check
  - 12h Bus error
  - 13h Vendor-unique

Output:

AH = 00h Successful
= 01h Bad adapter
02h Bad attribute
= 0Bh Bad socket
CF = 0 Successful
= 1 Error

Cont

Function 8Fh Get Status This function returns the PC Card status. It must not be invoked during hardware interrupt processing. It should not be invoked by the calling program's status change hardware interrupt handler.

Input:	AH	=	8Fh
	AL	=	Adapter number (zero-based)
	BL	=	Socket number (zero-based)
Output:	AH	=	00h Successful
		=	01h Bad adapter
		=	0Bh Bad socket
	BH	=	Card Status
			Bit 7 Card changed (1 is Enabled)
			Bit 6 Card Busy status (1 is Enabled)
			Bit 5 Card insertion complete (1 is Enabled)
			Bit 4 Card ejection complete (1 is Enabled)
			Bit 3 Card insertion request pending (1 is Enabled)
			Bit 2 Card ejection request pending (1 is Enabled)
			Bit 1 Card lock (1 is Enabled)
			Bit 0 Write protect (1 is Enabled)
	CF	=	0 Successful
		=	1 Error
	DH	=	Socket state
			Bit 7 Card changed (1 is Enabled)
			Bit 6 Card Busy status (1 is Enabled)
			Bit 5 Card insertion complete (1 is Enabled)
			Bit 4 Card ejection complete (1 is Enabled)
			Bit 3 Card insertion request pending (1 is Enabled)
			Bit 2 Card ejection request pending (1 is Enabled)
			Bit 1 Card lock (1 is Enabled)
			Bit 0 Write protect (1 is Enabled)
	DL	=	Card attributes (bit-mapped)
			Bit 7 XIP status (1 is Enabled)
			Bit 6 Card busy status (1 is Enabled)
			Bit 5 Battery status (1 is Enabled)
			Bit 4 Card lock status (1 is Enabled)
			Bit 3 Externally-generated signal to insert a PC Card (1 is Enabled)
			Bit 2 Externally-generated signal to eject a PC Card (1 is Enabled)
			Bit 1 Externally-generated signal from a card lock (1 is Enabled)
			Bit 0 PC Card Write-Protect signal (1 is Enabled)
	DI	=	IRQ level steering (valid I/O cards only)
			Bits 15-10 Reserved
			Bit 9 I/O and memory interface (1 is Enabled)
			Bit 8 Memory interface (1 is Enabled)
			Bit 7 IRQ enabled (1 is Enabled)
			Bit 5 IRQ high (1 is Enabled)
			Bits 4-0 IRQ level
			00h-0Fh IRQ 00h-0Fh
			10h NMI
			11h I/O check
			12h Bus error
			13h Vendor-unique

Cont

Function 90h Reset Socket This function resets the specified socket on the specified adapter and returns the socket hardware to the power-on default state: Vcc, Vpp1, and Vpp2 are set to 5VDC, IRQ routing is disabled, memory-type mapping is set, and all windows, pages, and EDC generators are disabled. The calling program must make sure a PC Card is not accessed before ready after this function returns. This function sets the RESET pin on the card to the reset state and then resets the RESET pin to non-reset state, ensuring that the minimum reset pulse width is met. Make sure that the card is not accessed before it is ready after returning.

Input:	AH	=	90h
	AL	=	Adapter number (zero-based)
	BL	=	Socket number (zero-based)
Output:	AH	=	00h Successful
		=	01h Bad adapter
		=	0Bh Bad socket
		=	14h No PC Card in socket
	CF	=	0 Successful
		=	1 Error

Cont

Function 95h Inquire EDC This function returns the capabilities of the specified EDC (Error Detection Code) generator.

Socket Services supports two types of EDC generation: 8-bit checksums and 16-bit CRC SDLC.

EDC generation can be produced by read or write accesses. Code that uses many sequential reads and writes must use EDC generation carefully. Bidirectional EDC generation may not work with flash EPROM programming routines because these routines typically require many reads and writes.

EDC generation may not be available with memory-mapped implementations. EDC generators must be configured via INT 1Ah AH = 97h Set EDC.

Not every hardware implementation provides EDC code generation. The output of this function describes the EDC functions of the specified EDC generator. EDC generators can be shared between sockets. Card Services or higher-level software arbitrates the use of EDC generators.

Cont

Function 95h Inquire EDC, cont

Input:
AH = 95h
AL = Adapter number (zero-based)
BL = Socket number (zero-based)

Output:
AH = 00h Successful
= 01h Bad adapter
= 04h Bad EDC
CF = 0 Successful
= 1 Error
CX = Assignable sockets (Bit 0 is socket 0, bit 1 is socket 1, etc)
Bits 7-5 Reserved
Bit 4 Socket 4
Bit 3 Socket 3
Bit 2 Socket 2
Bit 1 Socket 1
Bit 0 Socket 0
DH = EDC capabilities (bit-mapped)
Bits 7-5 Reserved (set to 0)
Bit 4 EDC can be paused
1 EDC generation can be paused.
Bit 3 Memory-mapped support
1 EDC generation supported during window access.
Bit 2 Register-based support
1 EDC generation is supported through register-based access.
Bit 1 Bidirectional code generation
1 The EDC generator supports bidirectional code generation.
Bit 0 Unidirectional code generation
1 The EDC generator supports unidirectional code generation.
DL = Supported EDC types
Bits 7-2 Reserved (set to 0)
Bit 1 16-Bit CRC-SDLC
1 The EDC generator supports 8-bit checksum code generation.
Bit 0 8-Bit checksum
1 The EDC generator supports 8-bit checksum code generation.

Cont

Function 96h Get EDC This function returns the current configuration of the specified EDC generator. A generator is not assigned if the socket number returned is zero.

Input:
AH = 96h
AL = Adapter number (zero-based)
BL = Socket number (zero-based)

Output:
AH = 00h Successful
= 01h Bad adapter
= 04h Bad EDC
BL = Socket number of the physical socket that the EDC generator is assigned to (zero-based).
CF = 0 Successful
= 1 Error
DH = EDC attributes (Bit-mapped)
Bits 7-2 Reserved (set to 0)
Bit 1 If unidirectional only (Bit 0) is 1
0 EDC computing only on read accesses.
1 EDC computing only on write accesses.
Bit 0 Unidirectional only
0 EDC computing on both read and write accesses.
1 EDC computing in only one direction.
DL = EDC type (mutually exclusive bitmap)
Bits 7-2 Reserved (set to 0)
Bit 1 16-Bit CRC-SDLC EDC checksum generated by EDC.
Bit 0 8-Bit checksum generated by EDC.

Cont

Function 97h Set EDC This function sets the error detection and correction configuration of the specified EDC generator.

Input:
AH = 97h
AL = Adapter number (zero-based)
BH = EDC generator number (zero-based)
BL = Socket number (zero-based)
DH = EDC attributes (Bit-mapped)
Bits 7-2 Reserved (set to 0)
Bit 1 EDC computes on reads or writes
0 Reads
1 Writes
Bit 0 Unidirectional
1 EDC generator compute in only one direction.
DL = EDC type (mutually exclusive bitmap)
Bits 7-2 Reserved (set to 0)
Bit 1 16-Bit CRC-SDLC
1 16-bit EDC checksum generated.
Bit 0 8-Bit CRC-SDLC
1 8-bit EDC checksum generated

Output:
AH = 00h Successful
= 01h Bad adapter
02h Bad attribute
= 04h Bad EDC
0Bh Bad socket
CF = 0 Successful
= 1 Error

Function 98h Start EDC This function starts the specified previously configured EDC generator. This function load initialization values into the EDC generator.

Input:
AH = 98h
AL = Adapter number (zero-based)
BH = EDC generator number (zero-based)

Output:
AH = 00h Successful
= 01h Bad adapter
02h Bad attribute
= 04h Bad EDC
CF = 0 Successful
= 1 Error

Cont

Function 99h Pause EDC This function pauses EDC generation on the specified configured and computing EDC generator. This function is only supported if Bit 4 of DH is set when INT 1Ah AH= 95h Inquire EDC is invoked.

Input:	AH	=	99h
	AL	=	Adapter number (zero-based)
	BH	=	EDC generator number (zero-based)
Output:	AH	=	00h Successful
		=	01h Bad adapter
		=	04h Bad EDC
	CF	=	0 Successful
		=	1 Error

Function 9Ah Resume EDC This function resumes the EDC generation on the specified configured and paused EDC generator. This function can only be used if bit 4 of DH as returned by the INT 1Ah AH = 95h Inquire EDC function is set.

Input:	AH	=	9Ah
	AL	=	Adapter number (zero-based)
	BH	=	EDC generator number (zero-based)
Output:	AH	=	00h Successful
		=	01h Bad adapter
		=	04h Bad EDC
	CF	=	0 Successful
		=	1 Error

Function 9Bh Stop EDC This function stops the EDC generation on the specified configured and computing EDC generator.

Input:	AH	=	9Bh
	AL	=	Adapter number (zero-based)
	BH	=	EDC generator number (zero-based)
Output:	AH	=	00h Successful
		=	01h Bad adapter
		=	04h Bad EDC
	CF	=	0 Successful
		=	1 Error

Cont

Function 9Ch Read EDC This function reads the calculated EDC value computed by the specified EDC generator. The computed value may be incorrect if the EDC generator has been used incorrectly.

Input:	AH	=	9Ch
	AL	=	Adapter number (zero-based)
	BH	=	EDC generator number (zero-based)
Output:	AH	=	00h Successful
		=	01h Bad adapter
		=	04h Bad EDC
	CF	=	0 Successful
		=	1 Error
	DX	=	Computed checksum or CRC. This can be an 8-bit or 16-bit value depending on the value of Bits 0 and 1 in DL as returned by INT 1Ah AH = 95h Inquire EDC.

Function 9Dh Get Vendor Info This function returns information about the vendor implementing socket services for the specified adapter.

Input:	AH	=	9Dh
	AL	=	Adapter number (zero-based)
	BH	=	EDC generator number (zero-based)
	ES:EDI	=	Address of buffer where vendor information is stored.
Output:	AH	=	00h Successful
		=	01h Bad adapter
		=	15h Bad function
	CF	=	0 Successful
		=	1 Error
	ES:EDI	=	Address of buffer where vendor information is stored.
	DX	=	Vendor release number in BCD

Buffer Format The buffer pointed to by the value in ES:EDI must have the following format:

```
typedef struct tagVISTRUCT {
   WORD wBufferlength = (BUF_SIZE - 4);
   WORD wDataLength;    Set by Socket Services
   char szImplementor[BUF_SIZE - 4];
} VISTRUCT;
```

If the wData Length value is greater than the wBufferLength value, the information is truncated.

Cont

Function 9Eh Acknowledge Interrupt This function returns status change information for sockets on the specified adapter. Socket Services does not enable interrupts while this function is being performed.

The calling program should enable status change interrupts from adapter hardware via INT 1Ah AH = 86h Set Adapter. The calling program must install an interrupt handler on the appropriate vector.

Specific events can be masked or unmasked for each socket via INT 1Ah AH = 8Eh Set Socket.

When a status change occurs, the calling program's status change handler receives control and invokes INT 1Ah AH = 9Eh Acknowledge Interrupt. This function permits Socket Services to prepare the adapter hardware to generate another interrupt if another status change occurs.

Socket Services preserves status change information if it is not preserved by the adapter hardware.

If this function is called and no status change has occurred on the specified adapter, Socket Services returns with AH and CX = 00h.

Input:	AH	=	9Eh
	AL	=	Adapter number (zero-based)
Output:	AH	=	00h Successful
		=	01h Bad adapter
	CF	=	0 Successful
		=	1 Error
	CX	=	A bitmap that represents the sockets that have changed status.

Cont

Function 9Fh Get and Set Prior Handler This function replaces or acquires the entry point of a prior handler for the specified adapter. If this handler is first in the INT 1Ah chain, the values returned when this function is issued with BL = 0 should be the entry point to the Time of Day handler. This function fails if the Socket Services it addresses are in the system BIOS as the first extension to the Time of Day handler. Register the value returned by this function to this Socket Services with a replacement Socket Services implementation.

> ***Warning***
>
> This function should only be used with the first adapter serviced by a Socket Services handler as returned by Function 83h Get SS Info. If a handler services more than one adapter, subsequent requests to the handler for adapters other than the first adapter return the same information and set the same internal variables.

> ***Warning***
>
> A calling program should not add Socket Services that increase the number of adapters or sockets supported. To provide support for additional adapters and sockets, new Socket Services handlers should be added to the end of the handler chain. Adjusting internal prior handlers should be used only to replace an old Socket Services implementation with an updated version.

Input:	AH	=	9Fh
	AL	=	Adapter number (zero-based)
	BL	=	Mode
		=	00h Get prior handler
		=	01h Set prior handler
	CX:DX		If BL = 1, contains a pointer to a new prior handler. It now returns the entry point of the old prior handler.
Output:	AH	=	00h Successful
		=	01h Bad adapter
			15h Bad function
			18h Busy
	CF	=	0 Successful
		=	1 Error
	CX:DX	=	Pointer to a new prior handler and returns the entry point of the old prior handler.

Cont

Function A0h Get and Set SS Address

> *Warning*
>
> Only issue this function for the first adapter serviced by a Socket Services handler as returned by Function 80h Get SS Info. If a handler services more than one adapter, subsequent requests to the handler for adapters other than the first adapter will return the same information and will set the same internal variables.

This function returns code and data area descriptions and provides a method for passing address mode-specific data area descriptors to a Socket Services handler. If Socket Services must access other memory regions, the value in CX is the number of unique memory regions that Socket Services must address as well as the main data segment.

Card Services uses the entry point returned by this function to establish the appropriate address mode-specific pointers to the code and main data areas before calling the entry point. The entry points returned by this function must receive control from a CALL instruction. The real mode, 16:16, and 16:32 entry points require a FAR CALL. The 00:32 entry point requires a NEAR CALL. When using an entry point that has been returned by this function in all address modes except real mode, the calling program must establish a pointer to the main data area in DS:ESI.

Cont

Function A0h Get and Set SS Address, cont

Input:	AH	=	A0h
	AL	=	Adapter number (zero-based)
	BH	=	Mode
		=	00h Real mode
		=	01h 16:16 Protected mode
		=	02h 16:32 Protected mode
		=	03h 00:32 Protected mode
	BL	=	Subfunction
			00h Socket Services returns the number of additional data areas in this parameter.
			01h Socket Services returns a description of additional data areas in the buffer supplied by the calling program in ES:EDI.
			02h Socket Services accepts the number of mode-specific pointers to additional data areas in the buffer pointed to in ES:EDI.
	ES:EDI	=	Contains a pointer to a buffer supplied by the calling program. The buffer must be the appropriate length.
Output:	AH	=	00h Successful
		=	01h Bad adapter
			02h Bad attribute
			15h Bad function
			16h Bad mode
			18h Busy
	CF	=	0 Successful
		=	1 Error
	CX	=	Number of additional data areas.
			BL = 00h The number of additional data areas in this parameter are returned.
			BL = 01h A description of other data areas in the buffer supplied by the calling program in ES:EDI is returned.
			BL = 02h Socket Services accepts the number of mode-specific pointers to additional data areas in the buffer pointed to in ES:EDI as specified in CX.
	ES:EDI	=	Contains a pointer to a buffer supplied by the calling program. The buffer must be the appropriate length.

Warning

CS selectors should be readable and executable so socket services can reference constant data that may reside in ROM. The calling program must also make sure that socket services has the appropriate privileges to permit access to I/O ports.

Cont

Function A0h Get and Set SS Address, cont

Buffer Table Entry if BL = 00h

Offset	Description
00h	32-bit linear base address of the code segment in system memory.
04h	Limit of the code segment. This value must be less than 64 KB in real mode and 16:16 in protected mode.
08h	Entry point offset. This value must be less than 64 KB in real mode and 16:16 in protected mode.
0Ch	32-bit linear base address of the main data segment in system memory. This field is ignored if 00:32 (flat) protected mode addressing is used.
10h	The limit of the data segment. This value must be less than 64 KB in real mode and 16:16 in protected mode.
14h	The data area offset (only used if 32-bit protected mode address used).

Buffer Table Entry if BL = 01h

Offset	Description
00h	32-bit linear base address of the additional data segment in system memory (ignored if 00:32 (flat) protected mode addressing is used).
04h	Limit of the code segment. This value must be less than 64 KB in real mode and 16:16 in protected mode.
08h	Data area offset (only used if 00:32 (flat) protected mode address used).

Buffer Table Entry if BL = 02h

Offset	Description
00h	32-bit offset (ignored if 16:16 protected mode addressing is used). 16:16 protected mode addressing assumes 0 in this field.
04h	Selector (only used if 00:32 (flat) protected mode address used).
08h	Reserved

Cont

Function A1h Get Access Offsets This function fills the buffer pointed to by ES:EDI with an array of offsets for low-level, adapter-specific, optimized PC Card access routines for adapters that use registers or I/O ports to access PC Card memory. Adapters that access PC Card memory through windows mapped to host system memory do not support this function. All requested offsets must be in the socket services code segment. All sockets on an adapter must use the same entry point for a certain address mode. These offsets can vary for different address modes. A calling program can use the values returned by this function to create an internal table, allowing the routines at these offsets to be called in a way that is appropriate for the address mode they are used in. 16-bit offsets are returned in all modes. The offset must be combined with information returned by Function A0h Get and Set SS Address that describes the location of the code segment. Offsets returned by this function are relative to the code segment. In real address mode, 16:16, and 16:32 address modes, the routines at these offsets use a FAR RET instruction to return to the calling program. This function must be invoked with a FAR CALL instruction. In 00:32 (flat) protected address mode, the routines at the returned offsets use NEAR RET instructions and must be invoked with a NEAR CALL instruction.

Access Offset Order The offsets are returned in the following order:

Offset	Size	Offset Name
00h	Word	Set Address
02h	Word	Set Auto Increment
04h	Word	Read Byte
06h	Word	Read Word
08h	Word	Read Byte with Auto Increment
0Ah	Word	Read Word with Auto Increment
0Ch	Word	Read Words
0Eh	Word	Read Words with Auto Increment
10h	Word	Write Byte
12h	Word	Write Word
14h	Word	Write Byte with Auto Increment
16h	Word	Write Word with Auto Increment
18h	Word	Write Words
1Ah	Word	Write Words with Auto Increment
1Ch	Word	Compare Byte
1Eh	Word	Compare Byte with Auto Increment
20h	Word	Compare Words
22h	Word	Compare Word with Auto Increment

Cont

Function A1h Get Access Offsets, cont

Input:	AH	=	A1h
	AL	=	Adapter number (zero-based)
	BH	=	Mode
		=	00h Real mode
		=	01h 16:16 Protected mode
		=	02h 16:32 Protected mode
		=	03h 00:32 Protected mode
	BL	=	Subfunction 00h Socket Services returns the number of additional data areas in this parameter. 01h Socket Services returns a description of any additional data areas in the buffer supplied by the calling program at ES:EDI. 02h Socket Services accepts the number of mode-specific pointers to additional data areas in the buffer pointed to in ES:EDI specified in CX.
	CX	=	Number of access offsets
	ES:EDI	=	Pointer to a buffer supplied by the calling program for an array of access offsets. CX specifies the number of buffer entries.
Output:	AH	=	00h Successful
		=	01h Bad adapter
			15h Bad function
			16h Bad mode
	CF	=	0 Successful
		=	1 Error
	DX	=	Number of access offsets supported by this Socket Services handler for the specified adapter.
	ES:EDI	=	Pointer to a buffer supplied by the calling program for the array of access offsets. CX has the number of entries.

Cont

Function AEh Vendor-Specific This function handles vendor-specific information. The vendor can add proprietary extensions to Socket Services via this interface. See the vendor technical documentation for additional information.

Input:	AH	=	AEh
	AL	=	Adapter number (zero-based)
			All other registers are vendor-specific
Output:	AH	=	00h Successful
	CF	=	0 Successful
		=	1 Error
			All other registers are vendor-specific

Socket Services Error Codes

Code	Description
00h	Successful
01h	Invalid adapter
02h	Invalid attribute
03h	Invalid base system memory address
04h	Invalid EDC generator
06h	Invalid IRQ level
07h	Invalid card offset
08h	Invalid page
09h	Incomplete read request
0Ah	Invalid window size
0Bh	Invalid socket
0Dh	Invalid window type
0Eh	Invalid Vcc level
0Fh	Invalid Vpp1 and Vpp2 level
11h	Invalid window
12h	Incomplete write request
14h	No card present
15h	Function not supported
16h	Invalid mode
17h	Invalid speed
18h	Busy

Intel Exchangeable Card Architecture (ExCA) Card Service Functions

Type	AL	Function
Client Services	00h	Get Number of Sockets
	02h	Register Client
	03h	Deregister Client
	05h	Register SCB
	06h	Deregister SCB
	0Ah	Get Status
	0Bh	Reset Card
	1Ch	Modify Window
	1Eh	Map Mem Page
Resource Management	19h	Request I/O
	1Ah	Release I/O
	1Bh	Request Memory
	1Dh	Release Memory
	22h	Request IRQ
	23h	Release IRQ
Bulk Memory Services	14h	Open Region
	15h	Read Memory
	16h	Write Memory
	17h	Copy Memory
	18h	Erase Memory
	24h	Close Region
Client Utilities	0Ch	Get First Tuple
	0Dh	Get Next Tuple
	0Eh	Determine First Region
	0Fh	Determine Next Region
	10h	Get First Region
	11h	Get Next Region
	12h	Get First Partition
	13h	Get Next Partition
Advanced Client Services	1Fh	Return SS Entry
	20h	Map Log To Physical
	21h	Map Log Physical To Log
	01h	Initialize
	04h	Enumerate Clients
	07h	Register MTD
	08h	Deregister MTD
	09h	Enumerate MTDs

INT 1Ah PCI Service

PCI BIOS Calls PCI is a way to physically interconnect highly integrated peripheral components and processor/memory systems. PCI BIOS functions provide a software interface to the PCI hardware.

PCI is an Intel specification for a 486 CPU Local Bus standard. The PCI specification also provides the electrical specifications for peripheral chip makers and the logic requirements for a PCI Controller. PCI establishes a local bus standard that permits a large variety of I/O components to be directly connected to the CPU bus using no glue logic. The PCI architecture is essentially a CPU-to-local bus bridge with FIFO buffers. PCI signals are multiplexed.

Unlike other local bus specifications, PCI has a standalone PCI Controller to manage the data transfer between PCI peripherals and the memory/CPU.

PCI Features Up to ten PCI peripherals can be used on the PCI bus, including the PCI Controller and an optional expansion bus controller for EISA, ISA, or MCA. PCI uncouples the CPU from the expansion bus while still maintaining a 33 MHz 32-bit path to peripheral devices. The PCI bus works at 33 MHz and can use either a 32-bit or 64-bit data connection path to the CPU.

The PCI Controller queues reads and writes between the memory/CPU and PCI peripheral devices.

Cont

Concurrent Operation The CPU in a PCI computer runs concurrently with PCI bus mastering peripherals. Although bus mastering peripheral devices are specified, impressive data transfer rates can be achieved without splitting resource utilization between the CPU and a bus mastering device. PCI peripheral devices can operate at data transfer rates up to 33 MBs in an ISA environment.

PCI Bus Mastering Up to ten bus mastering devices can operate simultaneously on the PCI bus. PCI devices can be bus masters, slaves, or a combination of bus master and slave. The PCI specification also provides for full burst mode for both reads and writes. The 486 CPU only permits burst mode on reads.

Multiplexing PCI is a multiplexed version of the Intel 486 and Pentium bus. Multiplexing allows more than one signal to be sent on the same electrical path. The control mechanisms are extended to optimize I/O support.

PCI Device Drivers The system BIOS in a PCI computer provides information about where the device is in memory or I/O space and which interrupt vector the device will generate. This information comes directly from the configuration registers of the peripheral component, not from CMOS RAM or an internal BIOS table. PCI BIOS functions can access these configuration registers and provide this information.

Expansion ROM Code All expansion ROM in a PCI computer must be fully relocatable. It must be able to call a PCI system BIOS function to see where its device was placed in memory or I/O space.

Cont

PCI BIOS Interface All software in a PCI computer should use system BIOS functions to access PCI features. The system BIOS in a PCI computer supports multiple operating and addressing modes. Some PCI system BIOS functions include:

- allows the calling program to find a PCI controller,
- provides access to special PCI functions,
- allows the calling program to determine the interrupt level, and
- allows the calling program to access configuration space (either memory or I/O ports).

PCI BIOS Calls PCI-specific BIOS function calls can be used in real mode, 16-bit protected mode, or 32-bit protected mode. Real mode function calls are made via INT 1Ah AH = B1h. Protected mode access is provided by calling the BIOS through a protected mode entry point, specified by calling INT 1Ah Function B1h AL = 01h/81h PCI BIOS Present.

INT 1Ah Function B1h Calling Conventions Every PCI function can be invoked with two codes: one for 32-bit mode and the other for all other modes. The EAX, EBX, ECX, and EDX registers and all flags may be modified by every function call. All other registers will be preserved. CF indicates the completion status of the function call.

Protected Mode PCI BIOS Function Calls Access the protected mode interface by calling through a protected mode entry point provided by the INT 1Ah Function B1h AL = 01h/81h PCI BIOS Present function. The code segment descriptor must specify protection level 0. All INT 1Ah Function B1h PCI BIOS functions must be invoked with CPL = 0. The code segment descriptor must permit access to the 64 KB of code that starts at the 16-byte boundary immediately below the protected mode entry point.

Cont

Function B1h Subfunction AL = 01/81 PCI BIOS Present This subfunction indicates if the PCI BIOS interface is present. The current PCI BIOS interface version level is also returned. Information about hardware mechanisms for accessing PCI configuration space and PCI Special Cycles support is also provided.

Input:	AH	=	B1h
	AL	=	01h Real mode operation
		=	81h Protected mode operation
	BH	=	EDC generator number (zero-based)
Output:	AH	=	00h PCI BIOS interface present
		=	Any other value is an error code.
	AL	=	Hardware mechanism
			5 1 Special cycle supported via Config mechanism 1
			4 1 Special cycle supported via Config mechanism 2
			1 1 Config. Mechanism 2 supported
			0 1 Config. Mechanism 1 supported
	BH	=	Interface Level Major Version (in BCD)
	BL	=	Interface Level Minor Version (in BCD)
	CF	=	0 PCI BIOS interface present
		=	1 No PCI BIOS interface present
	CL	=	Number of PCI buses (zero-based)
	EDI	=	Physical address of entry point to PCI BIOS functions for protected mode access
	EDX	=	" PCI" character string

Cont

Function B1h Subfunction AL = 02/82 Find PCI Device This subfunction returns the location of PCI devices. Specify the Device ID in CX, Vendor ID in DX, and a Device Index in SI. This function returns the PCI bus number in BL and the Device Number of the specified *(nth)* device in BH.

You can find all PCI devices with the same Vendor ID and Device ID by making consecutive calls to this function and incrementing the Device Index by one each time until code 86h is returned in AH.

Input:	AH	=	B1h
	AL	=	02h Real mode operation
		=	82h Protected mode operation
	CX	=	Device ID (0 through 65535)
	DX	=	Vendor ID (1 through 65534)
	SI	=	Device Index (0 through *n*)
Output:	AH	=	00h Successful 82h Incorrect Device ID
		=	83h Incorrect Vendor ID 86h Device not found
	BL	=	Bits 7-3 Device Number
	BH	=	Bus Number (0 through 255)
	CF	=	0 Successful
		=	1 Error

Cont

Function B1h Subfunction AL = 03/83 Find PCI Class Code This subfunction returns the location of PCI devices with the specified Class Code. Specify the Class Code in ECX and a Device Index in SI. The function returns the Bus Number in BL, the Device Number in BH, and the Function Number of the *nth* device in the bottom three bits of BH.

You can find all PCI devices with the same Class Code by making consecutive calls to this function and incrementing the Device Index by one each time until code 86h is returned in AH.

Input:	AH	=	B1h
	AL	=	03h Real mode operation
		=	83h Protected mode operation
	ECX	=	Class Code in low three bytes
	SI	=	Device Index (0 through *n*)
Output:	AH	=	00h Successful
			86h Device not found
	BL	=	Bits 7-3 Device Number
	BH	=	Bus Number (0 through 255)
	CF	=	0 Successful
		=	1 Error

Cont

Function B1h Subfunction AL = 06/86 Generate Special Cycle This subfunction generates a PCI Special Cycles broadcast on the specified PCI bus.

Input:	AH	=	B1h
	AL	=	06h Real mode operation
		=	86h Protected mode operation
	EDX	=	Special Cycle Data
Output:	AH	=	00h Successful
			81h Function not supported
	CF	=	0 Successful
		=	1 Error

Function B1h Subfunction AL = 08/88 Read Configuration Byte This subfunction reads individual bytes from the configuration space of the specified PCI device.

Input:	AH	=	B1h
	AL	=	08h Real mode operation
		=	88h Protected mode operation
	BL	=	Bits 7-3 Device Number
			Bits 2-0 Function Number
	BH	=	Bus Number (0 through 255)
	DI	=	Register Number (0 through 255)
Output:	AH	=	00h Successful
			84h Incorrect Bus Number
	CF	=	0 Successful
		=	1 Error
	CL	=	Byte that was read

Cont

Function B1h AL = 09/89 Read Configuration Word This function reads words from the configuration space of the specified PCI device. The register number must be a multiple of 2.

Input:	AH	=	B1h
	AL	=	09h Real mode operation
		=	89h Protected mode operation
	BL	=	Bits 7-3 Device Number
			Bits 2-0 Function Number
	BH	=	Bus Number (0 through 255)
	DI	=	Register Number (0 through 255)
Output:	AH	=	00h Successful
			84h Incorrect Bus Number
			87h Incorrect Register Number
	CF	=	0 Successful
		=	1 Error
	CX	=	Word that was read

Function B1h AL = 0A/8A Read Configuration Dword This function reads individual doublewords from the specified PCI device configuration space. The register number must be a multiple of 4.

Input:	AH	=	B1h
	AL	=	0Ah Real mode operation
		=	8Ah Protected mode operation
	BL	=	Bits 7-3 Device Number
			Bits 2-0 Function Number
	BH	=	Bus Number (0 through 255)
	DI	=	Register Number (0 through 255)
Output:	AH	=	00h Successful
			84h Incorrect Bus Number
			87h Incorrect Register Number
	CF	=	0 Successful
		=	1 Error
	ECX	=	Doubleword that was read

Cont

Function B1h AL = 0B/8B Write Configuration Byte This subfunction writes individual bytes to the configuration space of the specified PCI device.

Input:	AH	=	B1h
	AL	=	0Bh Real mode operation
		=	8Bh Protected mode operation
	BL	=	Bits 7-3 Device Number
			Bits 2-0 Function Number
	BH	=	Bus Number (0 through 255)
	CL	=	Byte value to write
	DI	=	Register Number (0 through 255)
Output:	AH	=	00h Successful
			84h Incorrect Bus Number
	CF	=	0 Successful
		=	1 Error

Function B1h AL = 0C/8C Write Configuration Word Writes individual words to the configuration space of the specified PCI device. The Register Number must be a multiple of 2.

Input:	AH	=	B1h
	AL	=	0Ch Real mode operation
		=	8Ch Protected mode operation
	BL	=	Bits 7-3 Device Number
			Bits 2-0 Function Number
	BH	=	Bus Number (0 through 255)
	CX	=	Word value to write
	DI	=	Register Number (0 through 255)
Output:	AH	=	00h Successful
		=	84h Incorrect Bus Number
		=	87h Incorrect Register Number
	CF	=	0 Successful
		=	1 Error

Cont

Function B1h Subfunction AL = 0D/8D Write Configuration Dword

This subfunction writes individual doublewords to the configuration space of the specified PCI device. The Register Number must be a multiple of 4.

Input:	AH	=	B1h
	AL	=	0Dh Real mode operation
		=	8Dh Protected mode operation
	BL	=	Bits 7-3 Device Number
			Bits 2-0 Function Number
	BH	=	Bus Number (0 through 255)
	ECX	=	Doubleword value to write
	DI	=	Register Number (0 through 255)
Output:	AH	=	00h Successful
		=	84h Incorrect Bus Number
		=	87h Incorrect Register Number
	CF	=	0 Successful
		=	1 Error

INT 1Ah Function B1h Error Codes The INT 1Ah Function B1h error codes in AH are:

AH Value	Description
00h	Successful
81h	Function Not Supported
82h	Incorrect Device ID
83h	Incorrect Vendor ID
84h	Incorrect Bus Number
86h	Device Not Found
87h	Incorrect Register Number
EEh	Internal Error

Cont

Function B1h Subfunction AL = 0E/8E Get IRQ Routing Information

This subfunction returns a bitmap of the IRQ channels that have been permanently assigned to PCI in BX.

Input:	AH	=	B1h
	AL	=	0Eh Real mode operation
		=	8Eh Protected mode operation
	BH	=	00h
	BL	=	00h
	ES:EDI	=	IRQ routing table header
	DS	=	Segment or selector for PCI BIOS data
			F000h Real mode
			16-bit PM Physical 000F0000h
			32-bit PM As specified by BIOS32 services directory.
Output:	AH	=	00h Successful
	AH	=	All other values are error codes.
	BX	=	Bitmap of IRQ channels permanently dedicated to PCI
	ES:DI		Size of returned data
	CF	=	0 Successful
		=	1 Error

Cont

Function B1h Subfunction AL = 0F/8F Set PCI IRQ This subfunction sets the PCI IRQ routing. assumes that the calling application has determined the IRQ routing topology, has made sure that the selected IRQ will not cause a conflict, and will update the interrupt line configuration register on all devices that currently use the IRQ line

Input:	AH	=	B1h
	AL	=	0Fh Real mode operation
		=	8Fh Protected mode operation
	BH	=	Bus number
	BL	=	Device and function number Bits 7-3 Device number Bits 2-0 Function number
	CH	=	Number of IRQ to connect
	CL	=	Number of interrupt pin to reprogram 0Ah INTA 0Bh INTB 0Ch INTC 0Dh INTD
	DS	=	Segment or selector for PCI BIOS data F000h Real mode 16-bit PM Physical 000F0000h 32-bit PM As specified by BIOS32 services directory.
Output:	AH	=	00h Successful
	AH	=	Error if any other value
	CF	=	0 Successful
		=	1 Error

Cont

Function B1h AL = 81h 32-bit Installation Check This subfunction indicates if the PCI BIOS interface is present. The current PCI BIOS interface version level is also returned. Information about hardware mechanisms for accessing PCI configuration space and PCI Special Cycles support is also provided.

Input:	AH	=	B1h
	AL	=	81h
Output:	AH	=	00h PCI BIOS interface present
		=	Any other value is an error code.
	AL	=	Hardware mechanism
			5 1 Special cycle supported via Config mechanism 1
			4 1 Special cycle supported via Config mechanism 2
			1 1 Config. Mechanism 2 supported
			0 1 Config. Mechanism 1 supported
	BH	=	Interface Level Major Version (in BCD)
	BL	=	Interface Level Minor Version (in BCD)
	CF	=	0 PCI BIOS interface present
		=	1 No PCI BIOS interface present
	CL	=	Number of PCI buses (zero-based)
	EDI	=	Physical address of entry point to PCI BIOS functions for protected mode access
	EDX	=	" PCI" character string

Cont

Function B1h AL = 82h 32-bit Find PCI Device This subfunction indicates if the PCI BIOS interface is present. The current PCI BIOS interface version level is also returned. Information about hardware mechanisms for accessing PCI configuration space and PCI Special Cycles support is also provided.

Input:	AH	=	B1h
	AL	=	82h
	CX	=	Device ID (0 – 65535)
	DX	=	Vendor ID (0 – 65534)
	SI	=	Device index (0 through n)
Output:	AH	=	00h Successful 82h Incorrect Device ID
		=	83h Incorrect Vendor ID 86h Device not found
	BH	=	Bits 7-3 Device Number Bus Number (0 through 255)
	CF	=	0 Successful
		=	1 Error

Function B1h Subfunction AL = 83h 32-Bit Find PCI Class Code This subfunction finds the PCI class code.

Input:	AH	=	B1h
	AL	=	83h
	ECX	=	Bits 23-0 Class code
	SI	=	Device index (0 – n)
Output:	AH	=	00h Successful 86h Device not found
	BL	=	Bits 7-3 Device Number
	BH	=	Bus Number (0 through 255)
	CF	=	0 Successful
		=	1 Error

Cont

Function B1h Subfunction AL = 86h 32-Bit Special Cycle This subfunction generates a PCI Special Cycle broadcast on the specified PCI bus.

Input:	AH	=	B1h
	AL	=	86h
	BH	=	Bus number
	EDX	=	Special cycle data
Output:	AH	=	00h Successful
			81h Function not supported
	CF	=	0 Successful
		=	1 Error

Function B1h Subfunction AL = 88h 32-Bit Read Configuration Byte This subfunction reads the PCI configuration byte.

Input:	AH	=	B1h
	AL	=	88h
	BH	=	Bus number
	BL	=	Device and function number
			Bits 7-3 Device number
			Bits 2-0 Function number
	DI	=	Register number (0000h-00FFh)
Output:	AH	=	B1h Successful
	AL	=	08h Successful

Function B1h Subfunction AL = 89h 32-Bit Read Configuration Word This subfunction reads the PCI configuration word.

Input:	AH	=	B1h
	AL	=	89h
	BH	=	Bus number
	BL	=	Device and function number
			Bits 7-3 Device number
			Bits 2-0 Function number
	DI	=	Register number (0000h-00FFh)
Output:	AH	=	B1h Successful
	AL	=	09h Successful

Cont

Function B1h Subfunction AL = 8Ah 32-Bit Read Configuration Dword

This subfunction reads the PCI configuration Dword.

Input:	AH	=	B1h
	AL	=	8Ah
	BH	=	Bus number
	DI	=	Register number (0000h-00FFh)
Output:	AH	=	00h Successful
	AL	=	0Ah Successful

Function B1h Subfunction AL = 8Bh 32-Bit Write Configuration Byte

This subfunction writes the PCI configuration byte.

Input:	AH	=	B1h
	AL	=	8Bh
	BH	=	Bus number
	BL	=	Device and function number
			Bits 7-3 Device number
			Bits 2-0 Function number
	CL	=	Byte to be written
	DI	=	Register number (0000h-00FFh)
Output:	AH	=	B1h Successful
	AL	=	0Bh Successful

Function B1h Subfunction AL = 8Ch 32-Bit Write Configuration Word

This subfunction writes the PCI configuration word.

Input:	AH	=	B1h
	AL	=	8Ch
	BH	=	Bus number
	BL	=	Device and function number
			Bits 7-3 Device number
			Bits 2-0 Function number
	CX	=	Word to write
	DI	=	Register number (0000h-00FFh)
Output:	AH	=	B1h Successful
	AL	=	0Ch Successful

Cont

Function B1h Subfunction AL = 8Dh 32-Bit Write Configuration Dword This subfunction writes the PCI configuration Dword.

Input:	AH	=	B1h
	AL	=	8Dh
	BH	=	Bus number
	ECX	=	Dword to write
	DI	=	Register number (0000h-00FFh)
Output:	AH	=	00h Successful
	AL	=	0Dh Successful

Desktop Management Interface and Plug and Play Functions The Desktop Management Interface (DMI) is a new way to manage computers. DMI parallels the Plug and Play initiative. It specifies methods for making computer upgrades much easier. The Desktop Management BIOS Specification follows the system device node model used in the Plug and Play BIOS specification. DMI uses Plug and Play functions to access DMI information. Plug and Play functions 50h — 57h have been assumed by the DMI BIOS interface. DMI specifies a database of system information (the Management Information Format (MIF) database. Each computer can have a number of MIF files that contain information about the motherboard, adapter cards, and other computer components. Using a utility program that can read MIF files, you can obtain a great deal of information about any DMI-aware computer. The American Megatrends PC Care diagnostics utility can provide this type of information.

Plug and Play (PnP) BIOS functions provide the runtime Plug and Play interface between the system BIOS and systems software. Systems software may detect the presence of the PnP BIOS functions by searching for a Plug and Play BIOS signature from F0000h to FFFFFh. The signature marks the beginning of a structure that contains basic information about the Plug and Play BIOS implementation. The signature and this structure are shown below.

DMI Data Structures The standard DMI data structures are described in Chapter 13, beginning on page 错误!未定义书签。.

Cont

System BIOS Plug and Play Signature

Offset	Length	Description
00h	Dword	Plug and Play BIOS Signature: ASCII characters $PnP
04h	Byte	Revision number of Plug and Play BIOS specification
05h	Byte	Length
06h	Word	Control flags Bits 15-2 Reserved Bits 1-0 Event notification method (see the GetEvent function) 00 Event notification is not supported. 01 Event notification is done via polling method. 10 Event notification performed via asynchronous method.
08h	Byte	Checksum
09h	Dword	Event notification flag address (if using polling method). Bit 0 of the byte at this address is set by the BIOS to signal systems software that an event has occurred (see GetEvent function).
0Dh	Word	Real mode entry point for Plug and Play BIOS functions (offset part).
0Fh	Word	Real mode entry point for Plug and Play BIOS functions (segment part)
11h	Word	Protected mode entry point for Plug and Play BIOS functions (offset part)
13h	Dword	Protected mode entry point for Plug and Play BIOS functions (segment base address part)

Cont

Calling Plug and Play Functions Plug and Play BIOS functions are called from real or protected mode by passing parameters on the stack as in the C language calling convention. The first parameter (the last parameter to be pushed onto the stack before calling) is the Plug and Play BIOS function number being called. All flags and registers are preserved. Return codes are in the AX register (EAX if called via protected mode entry point). The PnP BIOS return codes are shown below.

DMI and Plug and Play BIOS Error Codes

Value	Return Code Name	Description
00h	SUCCESS	DMI/PnP BIOS function completed successfully.
81h	UNKNOWN_FUNCTION	Caller passed invalid function number on the stack.
82h	FUNCTION_NOT_SUPPORTED	The DMI or PnP BIOS does not support the called function.
83h	INVALID_NODE_NUMBER	The DMI structure number or handle is not valid or an invalid PnP node number was passed to function 01h or 02h.
84h	BAD_PARAMETER	Bad parameter passed by the calling program.
85h	SET_FAILED	SetSystemDeviceNode (function 02h) failed. (PnP only).
86h	EVENTS_NOT_PENDING	No events pending.
87h	SYSTEM_NOT_DOCKED	Computer is not attached to a docking station. (PnP only)
88h	NO_ISA_PNP_CARDS	No PnP ISA adapter cards installed. (PnP only)
8Ch	BUFFER_TOO_SMALL	The DMI memory buffer specified by the calling program is not large enough to hold the data returned by the BIOS.

Cont

Function B4h Subfunction AL - 00h Plug and Play Autoconfiguration Installation Check This function checks if the Plug and Play Autoconfiguration option has been installed.

Input:	AH	=	B4h
	AL	=	00h
Output:	AX	=	0000h Installed
		=	0001h Specified action could not be completed
		=	0055h Unable to read/write configuration table from or to CMOS RAM
		=	0056h Not a valid configuration table or incorrect table version
		=	0059h Buffer too small
		=	0081h Unsupported function
	BH	=	ACFG major version (02h)
	BL	=	ACFG minor version (08h)
	CF	=	0 Installed
		=	1 Not installed
	CX	=	0002h
	EDX	=	47464341h 'GFCA' is a byte-swapped 'ACFG'.
	SI	=	001Fh

Function B4h Subfunction AL = 01h Autoconfiguration Get Default Configuration Table This subfunction gets the default Plug and Play configuration table.

Input:	AH	=	B4h
	AL	=	01h
Output:	AX	=	0000h Installed
		=	0001h Specified action could not be completed
		=	0055h Unable to read/write configuration table from or to CMOS RAM
		=	0056h Invalid configuration table or version
		=	0059h Buffer too small
		=	0081h Unsupported function
	BX	=	Maximum size of configuration table in bytes
	CF	=	0 Installed
		=	1 Not installed
	CX	=	Required configuration buffer size (includes scratch space used by ACFG code)
	EDI	=	linear/physical address of ESCD table
	SI	=	001Fh

Cont

Plug and Play Extended System Configuration Data Table This table contains information about the standard devices in the system, such as serial ports, parallel ports, etc. For each device, it includes at least the base I/O port address, the sum of all words in the table including the checksum field, with zero padding if the length is odd. The checksum must equal 0000h.

Offset	Size	Description
00h	Word	Total length of this table
02h	4 bytes	Signature "ACFG"
06h	Byte	Minor version number
07h	Byte	Major version number (currently 02h)
08h	Byte	Number of boards in the configuration data
09h	3 bytes	Reserved (00h)
0Ch	variable	Board data
varies	Word	Checksum

Extended System Configuration Data Board Header

Offset	Size	Description
00h	Word	Length of this header in bytes
02h	Byte	Slot number 00h Motherboard 01h-0Fh ISA or EISA board 10h-40h PCI board
03h	Byte	Reserved (00h)

Cont

Extended System Configuration Data Freeform Board Header

Offset	Size	Description
00h	4 bytes	Signature "ACFG"
04h	Byte	Minor version number
05h	Byte	Major version number (currently 02h)
06h	Byte	Board type 01h ISA 02h EISA 04h PCI 08h PCMCIA 10h ISA PnP 20h MCA
07h	Byte	Reserved (00h)
08h	Word	Disabled functions (bit N set = function N disabled)
0Ah	Word	Configuration error functions
0Ch	Word	Reconfigurable functions (bit N set = function N reconfigurable)
0Eh	Word	Reserved (00h)

Extended System Configuration Data Freeform PCI Device Data

Offset	Size	Description
00h	Byte	PCI bus number
01h	Byte	PCI device and function number
02h	Word	PCI device identifier
04h	Word	PCI vendor ID
06h	two bytes	Reserved (00h)

ESCD Freeform PnP ISA Board ID

Offset	Size	Description
00h	DWORD	Vendor ID (EISA device identifier)
04h	DWORD	Serial number

Additional Information See the Plug and Play Extended System Configuration Data Specification for information about additional ESCD tables.

Cont

GetDeviceNode **00** This function reports the number and maximum size of motherboard device nodes. Each node represents one motherboard device.

Prototype

```
int FAR PnP BIOS Call (Function, pNodeCount, pNodeSize);

int     Function;           = 00h
unsigned FAR * pNodeCount;  = Returned: number of nodes
unsigned FAR * pNodeSize;   = Returned: size of largest node

Return value:   0 if successful
        non-zero otherwise (see Plug and Play BIOS return codes)
```

GetSystemDeviceNode 01 This function copies the requested device node to the buffer that you must provide. The device node structure is described in the Plug and Play Specification version 1.0 and is summarized below. The resource configuration of most motherboard devices is fixed (for example: DMA channel 0 is always used for refresh). The device nodes of these fixed configuration devices contain only one set of resource settings. However, some motherboard devices may be configured at different resource settings (for example: an onboard serial port can be located at I/O port address 03F8h, 02F8h, 03E8h, or 02E8h). It can be configured with different resource settings; its resource node contains all possible resource options as well as the option currently active.

Prototype

```
int FAR Plug and Play BIOS Call (Function, pNodeNumber, pDevNodeBuffer);

int         Function;               = 01h
unsigned char FAR * pNodeNumber;    = Node number (0 .. NodeCount)

DEV_NODE FAR *  pDevNodeBuffer;     = Returned: node data

Return value:   0 if successful
        non-zero otherwise (see Plug and Play BIOS return codes)
```

Cont

Plug and Play System Device Node Structure

Offset	Length	Description
00h	Word	Size of this node in bytes.
02h	Byte	Node number of this node.
03h	Dword	Device ID (see Plug and Play specification)
07h	Three bytes	Device type code (see Plug and Play specification).
0Ah	Word	Node attributes Bits 15-6 Reserved Bit 5 1 Device is a docking station device. Bit 4 1 Device can be a boot IPL device. Bit 3 1 Device can be a boot input device. Bit 2 1 Device can be a boot output device. Bit 1 0 Device supports dynamic reconfiguration. 1 Device is static Bit 0 1 Device cannot be disabled.
0Ch	Variable	Resource descriptors that are currently active (see System Device Nodes).
variable	Variable	Resource descriptors of all possible resource settings (see System Device Nodes).
variable	Variable	Compatible device IDs (see System Device Nodes).

SetSystemDeviceNode 02 This function is used by systems software to dynamically change the resource configuration of motherboard devices that can use one of several resource options. For example, an onboard serial port may be located at I/O port address 03F8h, 02F8h, 03E8h, or 02E8h. Systems software can use this call to dynamically configure this serial port at any one of its possible port options. The device node structure is described in the Plug and Play Specification version 1.0.

Prototype

```
int FAR PnP BIOS Call (Function, pNodeNumber, pDevNodeBuffer);
int         Function;           = 02h
unsigned char NodeNumber;           = Node number (0 .. NodeCount)
DEV_NODE FAR * pDevNodeBuffer;  = New node settings
Return value:   0 if successful
        non-zero otherwise (see Plug and Play BIOS return codes)
```

Cont

GetISAConfigurationStructure 40 Systems software uses this call to retrieve the number of Card Select Numbers (CSNs) assigned by the BIOS during initialization and the I/O port addresses to be used for ISA Read Data and AEN Control.

Prototype

```
int FAR Plug and Play BIOS Call (Function, pISAConfig);

int         Function;           = 40h
ISA_Config FAR * pISAConfig;    = Returned: ISA configuration structure
                                  (see below)

Return value:   0           Successful
                non-zero    Otherwise (see Plug and Play BIOS return codes)
```

ISA PnP Configuration Structure

Offset	Length	Description
00h	Byte	Revision The revision number of the Plug and Play BIOS functions.
01h	Byte	CSNCount During initialization, the BIOS assigns CSNs to Plug and Play ISA devices starting at CSN 1 and continuing until all Plug and Play ISA devices have been assigned a CSN. This field contains the last CSN that the BIOS assigned to a Plug and Play ISA device.
02h	Word	ISAReadPort During initialization, the BIOS finds a conflict-free I/O port address to use for reading from Plug and Play ISA device configuration registers. This field contains the current ISA Read Data I/O port address.
04h	Word	AENControlPort Some hardware implementations may provide a method for controlling the AEN signal on an individual slot basis. This field contains the I/O port address that implements this feature.

Cont

GetEvent **03** In some computers, you can insert or remove devices while the computer is powered on, such as inserting a notebook computer in a docking station. The insertion or removal of PCMCIA cards is still handled by Socket Services. In these computers, the BIOS must notify systems software of dynamic events that affect the availability of devices and resources. There are currently two methods of notification: polled event notification, and asynchronous event notification.

PnP Polled Event Notification The Plug and Play BIOS signature, located between F0000h and FFFFFh, contains a 32-bit pointer to the Event Notification Flag. When a system event is detected, the BIOS sets bit 0 of the Event Notification Flag, signaling systems software that an event has occurred. The operating system monitors this flag. When the flag is set, the operating system calls the BIOS GetEvent function to determine the type of event that occurred.

PnP Asynchronous Event Notification Asynchronous Event Notification is not well defined in the Plug and Play specification.

Systems software may install a notification function that the BIOS has to call to notify the operating system of an event. The operating system then calls the BIOS GetEvent function similar to the Polled Event Notification method described above.

The GetEvent function returns a 16-bit code that specifies the type of event that just occurred. The defined event types are described in the table on the next page.

Cont

PnP Event Types

Event Name	Code	Description
SYSTEM_ABOUT_TO_DOCK	01h	Notifies the operating system that the computer will be inserted in a docking station. Computers with software control of the docking sequence should delay docking until the operating system sends an OK_TO_DOCK message to the BIOS via SendMessage.
SYSTEM_ABOUT_TO_UNDOCK	02h	Notifies the operating system that the computer will be removed from a docking station. Computers with software control of the docking sequence should delay docking until the operating system sends an OK_TO_DOCK message to the BIOS via SendMessage.
SYSTEM_DOCKED	03h	Notifies the operating system that the computer docked successfully.
SYSTEM_NOT_DOCKED	04h	Notifies the operating system that the computer did not dock successfully.
SYSTEM_UNDOCKED	05h	Notifies the operating system that the computer was successfully removed.
SYSTEM_NOT_UNDOCKED	06h	Notifies the operating system that the computer was not successfully removed.
SYSTEM_DEVICE_INSERTED	07h	A hot pluggable device (such as an external floppy) was added.
SYSTEM_DEVICE_REMOVED	08h	A hot pluggable device was removed from the computer.
UNKNOWN_SYSTEM_EVENT	09h	An unknown system event occurred.

Prototype

```
int FAR Plug and Play BIOS Call (Function, pMessage);
int         Function;           = 03h
unsigned int FAR * pMessage;    = Returned: Event code from table above

Return value:   0 if successful
           non-zero otherwise (see Plug and Play BIOS return codes)
```

Cont

SendMessage **04** This function is used by the operating system to send messages to the BIOS to manage system events. See the GetEvent function. The messages are:

Message Name	Code	Description
OK_TO_DOCK	01h	The operating system sends this message after SYSTEM_ABOUT_TO_DOCK to tell the BIOS to dock.
OK_TO_UNDOCK	02h	The operating system sends this message after SYSTEM_ABOUT_TO_UNDOCK to tell the BIOS to undock.
ABORT_DOCK	03h	The operating system sends this message after SYSTEM_ABOUT_TO_DOCK to tell the BIOS to abort docking.
ABORT_UNDOCK	04h	The operating system sends this message after SYSTEM_ABOUT_TO_UNDOCK to tell the BIOS to abort the removal.
UNDOCK_POWER_OFF	05h	The operating system sends this message to instruct the BIOS to power off and remove the computer. This message allows the operating system to implement a soft eject and power down operation.
UNDOCK_STANDBY	06h	The operating system sends this message to the BIOS to remove the computer and place the computer in Standby mode.

Prototype

```
int FAR Plug and Play BIOS Call (Function, Message);
int         Function;       = 04h
unsigned int Message;       = Message code from table above
Return value:   0 if successful
            non-zero otherwise (see Plug and Play BIOS return codes)
```

GetDockingStationIdentifier 05 The operating system issues this function to get the docking station product ID. This function allows a notebook computer to be used in more than one type of docking station (each may make available a different set of expansion devices or resources). The product ID of the docking station is returned in the buffer that you must provide. This function returns FFFFh if the current docking station has no product ID.

Prototype

```
int FAR Plug and Play BIOS Call (Function, pStationID);
int         Function;           = 05h
unsigned int FAR * pStationID;  = Returned: Product ID of docking station
Return value:   0 if successful
        non-zero otherwise (see Plug and Play BIOS return codes)
```

Cont

SelectPrimaryBootDevices 07 This function allows systems software to select the boot devices. Three devices must be selected to participate in a boot sequence:

- the primary input devices,
- the primary output device, and
- the initial program load device.

Device	Description
Primary Input Device	The primary input device is normally the keyboard. If this device is not the standard keyboard, it must provide an option ROM that implements the standard INT 09h interface on the device.
Primary Output Device	The primary output device is normally a video adapter card. If this device is not a standard video display, it must provide an option ROM that implements the standard INT 10h interface on the device.
Initial Program Load Device	The initial program load (IPL) device is normally a hard disk drive. If this device is not a standard floppy or hard drive controller, it must either: • provide an option ROM that implements the standard INT 13h interface on the device, or • provide a Plug and Play option ROM that contains a valid Bootstrap Entry Point.

Set Product IDs This function sets the three 32-bit Product IDs that the BIOS should use for the each of the three boot devices.

If the selected boot device is not a standard ISA boot device (Device ID set to FFFFFFFFh), the system BIOS checks the state of the INT 19h vector before issuing INT 19h. If an ISA device option ROM has hooked INT 19h, the system BIOS resets the vector to point back to the BIOS. The system BIOS saves the hooked address of INT 19h in case the attempt to boot to the selected device fails. and then the system BIOS restores the INT 19h vector to its hooked value and attempts to boot to the standard ISA device option ROM that originally hooked INT 19h.

Cont

SelectPrimaryBootDevices 07, cont

Prototype

```
int FAR Plug and Play BIOS Call (Function, Type, DeviceID, Unit, ControlFlags,
pPrefResources);
int         Function;           = 07h
int         Type;               = Type of device being selected
unsigned long DeviceID;         = Product ID
unsigned long SerialNum;        = Serial number
unsigned long LogicalDeviceID;  = Logical device ID
int         Unit;               = Unit number on device
int         ControlFlags        = Misc. flags (see below)
RES_DATA FAR *  pPrefResources  = Preferred resource configuration
Return value:   0 if successful
        non-zero otherwise (see Plug and Play BIOS return codes)
```

SelectPrimaryBootDevices Parameters The parameters passed to the SelectPrimaryBootDevices function are:

Parameter	Description
Type	This parameter specifies the boot device set by this call: 0 Set primary input device 1 Set primary output device 2 Set initial program load (IPL) device
DeviceID, SerialNum, LogicalDeviceID	These parameters indicate the device to be used. These parameters must be set to one of the following: • The Device Product ID field of a node that was returned by the GetSystemDeviceNode function. • The Device ID fields of a Plug and Play adapter card (FFFFFFFFh for standard ISA compatible boot device).
Unit	This parameter is meaningful only when selecting the IPL device. This parameter specifies unit number for controllers connected to more than one physical device. The option ROM of the IPL device can call the GetPrimaryBootDevices function to retrieve this unit number.
ControlFlags	Bits defined in the ControlFlags word are: Bits 15-1 Reserved Bit 0 0 Make sure a device is actually attached before attempting to boot. 1 Do not check for an attached device before attempting to boot.
pPrefResources	A pointer to a block of resource data that specifies the preferred resource configuration for the boot device. If no preferred resource settings are specified, this parameter points to an End Tag Identifier. The structure of this resource block and the End Tag Identifier are described in the Plug and Play ISA Specification version 1.0.

Cont

GetPrimaryBootDevices 08 This function retrieves information about the devices that the computer actually booted from. The information is supplied in parameters similar to the SelectPrimaryBootDevices function parameters.

Prototype

```
int FAR Plug and Play BIOS Call (Function, pType, pDeviceID, pUnit, pPrefResources);
int         Function;       = 07h
int FAR * pType;                = Returned: Type of device being selected
unsigned long FAR * pDeviceID;  = Returned: Product ID
unsigned long FAR * pSerialNum; = Returned: Product serial number
unsigned long FAR * pLogicalDeviceID;   = Returned: Product logical device ID
int FAR * pUnit;                        = Returned: Unit number on device
RES_DATA FAR * pPrefResources   = Returned: Preferred resource configuration
Return value:   0 if successful
        non-zero otherwise (see Plug and Play BIOS return codes)
```

Plug and Play Option ROMs

Plug and Play introduces a new option ROM format, which is an extension of the existing ISA option ROM format. The new Plug and Play Option ROM enables to BIOS to boot from a range of non-standard boot devices. In a Plug and Play computer, three devices must participate in the boot process:

- primary input device,
- primary output device, and
- initial program load (IPL) device.

Option ROMs Required for Nonstandard Devices Nonstandard devices that cannot be controlled by the system BIOS must provide an option ROM. Existing option ROMs gain control once during POST and must hook any vectors necessary to boot from the device. This method is inflexible and prevents the system BIOS from selecting a boot device. The PnP option ROM provides a more flexible and controllable approach. All Plug and Play devices that can be a boot device should include an option ROM that conforms to the PnP specification.

Cont

Types of PnP Option ROMs

Device	Description
Primary Input Device Option ROM	The Plug and Play primary input boot devices must provide an option ROM that implements the standard INT 09h interface on the device.
Primary Output Device Option ROM	Plug and Play primary output devices must provide an option ROM that implements the standard INT 10h interface on the device.
Initial Program Load Device Option ROM	Plug and Play initial program load (IPL) devices must provide an option ROM that does one of the following: • if the device is a traditional block device, the option ROM must provide the standard INT 13h interface on the device, • if the device is not a traditional block device, the option ROM must contain a valid Bootstrap Entry Point called directly by the system BIOS to boot to the device.

Plug and Play option ROMs include an expanded header to permit them to be identified as Plug and Play ROMs by the system BIOS. Standard (non-Plug and Play) ISA devices may be retrofitted with new option ROMs that conform to the Plug and Play standard. While these devices are not software-configurable, they can be recognized more easily by the system BIOS.

Option ROM Header Format

Offset	Size	Description
00h	Word	Signature (AA55h)
02h	Byte	Length of option ROM in units of 512 bytes.
03h	four bytes	JMP instruction initialization code.
07h	19 bytes	Reserved for OEM copyright messages or other data.
1Ah	Word	Offset to first expansion header structure.

The PnP option ROM header is a superset of the standard option ROM header. The word at offset 1Ah is a pointer to the first expansion header structure (see below). Expansion headers can be chained together in a linked list. In this manner, new expansion header formats can be added.

Option ROM Device Driver Initialization Model Option ROMs should support a Device Driver Initialization Model (DDIM) to reduce the amount of memory space required. Option ROMs can also use a DDIM to store POST data.

Cont

Installing an Option ROM that Supports DDIM The following occurs when installing an option ROM that supports DDIM:

Step	Action
1	The system BIOS copies the DDIM option ROM to RAM (read and write shadow).
2	The system BIOS executes a FAR CALL to offset 3 in the ROM segment.
3	The option ROM initializes its device.

The option ROM detects DDIM support by attempting to write and read a data pattern somewhere in its memory image. If DDIM is not supported, the option ROM exits.

If DDIM Support is Provided in System BIOS If DDIM support is detected, the option ROM:

Step	Action
1	Builds or updates any data structures inside its memory image.
2	Adjusts its length field at offset 02h of its segment to remove its initialization code.
3	Recalculates and adjusts its checksum.
4	Returns to the system BIOS.
5	The system BIOS write-protects the option ROM memory.

The system BIOS may now map the next option ROM to the next 2 KB boundary following the end of the most recently initialized DDIM option ROM.

Option ROM Boot Connection Routine PnP Option ROMs for devices that can act as a boot device must supply a boot connection routine. During POST, the system BIOS calls this routine in any of three boot devices controlled by an option ROM. The system BIOS passes the following information to the boot connection routine in AX:

Bits	Information
15 – 3	Reserved (set to 0).
2	1 This device is being used as the primary input device and should hook the INT 09h vector at this time.
1	1 This device is being used as the primary output device and should hook the INT 10h vector at this time.
0	1 This device is being used as an IPL device and should hook the INT 13h vector at this time (if providing an INT 13h interface).

Cont

Option ROM Boot Disconnection Routine PnP Option ROMs for devices that can be an IPL boot device must supply a Boot Disconnection Routine. The system BIOS calls this routine to do cleanup after an unsuccessful boot attempt.

Option ROM Bootstrap Entry Point PnP Option ROMs for devices that can be an IPL boot device but do not provide an INT 13h interface must supply a bootstrap entry point. The system BIOS executes a FAR CALL to this entry point instead of an INT 19h to boot the computer.

Option ROM Get Static Resource Usage Routine A non-Plug and Play ISA device may include an Option ROM with a Get Static Resource Usage routine to report device resource use. This routine copies device node information to the buffer that you must provide. The device node structure is described in PnP Specification version 1.0. This routine must provide the following C calling interface. This is the same interface as the GetSystemDeviceNode function.

Prototype

```
int FAR GetStaticResourceUsage (pDevNodeBuffer);
DEV_NODE FAR * pDevNodeBuffer;   = Returned: node data
Return value:  0 if successful
      non-zero otherwise (see Plug and Play BIOS return codes)
```

Transferring Control to the Operating System After loading and validating the operating system boot sector, the system BIOS transfers control to the boot sector code by passing the following parameters in the following registers:

Register	Value to be Loaded
ES:DI	Pointer to PnP System BIOS signature.
DL	Physical INT 13h device number that the operating system is being loaded from (for example: 80h).

A Plug and Play operating system verifies that the computer includes a PnP BIOS and uses the value in DL to make subsequent INT 13h calls, allowing the operating system to be loaded from any INT 13h drive. A non-Plug and Play operating system can ignore this information.

Cont

Plug and Play Option ROMs, Continued

Boot Error Recovery After the BIOS passes control to the IPL device boot sector in the ISA boot architecture, there is no way to return to the BIOS if an error occurs. In the PnP boot architecture, the IPL code issues INT 19h or INT 18h if an error occurs during boot. The BIOS traps these vectors and loads from another boot device.

Plug and Play BIOS Expansion Header Structure

Offset	Length	Description
00h	4 bytes	Signature. The ASCII characters *$PnP* (byte 0-24h, byte 1-50h, etc).
04h	Byte	Header Revision - This version of the header is version 01h.
05h	Byte	Header Length in paragraphs (16 bytes).
06h	Word	Offset of Next Header - Offset of the next expansion header structure in the linked list (0000h is this is the last header).
08h	Byte	Reserved (set to FFh).
09h	Byte	Checksum adjust - The sum of all bytes in the expansion header must be zero. The length is at offset 05h.
0Ah	Dword	Device Identifier - The Plug and Play device identifier is used in SelectPrimaryBootDevices and GetPrimaryBootDevices.
0Eh	Word	Offset of Manufacturer String - Offset to the option ROM base address of an ASCIIZ string with the manufacturer name. 0000h if no string.
10h	Word	Offset of Product Name String - Offset relative to the option ROM base address of an ASCIIZ string containing the product name. Set to 0 if there is no string.
12h	3 bytes	Device Type Code. Identifies the function of the device (for example: mass storage, video, etc.). The system BIOS can use this information to prioritize boot devices if no boot device has been explicitly selected.
15h	Byte	Device Indicators Bit 7 1 ROM supports the Device Driver Initialization Model. Bit 6 1 ROM may be copied to shadow RAM. Bit 5 1 ROM may be stored in secondary cache memory. Bit 4 0 ROM may be mapped out of the system address space or disabled. 1 ROM is only needed if this device is a boot device. Bit 3 Reserved (set to 0). Bit 2 1 This device can be an IPL device Bit 1 1 This device can be a primary input device. Bit 0 1 This device can be a primary output device.
16h	Word	Boot Connection Vector - Offset relative to the option ROM base address of the boot connection routine. The system BIOS calls this routine if this device has been selected as one of the boot devices.
18h	Word	Boot Disconnection Vector - Offset relative to the option ROM base address of the Boot Disconnection Routine. The system BIOS calls this routine to clean up after an unsuccessful boot to this device.
1Ah	Dword	Bootstrap Entry Point. If this device has been selected as the IPL device but does not have an INT 13h interface (indicated by a flag returned from initialization routine). The system BIOS issues a FAR CALL to this entry point instead of INT 19h. Set this vector to 00000000h if no bootstrap entry point is provided.
1Eh	Word	Static Resource Information Vector - Offset relative to the option ROM base address of the optional get static resource usage routine. A non-Plug and Play ISA device may include a ROM that uses this feature to report the device resource use. This routine must provide the same interface as the system BIOS GetSystemDeviceNode function.

PnP Option ROM Initialization Routine

Plug and Play option ROMs provide an initialization routine for backward compatibility with non-PnP system BIOS. The PnP system BIOS passes control to the initialization routine by issuing a FAR CALL to offset 03h in any valid Option ROM headers that it finds. If the Option ROM contains a valid PnP expansion header, the system BIOS passes the following information through registers to the Option ROM initialization routine:

Register	Value to be Loaded
BX	Card Select Number (CSN) for this card. If the card is not a PnP ISA device, this register is set to 0000h.
DX	Read Data Port. The I/O port address that the system BIOS reads data from the PnP ISA configuration space. If the card is not a PnP ISA device, this register is set to 0000h.
ES:DI	Pointer to the PnP System BIOS Signature structure.

During initialization routines, PnP Option ROMs may hook any vectors except those used for boot devices (INT 09h, INT 10h, and INT 13h). The PnP Option ROM must not modify any information in the BIOS Data Area or Extended BIOS Data Area.

Initialization Routine Output The initialization routine returns the following information in AX:

Bits	Description
15 – 9	Reserved (set to 0)
8	1 This IPL device provides an INT 13h interface.
7	1 This output device provides an INT 10h interface.
6	1 This input device provides an INT 09h interface.
5 – 4	00 No IPL device is attached. 01 Cannot determine if an IPL device is attached. 10 IPL device is attached.
3 – 2	00 No display device is attached. 01 Cannot determine if a display device is attached. 10 Display device is attached.
1 – 0	00 No input device is attached. 01 Cannot determine if an input device is attached. 10 Input device is attached.

Function 50h Get Number of DMI Structures

```
int FAR (*entryPoint)(Function,NumStructures,StructureSize,BiosSelector);
int Function;                                   /*PnP BIOS Function 50h*/
unsigned FAR *NumStructures;                    /*# of structures returned by BIOS*/
unsigned int FAR *StructureSize;                /*Size of largest DMI structure*/
unsigned int BiosSelector;                      /*PnP BIOS readable/writable*/
                                                /*selector*/
```

Returns

```
AH  =   00h Successful
    =   Error code (Bit 7 on)
All flags and other registers are preserved if an error occurs.
```

Description

Parameter	Description
Function	50h
NumStructures	The number of DMI structures that the system BIOS will return information about is returned in this parameter. These structures represent the DMI information that is embedded in the system BIOS.
StructureSize	The system BIOS returns the size (in bytes) of the largest DMI structure and all of its supporting data in this parameter. System software uses this information to set aside the maximum amount of memory needed to contain all DMI structures. The system BIOS may return a value larger than the largest DMI structure to facilitate the storage of hot docking and other dynamic DMI information. AH will contain 82h if the system BIOS does not support DMI.
BiosSelector	This parameter allows the system BIOS to update the system variables in the system BIOS memory.

Address Modes This function can be called from real mode or 16-bit protected mode.

Called from Protected Mode If this function is called from protected mode, you must create a data segment descriptor using:

- the 16-bit protected mode data segment base address specified in the PnP installation check data structure,
- a limit of 64 KB, and
- that the descriptor is read/write capable.

Called from Real Mode If this function is called from real mode, BiosSelector should be set to the Read Mode 16-bit data segment address as specified in the Plug and Play installation check structure.

Cont

Function 50h Get Number of DMI Structures, cont

Function 50h Get Number of DMI Structures Example The following code segment shows how the C-style call interface can be made directly from an assembly language code module:

```
PUSH  BiosSelector
PUSH  segment/selector of StructureSize   ;pointer to
                                          ;StructureSize
PUSH  offset of StructureSize
PUSH  segment/selector of NumStructures   ;pointer to
                                          ; NumStructures
PUSH  offset NumStructures
PUSH  GET_NUM_DMI_STRUCTURES
CALL  FAR PTR entryPoint
ADD   SP,12                               ;clean up stack
CMP   AX,SUCCESS                          ;function successful?
JNE   ERROR
```

Function 51h Get DMI Structure

```
int FAR (*entryPoint)(Function,Structure,dmiStrucBuffer,BiosSelector);
int Function;                             /*PnP BIOS Function 51h*/
unsigned int FAR *Structure;              /*Structure Number/handler*/
                                          /* returned by BIOS*/
unsigned char FAR *dmiStrucBuffer;      /*Pointer to structure data buffer*/
unsigned int BiosSelector;              /*PnP BIOS readable/writable selector*/
```

Returns

```
AH  =   00h Successful
    =   Error code (Bit 7 on)
All flags and other registers are preserved if an error occurs.
```

Description This function copies the information for the specified DMI structures to the buffer that you specified.

Parameter	Description
Function	51h
Structure	This is a pointer to the unique DMI Structure number (handle). If Structure contains zero, the system BIOS returns the first DMI Structure. This parameter is updated to the next structure number on return. If there are no more DMI structures, it will contain FFh.
dmiStrucBuffer	This parameter must contain a pointer to a caller-specified memory buffer.
BiosSelector	This parameter allows the system BIOS to update the system variables in the system BIOS memory.

Cont

Function 51h Get DMI Structure, cont

Address Mode This function can be called from real mode or 16-bit protected mode:

If	you must
this function is called from protected mode,	create a data segment descriptor using: • the 16-bit protected mode data segment base address specified in the PnP installation check data structure, with • a limit of 64 KB, and • the descriptor is read/write capable
this function is called from real mode,	BiosSelector should be set to the Read Mode 16-bit data segment address as specified in the Plug and Play installation check structure.

Function 51h Get DMI Structure Example The following code segment shows how the C-style call interface can be made directly from an assembly language code module:

```
PUSH  BiosSelector
PUSH  segment/selector of dmiStrucBuffer  ;pointer to
                                          ;DMIstructure buffer
PUSH  offset of dmiStrucBuffer
PUSH  segment/selector of Structure       ;pointer to Structure
PUSH  offset Structure
PUSH  GET_DMI_STRUCTURE
CALL  FAR PTR entryPoint
ADD   SP,12                               ;clean up stack
CMP   AX,SUCCESS                          ;function successful?
JNE   ERROR
```

Cont

Function 53h Get DMI Event Information Some computers allow you to add or remove devices while the computer is on. For example, you can insert a notebook computer in a docking station while both units are powered on. A DMI-aware system BIOS provides event notification facilities for system software. System software can use these BIOS functions so that it knows:

- when a device has been added to or removed from the computer, and
- when the DMI BIOS structures have been modified.

Event notification can be implemented as either a polled method or as asynchronous events. When system software is notified of an event by either method, it can then call the BIOS runtime function (Plug and Play BIOS function 3 Get Event) to ascertain the type of event.

Additional PnP Event DMI_EVENT_NOTICE (7FFFh) has been added to the PnP BIOS Specification. This message indicates that DMI information maintained by the system BIOS has changed. When system software receives a DMI_EVENT_NOTICE, it calls the BIOS runtime function 53h Get DMI Event Information to determine the cause of the DMI event.

Structure

```
int FAR (*entryPoint)(Function,dmiEventStructure,BiosSelector);
int Function;                              /*PnP BIOS Function 53h*/
unsigned char FAR *dmiEventStructure;      /*Pointer to DMI event structure*/
unsigned int BiosSelector;                 /*PnP BIOS readable/writable selector*/
```

Returns

```
AH  =   00h Successful
    =   Error code (Bit 7 on)
        All flags and other registers are preserved if an error occurs.
```

Cont

Function 53h Get DMI Event Information, Cont

Description This function provides a mechanism for system software to obtain information about DMI events.

Parameter	Description
Function	53h
dmiEventStructure	Pointer to a caller-defined memory buffer where the DMI event structure will be returned by the system BIOS.
BiosSelector	This parameter allows the system BIOS to update the system variables in the system BIOS memory.

Structure of Memory Buffer

Field	Offset	Length	Value
DMI Event Status	00h	Word	ENUM
DMI Structure	02h	Byte	Varies
Reserved	04h	Bytes	00h

DMI Event Status

Status Code	Description	Activity if returned
0000h	Reserved	
0001h	Other	
0002h	Unknown	
0003h	Single DMI Structure changed.	The number (handle) of the changed DMI structure is placed in DMI Structure.
0004h	Multiple DMI Structures changed.	The application program must enumerate all DMI structures to find all changes.
005h – FFFFh	Reserved	

Cont

Function 53h Get DMI Event Information, Cont

Address Mode This function can be called from real mode or 16-bit protected mode:

If	you must
this function is called from protected mode,	create a data segment descriptor using: • the 16-bit protected mode data segment base address specified in the PnP installation check data structure, with • a limit of 64 KB, and • the descriptor is read/write capable
this function is called from real mode,	BiosSelector should be set to the Read Mode 16-bit data segment address as specified in the Plug and Play installation check structure.

Error If No DMI Event If this function is called and there are no DMI events, error code 86h EVENTS_NOT_PENDING is returned in AH.

Function 53h Get DMI Event Information Example The following code segment shows how the C-style call interface can be made directly from an assembly language code module:

```
PUSH  BiosSelector
PUSH  segment/selector of dmiEventStructure
PUSH  offset of dmiEventStructure
PUSH  GET_DMI_EVENT
CALL  FAR PTR entryPoint
ADD   SP,8                          ;clean up stack
CMP   AX,SUCCESS                    ;function successful?
JNE   ERROR
```

Cont

Function 55h Get General Purpose NVRAM Information NVRAM is often called CMOS RAM, after the method of constructing the semiconductors used for this type of storage. NVRAM consumes very little power. It is used to store system configuration information. NVRAM is managed by the system BIOS, usually through a BIOS Setup utility. This function provides access to system configuration information stored in NVRAM.

Structure

```
int FAR (*entryPoint) (Function, Index, MinGPNVWriteSize, GPNVSize,
NVStorageBase, BiosSelector);
int Function;                       /*PnP BIOS Function 55h*/
unsigned int FAR *Index;            /*Identifies NVRAM area to be accessed*/
unsigned int FAR *MinGPNWriteSize;  /*Minimum NVRAM write buffer size in bytes*/
unsigned int FAR GPNVSize;          /*Size allocated for GPNV in NVS block*/
unsigned int FAR *NVStorageBase;    /*32-bit physical base address for*/
                                    /*memory-mapped NVRAM storage media*/
unsigned int BiosSelector;          /*PnP BIOS readable/writable selector*/
```

Returns

```
AH  =   00h Successful
    =   Error code (Bit 7 on)
All flags and other registers are preserved if an error occurs.
```

Description This function returns system configuration information stored in NVRAM.

Parameter	Description
Function	55h
Index	An index into the NVRAM area. Zero accesses only the first area. On return, this field is updated either with the next GPNV area number or FFFFh if there are no more areas in NVRAM.
MinGPNVWriteSize	The BIOS returns the minimum size (in bytes) of a caller-specified buffer where the NVRAM contents will be written in this field.
GPNVSize	The BIOS returns the overall size (in bytes) of the NVRAM in this field. The size of the non-volatile storage that contains the GPNV must not exceed 32 KB.
NVStorageBase	The BIOS returns the 32-bit absolute physical base address of the NVRAM area. From this address, you can construct a 16-bit data segment descriptor with a limit of 64 KB and read/write access. This descriptor allows the BIOS to read and write the memory-mapped NVRAM area to a protected mode environment. If the BIOS returns zero in this field, protected mode mapping is not required.
BiosSelector	Allows the system BIOS to update the system variables in the system BIOS memory.

Cont

Function 56h Read General Purpose NVRAM Data This function reads the entire NVRAM storage area to the buffer specified in the GPNVBuffer field. Make sure that the buffer is large enough to store the entire GPNV. Use the value returned by Function 55h in GPNVSize.

Structure

```
int FAR (*entryPoint) (Function,Index,GPNVBuffer,GPNVLock,GPNVSelector, BiosSelector);
int Function;                  /*PnP BIOS Function 56h*/
unsigned int *Index;           /*Identifies NVRAM area to be accessed*/
unsigned int FAR *GPNVBuffer;  /*Address of buffer where NVRAM is returned*/
unsigned int FAR GPNVLock;     /*Lock value*/
unsigned int FAR *GPNVSelector; /*Readable/Writable selector*/
unsigned int BiosSelector;     /*PnP BIOS readable/writable selector*/
```

Returns

AH = 00h Successful
= Error code (Bit 7 on)
All flags and other registers are preserved if an error occurs.

Parameter	Description
Function	56h
Index	Index to NVRAM. A value of zero accesses only the first area. On return, this field is updated either with the next GPNV area number or FFFFh if there are no more areas in NVRAM.
GPNVBuffer	The BIOS returns the current contents of the GPNV area to the buffer specified in this field.
GPNVLock	Contains a simple locking mechanism for cooperative use of the GPNV. Enter a zero to not lock the GPNV. You must provide a unique value (process ID) for this field. You must cooperate with other programs that access the GPNV based on the value of this call. The BIOS does not enforce mutually-exclusive access to the GPNV based on the value of this field. To lock the specified GPNV area, enter a non-zero lock value in this field. If the value is unmodified on return, the specified GPNV is locked. If the value in this field is modified on return, the GPNV has been locked by a different program and the GPNVLock value is set to the 1s complement of the input value. After the GPNV has been successfully locked, you must subsequently unlock the GPNV by issuing function 57h Write GPNV Data with the same *GPNVLock* value specified when issuing function 55h.
GPNVSelector	This is the protected mode selector. Its base is equal to NVStorageBase with a limit equal to or greater than the value returned by Function 55h in the GPNVSize field (assuming that the value returned in NVStorageBase was not zero).
BiosSelector	This parameter allows the system BIOS to update the system variables in the system BIOS memory.

Cont

Function 57h Write General Purpose NVRAM Data This function writes the entire NVRAM from the buffer that you constructed specified in *GPNVBuffer* to the specified NVRAM.

- first issue function 56h Read General Purpose NVRAM with a lock to pass the current NVRAM contents to the buffer specified in GPNVBuffer in function 56h,
- modify the NVRAM data, and
- pass the data back to NVRAM by issuing function 57h.

Structure

```
int FAR (*entryPoint) (Function, Index, GPNVBuffer, GPNVLock, GPNVSelector,
BiosSelector);
int Function;                   /*PnP BIOS Function 57h*/
unsigned int *Index;            /*Identifies NVRAM area to be accessed*/
unsigned int FAR *GPNVBuffer;   /*Address of buffer where NVRAM is stored*/
unsigned int FAR GPNVLock;      /*Lock value*/
unsigned int FAR *GPNVSelector; /*Readable/Writable selector*/
unsigned int BiosSelector;      /*PnP BIOS readable/writable selector*/
```

Returns

```
AH  =   00h Successful
        =   Error code (Bit 7 on)
All flags and other registers are preserved if an error occurs.
```

Description

Parameter	Description
Function	57h
Index	This is an index into the NVRAM area. A value of zero accesses only the first area. On return, this field is updated either with the next GPNV area number or FFFFh if there are no more areas in NVRAM.
GPNVBuffer	The contents of the buffer pointed to in this field are written to NVRAM.
GPNVLock	The non-zero GPNVLock value in this field must be the same as the value specified in this field when function 56h was issued. If a non-zero value is returned in this field, the operation has failed. The contents of the buffer pointed to by GPNVBuffer were not written to GPNVRAM. This field is cleared of all previous locks if this function is successful.
GPNVSelector	This is the protected mode selector. Its base is equal to NVStorageBase with a limit equal to or greater than the value returned by Function 55h in the GPNVSize field (assuming that the value returned in NVStorageBase was not zero).
BiosSelector	This parameter allows the system BIOS to update the system variables in the system BIOS memory.

Cont

Function 57h Write General Purpose NVRAM Data, Cont

Address Mode This function can be called from real mode or 16-bit protected mode:

If	you must
this function is called from protected mode,	create a data segment descriptor using: • the 16-bit protected mode data segment base address specified in the PnP installation check data structure, with • a limit of 64 KB, and • the descriptor is read/write capable
this function is called from real mode,	BiosSelector should be set to the Read Mode 16-bit data segment address as specified in the Plug and Play installation check structure.

Function 57h Write General Purpose NVRAM Data Example The following code segment shows how the C-style call interface can be made directly from an assembly language code module:

```
PUSH  BiosSelector
PUSH  segment/selector of dmiStrucBuffer  ;pointer to
                                          ;DMIstructure buffer
PUSH  offset of dmiStrucBuffer
PUSH  segment/selector of Structure       ;pointer to Structure
PUSH  offset Structure
PUSH  GET_DMI_STRUCTURE
CALL  FAR PTR entryPoint
ADD   SP,12                               ;clean up stack
CMP   AX,SUCCESS                          ;function successful?
JNE   ERROR
```

INT 1Bh <Ctrl> <Break>

Input: None

Output: None

Description INT 1Bh is called by the operating system to terminate the current application when you press <Ctrl> <Break>. The BIOS sets this routine to an IRET instruction. The next time the operating system boots, it resets the routine to point to its own interrupt service routine.

INT 1Ch Periodic Timer Interrupt

Input: None

Output: None

Description The system timer calls INT 08h 18.2 times per second. After each call to INT 08h, INT 1Ch is called to permit access to the system timer by any applications program. The BIOS sets this routine to an IRET instruction. The next time the operating system boots, it resets the routine to point to its own interrupt service routine.

INT 1Dh Video Parameter Table

Input: None

Output: None

Description The vector for INT 1Dh points to a table of video parameters.

INT 1Eh Floppy Disk Parameter Table

Input: None

Output: None

Description The vector for INT 1Eh points to a table of floppy disk parameters.

INT 1Fh Video Graphics Characters

Input: None

Output: None

Description The vector for INT 1Fh points to a table of video graphics characters.

INT 4Ah Alarm ISR

When the alarm is activated, the Real Time Clock generates an interrupt request at the time specified in INT 1Ah Function 06h. INT 4Ah can be invoked when the alarm occurs. The program that issued INT 1Ah Function 06h must redirect the INT 4Ah vector (0:128h) to a routine that processes the alarm.

INTs 70h through 77h

An ISA system has two interrupt controllers. The second controller calls INTs 70h to 77h. Only INTs 70h, 74h, 75h, and 76h are described in this book. The programmer cannot revector any of the interrupts from INT 70h – 77h to his own routine.

INT 70h Real Time Clock Interrupt (IRQ8)

Input: None

Output: None

Description AMIBIOS services INT 70h by determining the reason the interrupt was called and correcting the situation that caused INT 70h. INT 70h ticks about 1024 times per second.

INT 71h IRQ9

Input: None

Output: None

Description When IRQ9 occurs, the interrupt is routed through the IRQ2 transfer vector (INT 0Ah) by the BIOS and the slave interrupt controller's interrupt is cleared so the interrupt appears to be an IRQ2.

INT 74h PS/2 Mouse Interrupt (IRQ12)

Input: None

Output: None

Description INT 74h is the interrupt service routine for BIOS PS/2-type mouse support. The PS/2 mouse sends data to the keyboard controller. The keyboard controller generates IRQ12. Mouse data is transmitted in packets. The BIOS INT 74h collects these packets and stores them in the extended BIOS data area. INT 74h also sets the appropriate flags.

INT 75h Math Coprocessor Interrupt (IRQ13)

Input: None

Output: None

Description INT 75h is called when the math coprocessor in the computer generates an exception and the exception interrupt has been enabled. This interrupt is passed on to the BIOS INT 02h NMI processing routine.

INT 76h AT Hard Disk Drive Interrupt (IRQ14)

Input: None

Output: None

Description The hard disk drive controller calls INT 76h when a hard disk drive access is completed.

INT 77h Software Suspend Request (IRQ15)

Input: None

Output: None

Description: Some American Megatrends Power Management BIOSes process an INT 77h as a suspend request. The BIOS powers down the computer when it receives an INT 77h. Any applications software program can issue an INT 77h to power down the computer if the computer has one of these Power Management BIOSes.