AMIBIOS 98

Technical Reference

MAN-BIOS98-TR
5/1/98

American Megatrends, Inc.
6145F North Belt Parkway
Norcross, GA 30071

Revision History

4/92 Initial release.
10/92 Revised for 6/6/92 core AMIBIOS.
12/92 Corrected minor errors.
5/24/93 Added APM, PCI, and Socket Service BIOS function documentation.
8/1/93 Updated to PCMCIA Version 2.00 Socket Service specifications.
9/1/93 Added new INT 16h functions for Flash EPROM.
4/1/94 Added PnP functions and new POST checkpoint codes.
10/1/94 Updated for 07/25/94 core BIOS.
10/31/95 Updated for AMIBIOS 95.
5/6/96 Updated for V6.24 of 7/15/95 core AMIBIOS.
6/19/97 Converted to Microsoft Word Format.
7/9/97 Added additional PCI functions.
2/25/98 Corrected error in PCI function register use and INT 15h Function E801h use.
5/1/98 Added INT 15h Function 24h description.

Table of Contents

Preface

To the Reader This manual provides technical details about the operation of AMIBIOS for ISA and EISA PCI, Plug and Play, APM-aware, and DMI-aware computers. We assume are familiar with the Intel x86 architecture, x86 assembler language, and both ISA and EISA system architecture.

Technical Support AMI only provides technical support for AMI products purchased directly from AMI or from an AMI-authorized reseller.

If…	then…
You purchased this product from AMI or from a certified AMI reseller,	Call AMI technical support at 770-246-8600. Please be prepared to specify the serial number of the product.
This AMI product was installed as part of a system manufactured by a company other than AMI or you purchased an AMI product from an unauthorized reseller.	Call the technical support department of the computer manufacturer or the unauthorized reseller. AMI does not provide direct technical support in this case.

Web Site You can download technical and marketing information from our web site. The URL is:

`http://www.ami.com`

Technical Support Email You can send email to the American Megatrends technical support department at the following address:

`Support@ami.com`

1 Introduction

The architecture of ISA and EISA computers is layered. The lowest layer is the computer — the hardware itself. The highest layer is the applications software that the user interfaces with. Systems software lies between applications software and hardware.

Systems software can consist of several elements: the operating system kernel, the operating system shell, and additional device drivers. Operating environments (Microsoft® Windows™, for example) exist in a layer between the operating system and applications software, as do multitasking supervisors or DOS extenders like Desqview®.

The BIOS (Basic Input/Output System) is a collection of routines between the hardware and the systems software. The BIOS ROM contains hard disk utilities, device drivers, interrupt service routines, and other code and data between the system hardware and the systems software.

New BIOS Features and Functions In the last year, system BIOS has taken on an additional new layer of functionality. The EPA Green PC, PCI (Peripheral Component Interconnect), Plug and Play, APM (Advanced Power Management), and DMI (Desktop Management Interface) initiatives have all specified many additional functions that must be performed by the system BIOS in ISA and EISA computers. This manual has been updated to include all BIOS functions required by these new specifications and implemented in AMIBIOS.

In This Chapter The following topics are discussed in this chapter:

- BIOS as Interface,
- Parts of the ROM BIOS, and
- Types of BIOS.

The BIOS as Interface

The BIOS is the software layer between the systems software and the hardware in ISA and EISA systems. The BIOS works in two directions. One part of the BIOS receives and processes requests from programs to perform the standard BIOS I/O services. The mechanism for these requests is called an interrupt, discussed in detail later in this manual. Interrupts are invoked by software programs. In an assembler program, the INT mnemonic is followed by an interrupt number that specifies the type of service and a function number that specifies the exact service to be performed. For example:

```
MOV     AH,00h  ;specifies function 00h get ;character from keyboard
INT     16h  ;requests INT 16h Keyboard Service
```

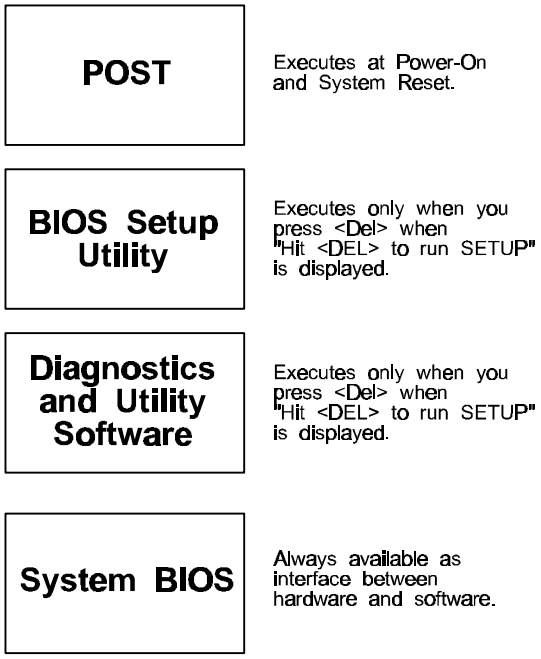
BIOS and the Hardware The other side of the BIOS communicates with the hardware (video display, disk drives, keyboard, serial and parallel ports, and so on) in the language and codes used by each device. This side of the BIOS also handles any hardware device-generated interrupts. For example, when a key is pressed on the keyboard, a hardware interrupt (IRQ1) is generated. The BIOS INT 09h interrupt service routine is called to handle this interrupt. The following figure illustrates the role that the system BIOS plays.

Parts of the System ROM BIOS

The system ROM elements in a computer with an AMIBIOS are:

- POST,
- the ROM BIOS itself, and
- the BIOS Setup utility.

The following graphic depicts these elements:



System BIOS The BIOS is a part of the code stored in ROM that is in active use the entire time that a computer is on. The ROM BIOS provides the fundamental services needed for the proper operation.

Cont'd

POST The ROM BIOS includes BIOS Power-On Self Test diagnostic and booting code that tests the computer components, initializes certain data structures, and boots DOS. POST performs several functions:

- executes a diagnostic and reliability test of the computer, the ROM programs, and RAM,
- initializes the chips and the standard parts of the computer and places a record of the system configuration in CMOS RAM and in low system memory,
- sets up the interrupt vector table,
- detects optional equipment, and
- boots the operating system.

See the BIOS POST discussion in Chapter 5 for additional information about POST.

AMIBIOS Setup Utility AMIBIOS Setup stores system configuration values in CMOS RAM.

Type of Setup	Description
Standard Setup	Sets the hard disk drive type, type of floppy drives and monitor, and the day, date, and time.
Advanced Setup	Sets more technical system values, such as the typematic rate, memory test patterns, cache setting, etc.
Chipset Setup	Sets system configuration data that is specific to the chipset-used on the motherboard. The Chipset Setup options are highly technical. The settings for Chipset Setup options are usually specific to the motherboard design and should not be changed.
PCI/PnP Setup	Sets system configuration data that relates to the use of the PCI bus and the Plug and Play configuration.
Power Management Setup	Power Management Setup configures options related to power conservation and use.
Peripheral Setup	Configures options such as the floppy and IDE controllers on the motherboard, the base I/O port addresses for the serial and parallel ports, and other options related to the I/O controller.

WINBIOS® Setup American Megatrends introduced the WINBIOS Setup utility in early 1994. It has been an instant hit because of its extremely easy-to-use Microsoft Windows-like graphical interface.

Types of BIOS

In ISA and EISA computers, the types of BIOS (Basic Input Output System) include:

- the system BIOS,
- the video BIOS, and
- optional option ROM BIOSes.

This manual describes the system BIOS features and functions. The video BIOS is best discussed in the context of EGA®, VGA®, and XGA® video standards, which all require a separate BIOS.

System BIOS

The system BIOS tests the computer components, loads (bootstraps) the operating system, and remains active for requests by the operating system to activate device drivers that service the hardware components. The BIOS takes the instructions from the operating system and translates these commands to the exact instructions that the hardware itself understands. The BIOS maintains data about the components. When a component is unable to perform, the BIOS reports it to the operating system.

An AMIBIOS system ROM BIOS with all features enabled is usually 128 KB long and resides at E0000h – FFFFFh in both EISA and ISA computers. This area is addressed to ROM but can be shadowed to RAM. ROM operates at about 120 – 180 ns; RAM usually operates at 60 – 100 ns. System BIOS are almost universally stored on flash EPROMs now so they can be upgraded easily.

The system BIOS now typically includes PCI, APM, Plug and Play, DMI, enhanced IDE, WINBIOS Setup utility, and Green PC support. Newer AMIBIOS support the Extended System Configuration Data Specification (ESCD).

Cont’d

Video BIOS All ISA and EISA computers that use EGA, VGA, or XGA have video BIOSes. The system BIOS video service only handles the most basic video functions. Advanced video modes must be translated via a video BIOS, usually installed on the video adapter card.

Option ROM BIOS Many adapter cards have code in ROM. For example, Ethernet network adapter cards have a ROM that assists in translating Ethernet commands to code the computer can understand and vice versa. Option ROM resides between C8000h and EFFFFh. This area also can be shadowed in AMIBIOS to speed the operation of the devices that have option ROMs, provided that the motherboard or chipset supports option ROM shadowing. Option ROM is also called Option ROM.

Specifications Supported In general, AMIBIOS supports the latest specification for each of the following standards:

- Plug and Play,
- DMI,
- Extended System Configuration Data,
- EISA,
- PCI,
- ACPI,
- AGP,
- SMART,
- IEEE 1394 Firewire,
- Ultra DMA,
- all Enhanced IDE, ATA and ATAPI standards,
- IrDA,
- MPS,
- Universal Serial Bus,
- APM (Advanced Power Management), and
- EPA Green PC.

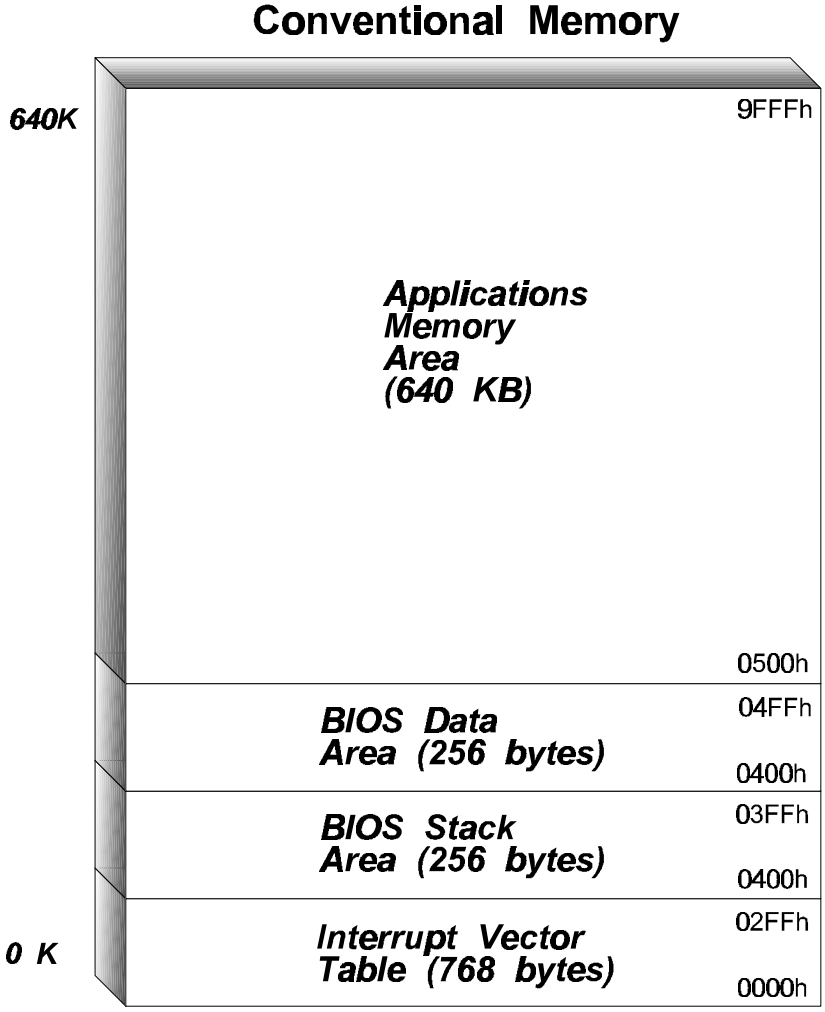
AMIBIOS Utilities

Utility	Audience	Description
AMIBIOS Setup	end users	This character-based utility is provided with Hi-Flex AMIBIOS. It configures the computer and manages the NVRAM (Non-Volatile Random Access Memory) in the computer. See the *Hi-Flex AMIBIOS User's Guide* for additional information.
WINBIOS Setup	end users	This graphically-oriented utility is provided with AMIBIOS. It configures the computer and manages the NVRAM (Non-Volatile Random Access Memory) in the computer. See the *AMIBIOS User's Guide for the 10/10/94 (or later) core AMIBIOS* for additional information.
System Configuration Utility	end users	The SCU configures both EISA and Plug and Play computers. See the *System Configuration Utility User's Guide* for additional information.
AMIBCP	BIOS customers	The BIOS Configuration Program allows computer manufacturers to customize AMIBIOS.
AMIEmbed	BIOS customers	Allows OEMs to embed option ROMs in AMIBIOS. Typical applications include adding SCSI BIOS, network BIOS, or VGA BIOS to AMIBIOS.
PCI Router	BIOS customers	Allows OEMs to customize the IRQ routing for PCI-based motherboards. PCI provides four hardware interrupts that can be used in many ways. Of necessity, motherboard IRQ routing must be different in each motherboard design. PCI Route allows motherboard designers to customize the IRQ routing in a standard AMIBIOS for a specific motherboard
AMIFlash	BIOS customers	AMIFlash allows OEMs and end users to update the BIOS in a computer where the system BIOS is stored on a flash EPROM.
DMI Wizard DMI Wizard 95	BIOS customers	DMI Wizard is a DOS utility with a graphical interface that allows you to display and modify DMI (Desktop Management Interface) system configuration data in any DMI-aware system BIOS ROM or BIOS .ROM file. DMI Wizard 95 is a Windows 95 utility that performs the same functions.
CPUSelect	BIOS customers	Allows OEMs to customize CPU support in an AMIBIOS.

2 System Memory Map

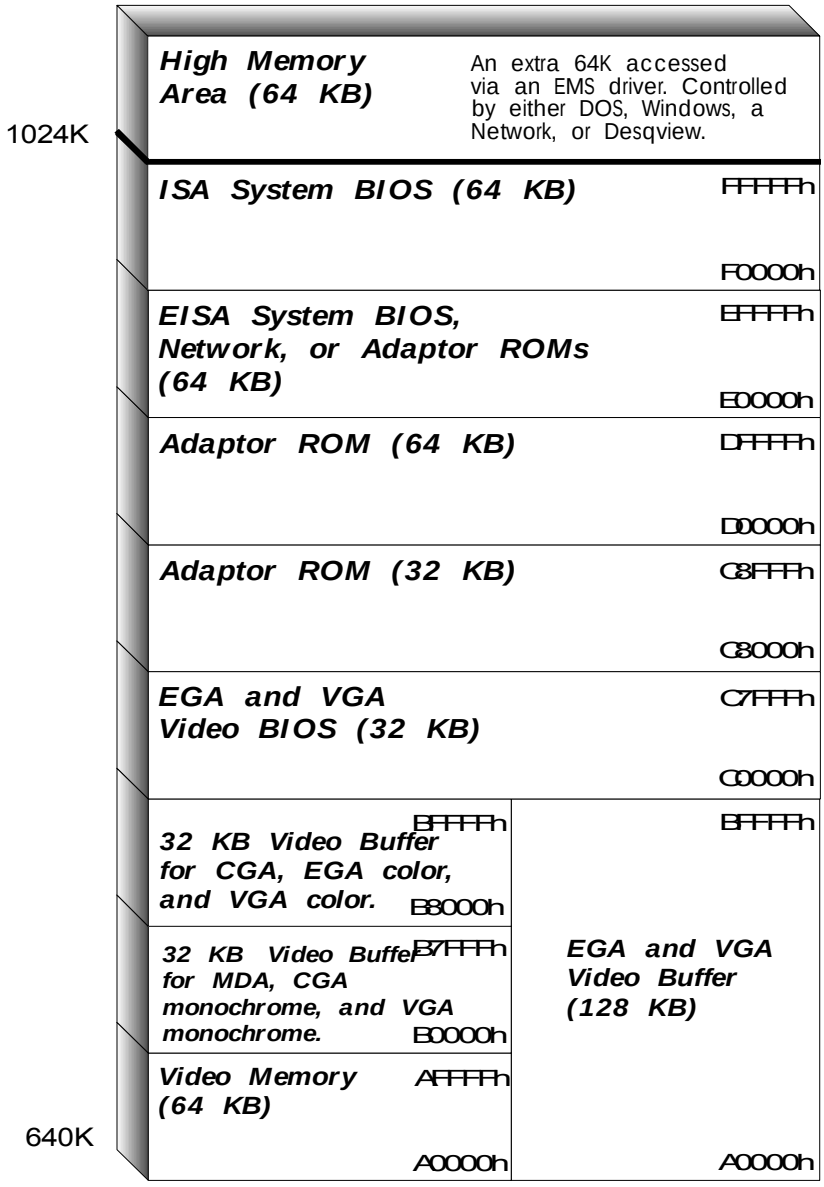
Conventional Memory Map

The following table shows the use of the first megabyte of memory.



Upper Memory Map

Upper Memory Blocks



3 BIOS Data Area

When an ISA or EISA computer is powered-on, the BIOS Data Area is created at location 000400h. It is 256 bytes long (000400 – 0004FFh) and contains information about the system environment. This information can be accessed and changed by any program. Much of the operation of ISA and EISA computers is controlled by this data. This data is loaded by POST (the BIOS Power-On Self Test) during startup. The following table lists the contents of all BIOS data area locations. All addresses are offsets from 000400h.

Offset	BIOS Service	Description
00h	INT 14h	Serial Port (COM) 1 — least significant byte.
01h	INT 14h	Serial Port (COM) 1 — most significant byte.
02h	INT 14h	Serial Port (COM) 2 — least significant byte.
03h	INT 14h	Serial Port (COM) 2 — most significant byte.
04h	INT 14h	Serial Port (COM) 3 — least significant byte.
05h	INT 14h	Serial Port (COM) 3 — most significant byte.
06h	INT 14h	Serial Port (COM) 4 — least significant byte.
07h	INT 14h	Serial Port (COM) 4 — most significant byte.
08h	INT 17h	Parallel Port (LPT) 1 — least significant byte
09h	INT 17h	Parallel Port (LPT) 1 — most significant byte
0Ah	INT 17h	Parallel Port (LPT) 2 — least significant byte
0Bh	INT 17h	Parallel Port (LPT) 2 — most significant byte
0Ch	INT 17h	Parallel Port (LPT) 3 — least significant byte
0Dh	INT 17h	Parallel Port (LPT) 3 — most significant byte
0Eh	POST	Extended BIOS data area segment — least significant byte
0Fh	POST	Extended BIOS data area segment — most significant byte
10h–11h	INT 11h	Equipment list Bits 15–14 Number of parallel adapters 00 None installed 01 One installed 10 Two installed 11 Three installed Bit 12 Game port installed 0 No 1 Yes Bits 11–9 Serial adapters installed 000 None installed 001 One installed 010 Two installed 011 Three installed 100 Four installed Bits 7–6 Number of floppy disk drives. 00 One drive 01 Two drives Bits 5–4 Initial video mode 00 EGA/VGA 01 40 x 25 CGA color 10 80 x 25 CGA color 11 80x25 Monochrome Bit 2 PS/2-type pointing device 1 Present Bit 1 Math coprocessor 1 Present Bit 0 Floppy disk drive A: 1 Present
12h	POST	Interrupt flag used in POST.
13h	INT 12h	Memory size in KB — least significant byte.
14h	INT 12h	Memory size in KB — most significant byte.

Offset	BIOS Service	Description
15h–16h		Reserved
17h	INT 16h	Keyboard status byte Bit 7 Insert Mode is on if set. Bit 6 <Caps Lock> Key on if set. Bit 5 <Num Lock> key on if set. Bit 4 <Scroll Lock> Key on if set. Bit 3 <Alt> key pressed if set. Bit 2 <Ctrl> key pressed if set. Bit 1 Left <Shift> Key pressed if set. Bit 0 Right <Shift> Key pressed if set.
18h	INT 16h	Extended keyboard status byte Bit 7 <Ins> key pressed if set. Bit 6 <Caps Lock> key pressed if set. Bit 5 <Num Lock> key pressed if set. Bit 4 <Scroll Lock> pressed if set. Bit 3 <Ctrl><Num Lock> state active Bit 2 <SysReq> key pressed if set. Bit 1 Left <Alt> key pressed if set. Bit 0 Left <Ctrl> key pressed if set.
19h		Reserved
1Ah–1Bh	INT 16h	Pointer to the address of the next character in the keyboard buffer.
1Ch–1Dh	INT 16h	Pointer to the address of the last character in the keyboard buffer.
1Eh–3Dh	INT 16h	Keyboard buffer (32 bytes). If the address in 1Ah is the same as the address in 1Ch, the buffer is empty. If the address in 1Ch is two bytes from the address in 1Ah, the buffer is full.
3Eh	INT 13h	Floppy disk drive calibration status Bit 7 Diskette hardware interrupt occurred 0 No 1 Yes Bits 6–4 Reserved. Should be 00h. Bit 1 Floppy Drive B: needs recalibration if 0. Bit 0 Floppy Drive A: needs recalibration if 0.
3Fh	INT 13h	Floppy disk drive motor status Bit 7 0 Current operation is Write or Format. 1 Current operation is Read or Verify. Bits 5–4 Drive select 00 Drive A: select 01 Drive B: select Bits 3–2 Reserved Bit 1 1 Drive A: motor is on. Bit 0 1 Drive B: motor is on.
40h	INT 13h	Floppy disk drive motor timeout This is decremented by one 18.2 times per second (via the INT 08h timer interrupt). When the value becomes zero, the drive motor is powered off. The value refers to the last disk drive accessed.

Offset	BIOS Service	Description
41h	INT 13h	Floppy disk drive status. These values are valid for the last floppy disk drive accessed. Bit 7 Drive not ready if set. Bit 6 Seek error detected if set. Bit 5 Floppy disk controller failed if set. Bits 4–0 Error codes 00h No error occurred 01h Illegal function requested 02h Address mark not found 03h Write protect error 04h Sector not found 06h Drive door was opened 08h DMA overrun error 09h DMA boundary error 0Ch Unknown media type 10h CRC failed on floppy read 20h Controller failure 40h Seek failed 80h Timeout
42h–48h	INT 13h	Floppy disk controller status bytes and command bytes for the hard disk controller.
49h	INT 10h	Current video display mode setting.
4Ah–4Bh	INT 10h	Number of text columns per line of current video mode.
4Ch–4Dh	INT 10h	Current page size in bytes.
4Eh–4Fh	INT 10h	Offset address of current display page, relative to the start of video RAM. Video RAM starts at B800h in CGA and B000h in MDA.
50h–5Fh	INT 10h	Current cursor position for each video page. Up to eight display pages are possible. Two bytes per page store the current cursor position for each page. The MSB specifies the row (line) value and the LSB specifies the column value of the cursor. Change the cursor position using INT 10h functions. *Do not change the values in this location.*
60h	INT 10h	Starting line of the cursor.
61h	INT 10h	Ending line of the cursor.
62h	INT 10h	Current video display page number.
63h–64h	INT 10h	I/O port address of the video display adapter (the CRT controller address register). 3B4h for monochrome adapter. 3D4h for color.
65h	INT 10h	Video display adapter mode register. The mode register is I/O port 3B8h for monochrome. 3D8h for CGA. 3D9h for EGA/VGA.
66h	INT 10h	Current palette color.
67h–6Bh		Option ROM address.
6Ch–6Fh	INT 1Ah	Counter for INT 1Ah. Incremented by one every time INT 08h occurs (18.2 times per second). Reset to 0 every 24 hours.
70h	INT 1Ah	Timer 24-hour flag. 0 if the timer is between 0 and 24 hours. Set to 1 when the time crosses 24 hours. Must be manually reset.
71h	INT 16h	Break Key pressed flag. Bit 7 1 <Ctrl><Break> or <Ctrl><C> pressed.
72h–73h	POST	Soft reset flag. If 1234h, memory test is skipped on reboot.

Offset	BIOS Service	Description
74h–77h	INT 13h	Status of last hard disk drive operation. 00h No error 01h Invalid function request 02h Address mark not found 04h Sector not found 05h Reset failed 07h Drive parameter activity failed 08h DMA overrun on operation 09h Data boundary error 0Ah Bad sector flag selected 0Bh Bad track detected 0Dh Invalid number of sectors on format 0Eh Control data address mark detected 0Fh DMA arbitration level out of range 10h Uncorrectable ECC or CRC error 11h ECC corrected data error 20h General controller failure 40h Seek operation failed 80h Timeout AAh Drive not ready BBh Undefined error occurred CCh Write fault on selected drive E0h Status error. Error register is 0 FFh Sense operation failed
75h	13h	Number of hard disk drives
76h–77h	13h	Hard disk drive work area
78h	INT 17h	Parallel port 1 timeout counter
79h	INT 17h	Parallel port 2 timeout counter
7Ah	INT 17h	Parallel port 3 timeout counter
7Bh		Reserved
7Ch	INT 14h	Serial port 1 timeout counter
7Dh	INT 14h	Serial port 2 timeout counter
7Eh	INT 14h	Serial port 3 timeout counter
7Fh	INT 14h	Serial port 4 timeout counter
80h–81h	INT 16h	Starting address of the keyboard buffer (usually 01Eh).
82h–83h	INT 16h	Ending address of the keyboard buffer (usually 03Eh).
84h	INT 10h	Number of displayed character rows minus one.
85h-86h	INT 10h	Height of character matrix.

Offset	BIOS Service	Description
87h	INT 10h	Bit 7 Equal to bit 7 of the video mode number passed to INT 10h by the program. Bits 6–4 Video RAM 000 64K 001 128K 010 192K 011 256K 100 512K 101 1024K 110 2048 KB 111 4096 KB Bit 3 0 Video subsystem active 1 Video subsystem inactive Bit 1 0 Color monitor 1 Monochrome monitor Bit 0 0 Alphanumeric cursor emulation disabled 1 Alphanumeric cursor emulation enabled
88h	INT 13h	Data transmission speed of the hard disk drive.
89h	INT 10h	VGA Video Flags Bits 7, 4 Number of lines in monochrome modes 00 350-line mode 01 400-line mode 10 200-line mode Bit 6 0 Display switch disabled 1 Display switch enabled Bit 3 1 Default palette loading enabled 0 Default palette loading disabled Bit 2 0 Color monitor 1 Monochrome monitor Bit 1 0 Gray scale summing disabled 1 Gray scale summing enabled Bit 0 0 VGA inactive 1 VGA active
8Ah–8Bh		Reserved
8Ch–95h	INT 13h	Hard disk and floppy disk drive variables.
96h	INT 16h	Extended Keyboard Status Bit 7 Read ID in progress if set. Bit 6 Last code was first ID if set. Bit 5 Forced Num Lock if set. Bit 4 101 and 102-key keyboard used if set Bit 3 Right <Alt> key active Bit 2 Right <Ctrl> key active. Bit 1 Last code was E0h. Bit 0 Last code was E1h.
97h	INT 16h	Extended Keyboard Status Bit 7 Keyboard error occurred if set. Bit 6 LED is being updated if set. Bit 5 Resend code received if set. Bit 4 Acknowledge code received if set. Bit 2 Caps Lock LED is on if set. Bit 1 Num Lock LED is on if set. Bit 0 Scroll Lock LED is on if set.
98h–99h		Segment part of user wait flag address
9Ah–9Bh		Offset part of user wait flag address

Offset	BIOS Service	Description
9Ch–9Fh		Wait count
A0h	INT 1Ah	Wait active flag Bit 7 Wait time has elapsed if set. Bit 0 INT 15h AH = 86h occurred if set.
A1h–A7h		Reserved
A8h–ABh	INT 10h	INT 10h pointer to EGA and VGA parameter control block
ACh–EFh		Reserved
F0h–FFh		Intra-Applications Communication Area. Stores data used by applications programs.

4 BIOS Data

AMIBIOS includes device parameters that help it to initialize the computer. This information is stored in BIOS arrays and tables described in this chapter:

- APM Information,
- Location of AMIBIOS routines,
- AMIBIOS compatibility information,
- Floppy Disk Drive Parameter Table,
- Hard Disk Drive Parameter Table,
- MFM Hard Drive Types,
- Hard Drive Data Transfer Rates,
- Video Parameter Table,
- System Configuration Data Table,
- Data Transmission Rate Initialization Table, and
- Compatibility Information Tables.

APM Information

Address	Description
F000:F104h	APM Connection Information Bit 7 Reserved Bit 6 0 APM BIOS power management enabled 1 APM BIOS power management disabled Bit 5 0 APM CPU idle call stops CPU clock 1 APM CPU idle call slows CPU clock Bit 4 0 APM BIOS active version 1.0 1 APM BIOS active version above 1.0 Bit 2 1 APM 32-bit protected mode connection established. Bit 1 1 APM 16-bit protected mode connection established. Bit 0 1 APM real mode connection established.
F000:F105h	APM State Information Bit 7 1 Power management enabled by APM Bit 6 1 Cooperative power management between the APM BIOS and an APM driver is enabled for this device (V 1.1 only). Bit 5 1 APM Function Number 0Ch is currently enabled for this device (V 1.1 only). Bit 4 Reserved Bit 3 1 APM controlled device (V 1.1 only). Bits 2-0 APM device state 000 APM enabled state 001 APM standby state 010 APM suspend state 011 APM off state 111 APM on state

Address	Description
F000:F106h	Pending APM Event Information Bits15-11 Reserved Bit 10 1 System standby resume notification (V 1.1 only). Bit 9 1 User system suspend request notification (V 1.1 only). Bit 8 1 User system standby request notification (V 1.1 only). Bit 7 1 Critical system suspend notification (V 1.1 only). Bit 6 1 Update time notification (V 1.1.only). Bit 5 1 Power status change notification (V 1.1 only). Bit 4 1 Battery low notification Bit 3 1 Critical resume system notification Bit 2 1 Normal resume system notification Bit 1 1 System suspend request notification Bit 0 1 System standby request notification
F000:FEA8h	Pointer to Bootstrap CPU information array
F000:FEACh	Pointer to FPU information array
F000:FEB0h	Pointer to Application CPU information array

F000:EEC0h - F000:EF56h is used for routine absolute addresses to provide a common interface to commonly-used system BIOS routines. These routines can even be called by a segment different from F000h without duplicating the routines.

Location of AMIBIOS Routines

Address	Description
F000:EEC0h	Uncompress routine Input: AX:BX Source compressed code segment:offset address. CX:DX Destination segment:offset address where the code is to be uncompressed. Output: None Registers: None affected Use: Must be invoked as CALL FAR
F000:EEC3h	System BIOS FIXED_DELAY routine. Must be invoked as CALL FAR.
F000:EEC6h	Get_Processor_Info routine. Must be invoked as CALL FAR.
F000:EEC9h	Get_Reset_ID routine. Must be invoked as CALL FAR.
F000:EECCh	Get_CPU_ID routine. Must be invoked as CALL FAR.
F000:EECFh	Get_Vendor_Name routine. Must be invoked as CALL FAR.
F000:EED2h	BIOS INT-13 Hard Disk Handler entry point. Must be invoked as INT.
F000:EED5h	Check_CMOS_Data routine. Must be invoked as CALL FAR.
F000:EED8h	Uncompress_General_Module routine. Must be invoked as CALL FAR.
F000:EEDBh	Uncompress_PCI_Module routine. Must be invoked as CALL FAR.
F000:EEDEh - F000:EEE3h	Reserved for internal use.
F000:EEE4h - F000:EE4Ch	Reserved for future use
F000:EF4Dh	APM 16-bit protected mode entry point. Must be invoked as CALL FAR.
F000:EF50h	APM 32-bit protected mode entry point. Must be invoked as CALL FAR.

ROM Compatibility Table

Both ISA and EISA system BIOS assure compatibility with older standards via the following vectors:

Address	BIOS Service Table
FE05Bh	POST entry point
FE2C3h	NMI Handler entry point
FE3Feh	INT 13h Hard Disk Drive Service entry point
FE401h	Hard Disk Drive Parameter Table
FE6F2h	INT 19h Boot Load Service entry point
FE6F5h	Configuration Data Table
FE729h	Data Transmission Rate Generator Table
FE739h	INT 14h Serial Communications Service entry point
FE82Eh	INT 16h Keyboard Service entry point
FE987h	INT 09h Keyboard Service entry point
FEC59h	INT 13h Floppy Disk Service entry point
FEF57h	INT 0Eh Floppy Disk Hardware Interrupt Service Routine entry point
FEFC7h	Floppy Disk Controller Parameter Table
FEFD2h	INT 17h Parallel Printer Service entry point
FF045h	INT 10h Video Service Functions 00h through 0Fh entry point
FF065h	INT 10h Video Service entry point
FF0A4h	MDA and CGA Video Parameter Table (INT 1Dh)
FF841h	INT 12h Memory Size Service entry point
FF84Dh	INT 11h Equipment List Service entry point
FF859h	INT 15h Systems Services entry point
FFA6Eh	Low-order 128 characters of the 320 x 200 and 640 x 200 graphics fonts
FFE6Eh	INT 1Ah Time-of-Day Service entry point
FFEA5h	INT 08h System Timer Interrupt Service Routine entry point
FFEF3h	Initial Interrupt Vector offsets loaded by POST
FFF53h	IRET Instruction for Dummy Interrupt Handler
FFF54h	INT 05h Print Screen Service entry point
FFFF0h	Power-On entry point
FFFF5h	ROM Date (in ASCII). Eight characters in mm/dd/yy format.
FFFFEh	System Model ID (always FCh).

Floppy Disk Drive Parameters

The floppy diskette parameter table is pointed to by the INT 1Eh vector. The table is eleven bytes long.

Offset	Description
00h	Bits 7–4 Head Unload Time in milliseconds. The amount of time needed to allow the drive head to settle after it lifts from the drive surface. 0h 32 ms 1h 64 ms 2h 96 ms 3h 128 ms The default for 2.88 MB drives is 120 4h 160 ms 5h 192 ms 6h 240 ms The default value for 1.2 MB drive. 7h 256 ms 8h 288 ms 9h 320 ms Ah 352 ms Bh 384 ms Ch 399 ms The default for 360 KB and 1.2 MB. Dh 448 ms Eh 480 ms The default for 360 KB and 720 KB. 0Fh 512 ms Bits 3–0 Step Rate in milliseconds. The amount of time needed for a drive head to move from one track to another. 00h 2 The default for 1.2 MB and 2.88 drives is 3.0 ms. 01h 4 The default for 360 KB in 1.2 MB drive is 4.8 ms. 02h 6 The default for 360 KB, 720 KB, and 1.44 MB 03h 8 04h10 05h 12 06h14 07h 16 08h18 09h 20 0Ah 22 0Bh 24 0Ch 26 0Dh 28 0Eh 30 0Fh 32
01h	Head Load Time. The amount of time in milliseconds needed to allow the drive head to settle after it is lowered onto the drive surface. The value ranges from 00h – 7Fh in increments of 4 milliseconds. See the following table for default values. Bits 7–0 00h 4 ms 01h 8 ms 02h 12 ms 03h 16 ms 04h 20 ms 05h 24 ms 06h 28 ms 07h 32 ms 08h 36 ms 09h 40 ms ... ... 7Fh 512 ms Bit 0 Non-DMA Mode Flag (Always 0 to indicate that DMA is used).

Offset	Description
02h	Motor Wait Timer. The length of time a floppy drive can be inactive before the drive motor is shut off. This value ranges from 0 to 255 in increments of 1. The timer ticks approximately 18.2 per second. The Motor Wait Time value can be calculated as follows: TIME = Selected timer tick value divided by 18.2 Bits 7–0 00h 0 timer ticks 01h 1 timer tick 02h 2 timer ticks ... ... 37h 37 timer ticks (Default for all floppy drives — approximately 2.03 seconds) ... ... FFh 255 timer ticks
03h	Number of Bytes per Sector Bits 7–0 00h 128 bytes per sector 01h 256 bytes per sector 02h 512 bytes per sector (Default for all floppy drives). 03h 1024 bytes per sector
04h	Number of Sectors Per Track Bits 7–0 08h 8 sectors per track (320 KB 5¼" drives) 09h 9 sectors per track (360 KB and 720 KB 5¼" drives) 0Fh 15 sectors per track (1.2 MB 5¼" drives) 12h 18 sectors per track (720K and 1.44 MB 3½" drives) 24h 36 Sectors per track (2. 88 MB 3½" drives)
05h	Gap Length. The length of the gap between sectors. Bits 7–0 00h 0 01h 1 ... ... 1Bh 27 The default value for all 3½" drives and 1.2 MB 5¼" drives. ... ... 2Ah 42 The default for 360 KB and 720 KB floppy drives. ... ... FFh 255
06h	Data Length — always set to FFh.
07h	Gap length for format. This value is used for the same purpose as the gap length, but it is used in formatting only. Bits 7–0 00h 0 01h 1 ... ... 50h 80 The default value for 360 KB, 720 KB, and 2.88 MB floppy drives. 51h 84 The default value for 1.2 MB floppy drives. ... ... 57h 108 The default value for 1.44 MB floppy drives. ... ... FFh 255
08h	Fill Byte for Formatting — always set to F6h
09h	Head Settle Time. The amount of time in milliseconds that must elapse to allow the heads to settle after a Seek operation. Bits 7–0 00h 0 ms 01h 1 ms 02h 2 ms ... ... 0Fh 15 ms (default for all floppy drives) ... ... FFh 255 ms

Offset	Description
0Ah	Motor Start Time. The amount of time it takes the drive motor to reach optimal speed. The values are in eighths of a second. Bits 7–0 00h 0 01h one/eighth of a second 02h 2 ¼ second ... ... 08h 8 one second (default for all floppy drives) ... ... FFh 255 one/32nd of a second

Summary of Default Settings for Floppy Drives

The following table summarizes the default settings for all floppy disk parameter table values in the AMIBCP:

Parameter	360 KB Floppy in 360KB Drive	360 KB Floppy in 1.2 MB Drive	1.2 MB Floppy in 1.2 MB Drive	720 KB 3½"	1.44 MB 3½"	2.88 MB 3½"
Step Rate (ms)	6.0	4.8	3.0	6.0	6.0	3.0
Head Unload Time (ms)	480	399	240	480	240	120
Head Load Time (ms)	4.0	3.3	2.0	4.0	2.0	1.0
Motor Wait Time (in timer ticks)	37	37	37	37	37	37
Gap Length	42	42	27	42	27	27
Gap Length for Format	80	80	84	80	108	80
Head Settle Time (ms)	15	15	15	15	15	15
Motor Start Time (in _ths of a second)	8	8	8	8	8	8
Number of Bytes per Sector	512	512	512	512	512	512
Cluster Size	1024	1024	512	1024	512	512
Tracks	40	40	80	80	80	80
Sectors per Track	9	9	15	9	18	36

Hard Disk Parameter Table

The hard disk drive parameter table (drive type table) is located at F000:E401h. The vector table entries for INT 41h contains the entry points for the hard disk drive types selected via BIOS Setup for hard disk drive C:. INT 46h contains the vector for hard disk drive D:. Each drive type entry consists of 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 most significant byte. The size of a sector gets progressively smaller as the track diameter diminishes. Yet each sector must still hold 512 bytes. Write precompensation circuitry on the hard disk compensates for the physical difference in sector size by boosting the write current for sectors on inner tracks. This parameter is the track number where write precompensation begins.
07h	Reserved
08h	Control Byte Bits 7–6 Enable retries 00 Enable retries. All other values disable retries. Bit 5 Set if defect map is located 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. This number is the cylinder location where the heads normally park when the computer is shut down.
0Eh	Number of Sectors per Track. Hard disk drives that use MFM have 17 sectors per track. RLL drives have 26 sectors per track. RLL and ESDI drives have 34 sectors per track. IDE and SCSI drives may have even more sectors per track.
0Fh	Reserved

Cont’d

Hard Disk Parameter Table, Continued

AMIBIOS uses a standard hard disk drive type table that has 45 entries for drive types 0 – 14 and 16 – 46. A complete list of the hard disk drive parameters appears later in this manual.

These drive types can be used to configure DOS drives C: and D:.

This table can also be modified using AMIBCP. See the *AMIBCP User's Guide* for more information.

User-Definable Drives Older versions of AMIBIOS also support two user-definable drive types (Type 47 C: and Type 47 D:) to be used with hard disk drives not defined in the standard drive table. You can configure these drive types via AMIBIOS Setup. AMIBIOS with a date of 7/15/95 or later automatically configure IDE drives.

Location of Hard Drive Parameters The hard disk drive parameters are stored in CMOS RAM registers 1Bh – 23h (drive C:) and 24h – 2Ch (drive D:). The format is shown later in this manual. The BIOS rewrites these parameters at system boot to a different location to permit quicker access.

Cont'd

Hard Disk Parameter Table, Continued

Hard Disk Drive Type Selection AMIBIOS first makes sure shadow RAM is enabled. If so, the BIOS copies these parameters to the locations in the drive table specified by the INT 41h (Drive C: or primary master) and INT 46h (Drive D: or primary slave) vectors.

If shadow RAM is disabled or the computer does not support shadow RAM, the parameters are copied to either of two secondary locations:

- the BIOS Stack Area (000300h – 000301h), or
- the upper 1 KB of DOS memory (09FFFEh – 09FFFFh).

You can select the secondary location to be used through AMIBCP, or can allow you to choose the secondary location via AMIBIOS Setup.

Hard Disk Drive Capacity The capacity of a hard disk drive can be determined using the following formula:

(Number of heads) X (Number of cylinders) X (Number of sectors per track) X (512 - Number of bytes per sector)

Hard Disk Drive Types

The following values are the default MFM hard drive parameter values. The OEM can modify these values using AMIBCP.

The Control Byte, the only hard disk drive parameter not shown in the following table, is described later in this manual. The Control Byte is almost always 00h in the AMIBIOS Hard Disk Parameter Table. The only exceptions are drive types 9, 25, 27, 29, 32, 35, 44, 45, and 46, where it is 80h.

Cont'd

Hard Drive Types, Continued

Type	Number of Cylinders	Number of Heads	Write Precompensation	Landing Zone	Number of Sectors	Size
1	306	4	128	305	17	10 MB
2	615	4	300	615	17	20 MB
3	615	6	300	615	17	31 MB
4	940	8	512	940	17	62 MB
5	940	6	512	940	17	47 MB
6	615	4	65535	615	17	20 MB
7	462	8	256	511	17	31 MB
8	733	5	65535	733	17	30 MB
9 *	900	15	65535	901	17	112 MB
10	820	3	65535	820	17	20 MB
11	855	5	65535	855	17	35 MB
12	855	7	65535	855	17	50 MB
13	306	8	128	319	17	20 MB
14	733	7	65535	733	17	43 MB
16	612	4	0	663	17	20 MB
17	977	5	300	977	17	41 MB
18	977	7	65535	977	17	57 MB
19	1024	7	512	1023	17	60 MB
20	733	5	300	732	17	30 MB
21	733	7	300	732	17	43 MB
22	733	5	300	733	17	30 MB
23	306	4	0	336	17	10 MB
24	925	7	0	925	17	54 MB
25 *	925	9	65535	925	17	69 MB
26	754	7	754	754	17	44 MB
27 *	754	11	65535	754	17	69 MB
28	699	7	256	699	17	41 MB
29 *	823	10	65535	823	17	68 MB
30	918	7	918	918	17	53 MB
31	1024	11	65535	1024	17	94 MB
32 *	1024	15	65535	1024	17	128 MB
33	1024	5	1024	1024	17	43 MB
34	612	2	128	612	17	10 MB
35 *	1024	9	65535	1024	17	77 MB
36	1024	8	512	1024	17	68 MB
37	615	8	128	615	17	41 MB
38	987	3	987	987	17	25 MB
39	987	7	987	987	17	57 MB
40	820	6	820	820	17	41 MB
41	977	5	977	977	17	41 MB
42	981	5	981	981	17	41 MB
43	830	7	512	830	17	48 MB
44 *	830	10	65535	830	17	69 MB
45 *	917	15	65535	918	17	114 MB
46 *	1224	15	65535	1223	17	152 MB
47	Enter hard disk drive parameters supplied by disk drive manufacturer.					

* Control Byte is 80h

Enhanced IDE Hard Disk Drive Parameters

Additional parameter support is needed to support enhanced IDE drives. The physical enhanced IDE parameter table structure is:

Offset	Length	Description
00h	Word	Physical number of cylinders
02h	Byte	Physical number of heads
03h	Byte	Reserved
04h	Byte	Reserved
05h	Word	Starting write precompensation cylinder
07h	Byte	Reserved
08h	Byte	Drive control byte
09h	Word	Reserved
0Bh	Byte	Reserved
0Ch	Word	Landing Zone
0Eh	Byte	Physical sectors per track
0Fh	Byte	Reserved

Enhanced IDE Drive Parameter Table The logical parameters are used for the INT 13h Function 08h interface. The physical parameters are used for the IDE interface (the Identify Drive command, Set Parameters command, etc.).

Offset	Length	Description
00h	Word	Logical number of cylinders
02h	Byte	Logical number of heads
03h	Byte	A0h (A signature that specifies a translated parameter table)
04h	Byte	Physical sectors per track
05h	Word	Starting write precompensation cylinder
07h	Byte	Reserved
08h	Byte	Drive control byte
09h	Word	Physical number of cylinders
0Bh	Byte	Physical number of heads
0Ch	Word	Landing Zone
0Eh	Byte	Logical sectors per track
0Fh	Byte	Checksum. The value of the Checksum byte is such that the summation of all the bytes (including the checksum byte) in the parameter table is 00.

Extended Drive Parameter Table

The extended drive parameter table stores enhanced IDE drive configuration information

Offset	Length	Description
00h	Word	I/O Port Base Address for accessing the drive.
02h	Word	Control Port Address is the drive control port address.
04h	Byte	Master/Slave/LBA information Bit 7 Reserved (set to 1) Bit 6 LBA Enable 0 Disabled 1 Enabled Bit 5 Reserved (set to 1) Bit 4 Master/Slave bit 0 Master 1 Slave Bits 3-0 Reserved (set to 0)
05h	Byte	Reserved (set to 0)
06h	Byte	IRQ information Bits 7-4 Reserved (set to 0) Bits 3-0 IRQ used by this drive
07h	Byte	Sector Count for Block mode (multisector) transfer. The number of sectors transferred in one hardware interrupt.
08h	Byte	If the drive is configured for DMA data transfer, the DMA channel and type are specified: Bits 7-4 DMA Type Bits 3-0 DMA Channel
09h	Byte	Advanced PIO Mode information. The PIO mode is specified if the drive is configured for advanced PIO mode data transfer. Bits 7-4 Reserved (set to 0) Bits 3-0 PIO Mode 0 Mode 0 1 Mode 1 2 Mode 2 3 Mode 3 4 Mode 4

Offset	Length	Description
0Ah	Word	Configuration information Bits 15-8 Reserved (set to 0) Bit7 32-bit Transfer Mode 0 Disable 1 Enable Bit 6 CD-ROM 0 Disable 1 Enable Bit 5 Removable Media 0 Disable 1 Enable Bit 4 LBA Translation 0 Disable 1 Enable Bit 3 CHS Translation. If the drive has more than 1024 cylinders and AMIBIOS handles it without LBA mode support (AMIBIOS translates physical parameters to logical parameters internally to support more than 1024 cylinders), this bit is 1. Bit 2 Block Mode Transfer. See offset 07h for the number of sectors that can be transferred per hardware interrupt. 0 Disable 1 Enable Bit 1 DMA access. See offset 08h for the DMA channel and type. 0 Disable 1 Enable Bit 0 Advanced PIO Mode. See offset 09h for the configured PIO mode. 0 Disable 1 Enable
	Word	Reserved (set to 0)
0Eh	Byte	Revision Number of this extension in BCD. The current revision is 10h. The first digit is the major revision number. The second digit is the minor revision number.
0Fh	Byte	Checksum

IDE LBA Mode

IDE LBA Mode LBA Mode must be set as follows depending on the operating system.

Operating System	Necessary Action
SCO UNIX 3.2.4	LBA mode must be disabled in WINBIOS Setup.
Novell NetWare	LBA mode must be disabled in WINBIOS Setup.
DOS	LBA mode must be enabled in WINBIOS Setup if the IDE hard disk capacity is greater than 528 MB.
Windows 3.x	LBA mode must be enabled in WINBIOS Setup if the IDE hard disk capacity is greater than 528 MB.
Windows 95	LBA mode must be enabled in WINBIOS Setup if the IDE hard disk capacity is greater than 528 MB.
Windows NT	LBA mode must be enabled in WINBIOS Setup if the IDE hard disk capacity is greater than 528 MB.
OS/2 2.1	LBA mode must be enabled in WINBIOS Setup if the IDE hard disk capacity is greater than 528 MB.

Hard Disk Drive Data Transfer Rates

Data transfer rates for some common hard disk drives are:

Drive Interface Type	Megabits per second	Megabytes per second
ST506 and ST412	5 Mbs	0.625 MBs
RLL	7.5 Mbs	0.9375 MBs
IDE	7.5 Mbs	0.9375 MBs
Enhanced IDE	88.0 Mbs	11 MBs
ESDI	10 Mbs	1.25 MBs
SCSI-2	80 – 320 Mbs	10 – 40 MBs

Video Parameter Table

The video parameter table always contains one or more entries for each available video mode, including MDA, CGA, EGA, PGA, XGA, or VGA standard modes. If VGA is used, this table contains at least 29 entries in the following format:

Offset	Description
00h	Number of displayed character columns (the same value as in 40:49h).
01h	Number of displayed character rows - 1 (the same value as in 40:84h).
02h	Height of character matrix (the same value as in 40:85h).
03h	Size of video buffer in bytes (the same value as in 40:4Ch).
05h	The value for Sequencer Registers 1 through 4.
09h	The value for the Miscellaneous Output Register.
0Ah	Values for CRTC Registers 00h through 18h.
23h	The values for Attribute Control Registers 00h through 13h.
37h	The values for Graphics Controller Registers 0 through 8.

System Configuration Data

The System Configuration Table is at F000:E6F5h. It can be moved to system memory via INT 15h Function C0h.

Offset	Description
00h	Number of bytes in this table. It must be at least eight bytes.
02h	Model Byte (always FCh)
03h	Submodel Byte (always 01h)
04h	BIOS Revision Level. Should be zeros if this is the first release of the BIOS.
05h	Bit 7 1 The hard disk drive BIOS uses DMA Channel 3. Bit 6 1 A second interrupt controller chip is present. Bit 5 1 A Real Time Clock is present. Bit 4 1 INT 15h Function 4Fh has been called by INT 09h. Bits 3–0 Reserved, should be zeros.
06h – 09h	Reserved, should be zeros.

Serial Port Data Transmission Rates

Data Transmission Rate Initialization Table This table is at F000:E729h in the ROM BIOS.

Data Transmission Rate	Divisor
110	0417h
150	0300h
300	0180h
600	00C0h
1200	0060h
2400	0030h
4800	0018h
9600	000Ch
14400	
19200	0006h
28800	
57600	
119200	

Data Transmission Rate Divisors The input frequency to the device is 1.8432 MHz. The values in the table are calculated as follows: 11,843,200 divided by 16 = 115,200 divided by data transmission rate = Divisor. For example, a data transmission rate of 2400 has a divisor of 115,200 divided by 2,400, which equals 30h.

ROM Compatibility Information

Address	Description
F000:F104h	APM Connection Information Bit 7 Reserved Bit 6 0 APM BIOS power management enabled 1 APM BIOS power management disabled Bit 5 0 APM CPU idle call stops CPU clock 1 APM CPU idle call slows CPU clock Bit 4 0 APM BIOS active version 1.0 1 APM BIOS active version above 1.0 Bit 2 1 APM 32-bit protected mode connection on. Bit 1 1 APM 16-bit protected mode connection on. Bit 0 1 APM real mode connection established
F000:F105h	APM State Information Bit 7 1 Power management enabled by APM Bit 6 1 Cooperative power management between the APMBIOS and an APM drive is enabled for this device (Version 1.1 only). Bit 5 1 APM Function Number 0Ch is currently enabled for this device (Version 1.1 only). Bit 3 1 APM controlled device (Version 1.1 only). Bits 2-0 APM device state 000 APM enabled state 001 APM standby state 010 APM suspend state 011 APM off state 111 APM on state
F000:F106h	Pending APM Event Information Bit 10 1 System standby resume notification (V1.1 only). Bit 9 1 User system suspend request notification (V1.1). Bit 8 1 User system standby request notification (V1.1 only Bit 7 1 Critical system suspend notification (Version 1.1). Bit 6 1 Update time notification (Version 1.1.only). Bit 5 1 Power status change notification (V1.1 only). Bit 4 1 Battery low notification Bit 3 1 Critical resume system notification Bit 2 1 Normal resume system notification Bit 1 1 System suspend request notification Bit 0 1 System standby request notification
F000:FEA8h	Pointer to Bootstrap CPU information array
F000:FEACh	Pointer to FPU information array
F000:FEB0h	Pointer to Application CPU information array

ROM Compatibility Table

The contents of the F000:EEC0h - F000:EF56h system memory area contains absolute addresses to provide a common interface to commonly-used system BIOS routines. These routines can be called by a segment different from F000h without duplicating the routines.

Address	Description
F000:EEC0h	Uncompress routine **Input:** AX:BX Source compressed code segment:offset CX:DX Destination segment:offset where the code is uncompressed. **Output:** None **Registers:** None affected **Use:** Must be invoked as CALL FAR
F000:EEC3h	System BIOS FIXED_DELAY routine, invoked as CALL FAR.
F000:EEC6h	Get_Processor_Info routine, invoked as CALL FAR.
F000:EEC9h	Get_Reset_ID routine, invoked as CALL FAR.
F000:EECCh	Get_CPU_ID routine, invoked as CALL FAR.
F000:EECFh	Get_Vendor_Name routine, invoked as CALL FAR.
F000:EED2h	BIOS INT-13 Hard Disk Handler entry point, invoked as INT.
F000:EED5h	Check_CMOS_Data routine, invoked as CALL FAR.
F000:EED8h	Uncompress_General_Module routine, invoked as CALL FAR.
F000:EEDBh	Uncompress_PCI_Module routine, invoked as CALL FAR.
F000:EEDEh -F000:EE4Ch	Reserved for internal use.
F000:EF4Dh	APM 16-bit protected mode entry point, invoked as CALL FAR.
F000:EF50h	APM 32-bit protected mode entry point, invoked as CALL FAR.

5 CMOS RAM

Computers that adhere to ISA standards have at least 64 bytes of CMOS RAM to store system initialization and configuration parameters.

How CMOS RAM is Configured Most of these parameters are set by the computer manufacturer and the user via the BIOS Setup utility. AMIBIOS provides the AMIBIOS Setup utility in the BIOS ROM that can be accessed during startup.

Accessing CMOS RAM Directly You can access CMOS RAM via an assembly language program. To read CMOS RAM, use the following Intel x86 assembler instructions:

```
OUT  70h,Register Number
IN  71h
```

To write to CMOS RAM, use the following instructions:

```
OUT  70h,Register Number
OUT  71h,New_Value
```

If the most significant bit of the *Register Number* is set when reading or writing CMOS RAM, the Nonmaskable Interrupt (NMI) is disabled during the operation.

How CMOS RAM is Organized

CMOS RAM is divided into several parts:

Location	Length	Description
00h – 0Fh	16 bytes	Real Time Clock data.
10h – 2Fh	32 bytes	ISA configuration data.
30h – 3Fh	16 bytes	AMIBIOS-specific configuration data.
40h – 7Fh	64 bytes	Extended CMOS RAM. Available in many computers. Many chipsets incorporate this additional CMOS RAM to store advanced configuration information.

EISA CMOS RAM EISA Extended CMOS RAM stores EISA-specific information and is configured by the EISA Configuration Utility (ECU). EISA Extended CMOS RAM consists of between 4,096 and 8,192 bytes of CMOS memory and is accessed via INT 15h Function D8h. EISA Extended CMOS RAM can also be accessed via a series of I/O ports.

AMIBIOS 98 CMOS RAM Map

There is no generic map of CMOS RAM (NVRAM) for AMIBIOS 98. The AMISSP Setup script parameter utility automatically assigns available CMOS RAM locations, but AT-compatible CMOS RAM registers cannot be changed:

Offset	Description
00h - 10h	Standard IBM AT-compatible RTC and status register data definitions.
11h - 13h	Varies
14h	Bits 7-6 Number of floppy drives 00 1 Drive 01 2 Drives Bits 5-4 Monitor Type 00 Monochrome 01 40x25 CGA 10 80x25 CGA 11 VGA/EGA Bit 3 Display Enabled 0 Disabled 1 Enabled Bit 2 Keyboard Enabled 0 Disabled 1 Enabled Bit 1 Math Coprocessor 0 Absent 1 Present Bit 0 Floppy Drive 0 Disabled 1 Enabled
15h	Base Memory (in 1 KB increments), Low Byte
16h	Base Memory (in 1 KB increments), High Byte
17h	IBM-compatible RAM (in 1 KB increments), Low Byte
18h	IBM-compatible RAM (in 1 KB increments), High Byte (Max 15 MB)
19h - 2Dh	Varies
2Eh	Standard CMOS RAM checksum, high byte
2Fh	Standard CMOS RAM checksum, low byte
30h	IBM-compatible Extended Memory, Low Byte (POST) in KB
31h	IBM-compatible Extended Memory, High Byte (POST) in KB
32h	Century byte
33h	Reserved. Do not use.
34h	Reserved. Do not use.
35h	Low byte of extended memory (POST) in 64 KB
36h	High byte of extended memory (POST) in 64 KB
37h - 3Dh	Varies
3Eh	Extended CMOS Checksum, Low Byte (includes 34h - 3Dh)
3Fh	Extended CMOS Checksum, High Byte (includes 34h - 3Dh)

AMIBIOS Setup Monochrome Table

The following table lists the monochrome values for AMIBIOS Setup screens.

Color Number	Third Window	First Window	Second Window	Main Window
0	07h	70h	70h	07h
1	70h	70h	70h	70h
2	07h	07h	07h	07h
3	70h	07h	07h	70h
4	70h	70h	07h	07h
5	07h	70h	70h	70h
6	07h	07h	07h	70h
7	70h	07h	07h	07h
8	70h	70h	70h	07h
9	07h	70h	07h	07h
A	07h	07h	70h	70h
B	70h	07h	70h	07h
C	07h	70h	07h	70h
D	07h	07h	70h	07h
E	70h	07h	70h	70h
F	70h	70h	07h	70h

AMIBIOS Setup Color Table

The following table lists the color values for AMIBIOS Setup screens.

Color Number	Third Window	First Window	Second Window	Main Window
0	57h	60h	17h	20h
1	20h	47h	30h	57h
2	70h	30h	57h	60h
3	60h	17h	20h	70h
4	57h	30h	47h	20h
5	20h	17h	60h	57h
6	17h	60h	57h	30h
7	30h	47h	20h	17h
8	70h	20h	17h	60h
9	60h	57h	30h	70h
A	70h	70h	70h	70h
B	07h	07h	07h	07h
C	70h	07h	70h	07h
D	07h	70h	07h	70h
E	17h	20h	47h	30h
F	30h	57h	60h	17h

6 I/O Port Addresses

The CPU communicates with and controls many parts of the computer via the I/O ports. I/O ports are doorways through which information passes as it travels from an I/O device (such as a keyboard or serial port) to the CPU and back. The ISA architecture includes a 64 KB I/O memory area used to access external devices. Intel 386, 486, Pentium®, and Pentium Pro architectures allow for 8-, 16-, or 32-bit I/O ports. The I/O ports from 0000h – 00FFh address devices on the motherboard. Ports 0100h – 02FFh address devices attached to the computer via expansion slots. I/O port addresses 01F0h – 01F8h are reserved for a hard disk controller.

Components that Use the I/O Ports Most of the support chips in an AT (Intel 8259 Programmable Interrupt Controller, Intel 8254 Programmable Interval Timer, Intel 8237 Programmable DMA Controller and equivalents) use the I/O port to communicate with other parts of the computer.

How Ports Are Identified Each port is identified by a 16-bit port number (from 0 – 65,535, or 00000h – FFFFFh).

Ports Accessed by Hex Port Number The CPU sends data or control information to a specific I/O port by specifying the port number. The I/O port passes data or status information through the port to the CPU. Just as it does when accessing memory, the CPU uses the data and address buses as paths for communication with the I/O ports. To access a port, the CPU first sends a signal on the control bus. This signal notifies all I/O devices that the information on the bus is an I/O port address. Then it sends the I/O port address. The device assigned to that specific I/O port responds. The I/O port number addresses a memory location that is part of the I/O device, but is not part of system memory. Special assembler I/O instructions are used to signal a port access and send information to and from I/O devices.

Accessing I/O Ports

Programmer Access A programmer can use the Intel assembly language instructions IN and OUT to write to and read from I/O port addresses. For example:

```
MOV     DX,03CCh ;read video register address
IN      AL,DX   ;read byte from I/O port
OR      AL,10h   ;set bit 4 for RAMDAC control
MOV     DX,03C2h ;write register address
OUT     DX,AL   ;write value to I/O port
```

ISA and EISA I/O Port Assignments

All ISA and EISA computers use standard I/O ports. The standard I/O port addresses for all AMIBIOS products are listed in the following tables. Some computers also use customized I/O port assignments.

Hardware I/O Port Addresses The I/O port addresses from 0000h–00FFh are used by components mounted on the motherboard.

I/O Port	Read/ Write	Description
0000h – 001Fh are used by the 8237 DMA Controller 1		
0000h	R/W	DMA channel 0 address byte 0 (low byte), followed by byte 1.
0001h	R/W	DMA channel 0 word count byte 0 (low byte), followed by byte 1.
0002h	R/W	DMA channel 1 address byte 0 (low byte), followed by byte 1.
0003h	R/W	DMA channel 1 word count byte 0 (low byte), followed by byte 1.
0004h	R/W	DMA channel 2 address byte 0 (low byte), followed by byte 1.
0005h	R/W	DMA channel 2 word count byte 0 (low byte), followed by byte 1.
0006h	R/W	DMA channel 3 address byte 0 (low byte), followed by byte 1.
0007h	R/W	DMA channel 3 word count byte 0 (low byte), followed by byte 1.
0008h	R	DMA channels 0–3 status register Bit 7 1 Channel 3 request Bit 6 1 Channel 2 request Bit 5 1 Channel 1 request Bit 4 1 Channel 0 request Bit 3 1 Terminal count on channel 3 Bit 2 1 Terminal count on channel 2 Bit 1 1 Terminal count on channel 1 Bit 0 1 Terminal count on channel 0

I/O Port	Read/ Write	Description
0008h	W	DMA channels 0–3 command register Bit 7 0 DACK sense active low 1 DACK sense active high Bit 6 0 DREQ sense active low 1 DREQ sense active high Bit 5 0 Late write selection 1 Extended write selection Bit 4 0 Fixed priority 1 Rotating priority Bit 3 0 Normal timing 1 Compressed timing Bit 2 0 Enable controller 1 Disable controller Bit 1 0 Disable memory-to-memory transfer 1 Enable memory-to-memory transfer Bit 0 Reserved
0009h	W	DMA write request register
000Ah	R/W	DMA channel 0–3 mask register Bits 7–3 Reserved Bit 2 0 Clear mask bit 1 Set mask bit Bits 1–0 Channel select 00 Channel 0 01 Channel 1 10 Channel 2 11 Channel 3
000Bh	W	DMA channel 0–3 mode register Bits 7–6 Mode select 00 Demand mode 01 Single mode 10 Block mode 11 Cascade mode Bit 5 0 Address increment select 1 Address decrement select Bit 4 0 Disable autoinitialization 1 Enable autoinitialization Bits 3–2 Select type of operation 00 Verify operation 01 Write to memory 10 Read from memory 11 Reserved Bits 1–0 Channel select 00 Channel 0 01 Channel 1 10 Channel 2 11 Channel 3
000Ch	W	DMA clear byte pointer flip/flop
000Dh	R	DMA read temporary register
000Eh	W	DMA clear mask register
000Fh	W	DMA write mask register
I/O ports 0020h – 0021h are used by the programmable interrupt controller.		

I/O Port	Read/ Write	Description
0020h	W	If bit 4 is set, this is the programmable interrupt controller Initialization Command Word 1 (ICW1). Bits 7–5 000 Only used in 8080 or 8085 mode Bit 4 1 Using ICW1 Bit 3 0 Edge-triggered mode 1 Level-triggered mode Bit 2 0 Successive interrupt vectors separated by eight bytes 1 Successive interrupt vectors separated by four bytes Bit 1 0 Cascade mode 1 Single mode. ICW3 is not necessary Bit 0 0 ICW4 is not necessary 1 ICW4 is necessary
0021h	W	This port can represent ICW2, ICW3, and ICW4 in sequence after ICW1 is written to I/O port 0020h. If ICW2 Bits 7–3 Address lines A0–A3 of the base vector address for the interrupt controller. Bits 2–0 Reserved, should be zeros. If ICW3 for the slave controller (00A1h) Bits 7–3 Reserved Bits 2–0 Slave ID If ICW4 Bits 7–5 Reserved (should be zeros) Bit 4 0 No special fully-nested mode 1 Special fully-nested mode Bits 3–2 Mode 00 Non-buffered mode 01 Non-buffered mode 10 Buffered mode/slave 11 Buffered mode/master Bit 1 0 Normal EOI 1 Auto EOI Bit 0 0 8085 mode 1 8086 and 8088 mode
0021h	R/W	Programmable interrupt controller master interrupt mask register — Operation Command Word 3 (OCW1) Bit 7 0 Enable parallel printer interrupt Bit 6 0 Enable floppy disk drive interrupt Bit 5 0 Enable hard disk drive interrupt Bit 4 0 Enable serial port 1 interrupt Bit 3 0 Enable serial port 2 interrupt Bit 2 0 Enable video interrupt Bit 1 0 Enable keyboard/mouse/RTC interrupt Bit 0 0 Enable timer interrupt

I/O Port	Read/ Write	Description
0020h	W	If Bits 4 and 3 are 0, Programmable interrupt controller OCW2. Bits 7–5 000 Rotate in automatic EOI mode (clear) 001 Non-specific EOI 010 No op 011 Specific EOI 100 Rotate in automatic EOI mode (set) 101 Rotate on nonspecific EOI command 110 Set priority command 111 Rotate on specific EOI command Bits 4–3 Reserved (should be zeros) Bits 2–0 The interrupt request the command applies to
0020h	R	Programmable interrupt controller Interrupt request and In-Service registers programmed by OCW3 Interrupt request register Bits 7–0 0 No active request for the corresponding interrupt line 1 Active request for the corresponding interrupt line Interrupt in-service register Bits 7–0 0 The corresponding interrupt line is not being serviced now 1 The corresponding interrupt line is being serviced now
0020h	W	If Bit 4 is 0 and Bit 3 is 1, Programmable interrupt controller OCW3 Bit 7 Reserved (should be zero) Bits 6–5 00 No op 01 No op 10 Reset special mask 11 Set special mask Bit 4 Reserved, should be zero. Bit 3 Reserved, should be one. Bit 2 0 No poll command 1 Poll command Bits 1–0 00 No op 01 No op 10 Read interrupt request register on next read of port 0020h 11 Read interrupt in-service register on next read of I/O port 0020h
I/O ports 0040 – 005Fh can be used by the Programmable Interrupt Timer.		
0040h	R/W	Programmable interrupt timer R/W counter 0, keyboard controller channel 0
0041h	R/W	Programmable interrupt timer channel 1
0042h	R/W	Programmable interrupt timer miscellaneous register channel 2

I/O Port	Read/ Write	Description
0043h	W	Programmable interrupt timer mode port. Control word register for counters 0 and 2 Bits 7–6 00 Counter 0 select 01 Counter 1 select 10 Counter 2 select Bits 5–4 00 Counter latch command 01 R/W counter bits 0–7 only 10 R/W counter bits 8–15 only 11 R/W counter bits 0–7 first, then bits 8–15 Bits 3–1 Select mode 000 Mode 0 programmable one-shot x10 Mode 1 rate generator x11 Mode 3 square wave generator 100 Mode 4 software-triggered strobe 101 Mode 5 hardware-triggered strobe Bit 0 0 Binary counter is 16 bits 1 Binary code decimal (BCD) counter
0044h	W	Programmable interrupt controller miscellaneous register (EISA)
0047h	W	Programmable interrupt timer Control word register four counter 0 (EISA) Bits 7–6 00 Counter 0 All other values reserved Bits 5–4 00 Counter latch command select counter 0 01 R/W counter bits 0–7 only All other values reserved
0048h	R/W	Programmable interrupt Timer
0060h	R	Keyboard controller data port or keyboard input buffer. If Keyboard input buffer (can also be 64h): Bit 7 0 Keyboard inhibited Bit 6 0 Primary display is VGA 1 Primary display is MDA. Bit 5 0 System BIOS performs diagnostics on the motherboard in an infinite loop. 1 Any other diagnostic function Bit 4 Motherboard RAM 0 256 KB 1 512 KB or greater Bits 3–1 Reserved Bit 0 0 The motherboard passed the diagnostics tests when diagnostic mode was enabled. The LED blinks in manufacturing diagnostic mode.
0060h	W	Keyboard output port (can also be port 64h) Bit 7 0 Keyboard data is being transferred Bit 6 0 The keyboard clock signal is being used in data transfer Bit 5 0 PC-type mouse being used 1 PS/2-type mouse being used Bit 4 0 Output buffer full, IRQ1 generated 1 Output buffer not full Bit 1 0 The processor address 20 line is inhibited on the system bus. 1 Address line 20 is not inhibited Bit 0 0 Reset system processor 1 This bit should always be kept at 1.

I/O Port	Read/ Write	Description
0061h	R	Port B control register (EISA computers) Bit 7 1 Parity check Bit 6 1 Channel check Bit 5 1 Timer 2 output Bit 4 1 Toggles with each refresh request Bit 3 1 Channel check enable Bit 2 1 Parity check enable Bit 1 1 Speaker data enable Bit 0 1 Timer 2 gate to speaker enable
0061h	W	Port B Control register (EISA computers) Bit 3 1 Channel check enable Bit 2 1 Parity check enable Bit 1 1 Speaker data enable Bit 0 1 Timer 2 gate to speaker enable
0064h	R	Keyboard controller read status Bit 7 0 No parity error 1 Parity error on last byte of transmission from keyboard Bit 6 0 No timeout 1 Received a timeout on last transmission Bit 5 0 No timeout 1 Transmission from keyboard controller to keyboard timed out Bit 4 0 Keyboard inhibited 1 Keyboard not inhibited Bit 3 0 Data. The computer writes to input buffer via I/O port 60h. 1 Command. The computer writes to input buffer via I/O port 64h Bit 2 System Flag status. Set to 0 after a power on reset. The keyboard controller sets this bit according to the command from the computer. Bit 1 0 Input buffer (60h or 64h) is empty 1 Input buffer full Bit 0 0 Output buffer has no data 1 Output buffer full
0070h	R	Real Time Clock (CMOS RAM) register and NMI mask Bit 7 1 NMI disabled Bits 6–0 0 CMOS RAM index
0071h	R/W	CMOS RAM data register port
0080h	R	Manufacturing test port (POST checkpoints can be accessed via this port).
0080h	R/W	Temporary storage for additional DMA page register
0081h	R/W	DMA channel 2 address byte 2
0082h	R/W	DMA channel 2 address byte 3
0083h	R/W	DMA channel 1 address byte 2
0084h	R/W	Additional DMA page register
0085h	R/W	Additional DMA page register
0086h	R/W	Additional DMA page register
0087h	R/W	DMA channel 0 address byte 2
0088h	R/W	Additional DMA page register
0089h	R/W	DMA channel 6 address byte 2

I/O Port	Read/ Write	Description
008Ah	R/W	DMA channel 7 address byte 2
008Bh	R/W	DMA channel 5 address byte 2
008Ch	R/W	Additional DMA page register
008Dh	R/W	Additional DMA page register
008Eh	R/W	Additional DMA page register
008Fh	R/W	DMA refresh page register
00A0h – 00A1h are used for the slave programmable interrupt controller. Except for the differences noted below, the bit definitions are the same as those for addresses 0020h – 0021h.		
00A0h	R/W	Programmable interrupt controller 2
00A1h	R/W	Programmable interrupt controller 2 mask (OCW1) Bit 7 0 Reserved Bit 6 0 Enable hard disk drive interrupt Bit 5 0 Enable coprocessor exception interrupt Bit 4 0 Enable mouse interrupt Bits 3–2 Reserved (should be zeros) Bit 1 0 Enable redirect cascade Bit 0 0 Enable real time clock interrupt
00C0h	R/W	DMA channel 4 memory address bytes 1 and 0 (low)
00C2h	R/W	DMA channel 4 transfer count bytes 1 and 0 (low)
00C4h	R/W	DMA channel 5 memory address bytes 1 and 0 (low)
00C6h	R/W	DMA channel 5 transfer count bytes 1 and 0 (low byte)
00C8h	R/W	DMA channel 6 memory address bytes 1 and 0 (low byte)
00CAh	R/W	DMA channel 6 transfer count bytes 1 and 0 (low byte)
00CCh	R/W	DMA channel 7 memory address bytes 1 and 0 (low byte)
00CEh	R/W	DMA channel 7 transfer count bytes 1 and 0 (low byte)
00D0h	R	DMA channels 4–7 status register Bit 7 1 Channel 7 request Bit 6 1 Channel 6 request Bit 5 1 Channel 5 request Bit 4 1 Channel 4 request Bit 3 1 Terminal count on channel 7 Bit 2 1 Terminal count on channel 6 Bit 1 1 Terminal count on channel 5 Bit 0 1 Terminal count on channel 4
00D0h	W	DMA channel 4–7 command register Bit 7 0 DACK sense active low 1 DACK sense active high Bit 6 0 DREQ sense active high 1 DREQ sense active low Bit 5 0 Late write selection 1 Extended write selection Bit 4 0 Fixed priority 1 Rotating priority Bit 3 0 Normal timing 1 Compressed timing Bit 2 0 Enable controller 1 Disable controller Bit 1 0 Disable memory-to-memory transfer 1 Enable memory-to-memory transfer Bit 0 Reserved
00D2h	W	DMA channel 4–7 write request register

I/O Port	Read/ Write	Description
00D4h	W	DMA channel 4–7 write single mask register bit Bits 7–3 Reserved (should be zeros) Bit 2 0 Clear mask bit 1 Set mask bit Bits 1–0 00 Channel 4 select 01 Channel 5 select 10 Channel 6 select 11 Channel 7 select
00D6h	W	DMA channels 4–7 mode register Bits 7–6 00 Demand mode 01 Single mode 10 Block mode 11 Cascade mode Bit 5 0 Address increment select 1 Address decrement select Bit 4 0 Disable Autoinitialization 1 Enable autoinitialization Bits 3–2 00 Verify operation 01 Write to memory 10 Read from memory 11 Reserved Bits 1–0 00 Channel 4 select 01 Channel 5 select 10 Channel 6 select 11 Channel 7 select
00D8h	W	DMA channel 4–7 clear byte pointer flip/flop
00DAh	R	DMA channel 4–7 read temporary register
00DAh	W	DMA channel 4–7 master clear
00DCh	W	DMA channel 4–7 clear mask register
00DEh	W	DMA channel 4–7 write mask register
00F0h		Math coprocessor clear busy latch
00F1h		Math coprocessor reset
00F28–00FFh	R/W	Math coprocessor
0168h–016Fh	R/W	Hard Disk Controller
I/O ports 0170h–0177h are used as a secondary hard disk area. See the definition of I/O ports 01F0–01F7h for the bit definitions.		
0170h	R/W	Hard disk 1 data register
0171h	R	Hard disk 1 error register
0171h	W	Hard disk 1 write precompensation register
0172h	R/W	Hard disk 1 sector count
0173h	R/W	Hard disk 1 sector number
0174h	R/W	Hard disk 1 number of cylinders, low byte
0175h	R/W	Hard disk 1 number of cylinders, high byte
0176h	R/W	Hard disk 1 drive/head register
0177h	R	Hard disk 1 status register
0177h	W	Hard disk 1 command register
0!E8h–01EFh	R/W	Enhanced IDE Disk controller
01F0h	R	Hard disk 0 data register base port (MFM)
01F0h	R	Read Data (EIDE)
01F0h	W	Write Data (EIDE)
01F1h	R	Error Register (EIDE)
01F1h	W	Set features data (EIDE)

I/O Port	Read/ Write	Description
01F1h	R	Hard disk 0 error register (MFM) Diagnostic mode Bits 7–3 Reserved Bits 2–0 Diagnostic mode errors 001 No errors 010 Controller error 011 Sector buffer error 100 ECC device error 101 Control processor error Operation mode Bit 7 0 Block is not bad 1 Bad block detected Bit 6 0 No error 1 Uncorrectable ECC error Bit 5 Reserved Bit 4 0 ID not found 1 ID found Bit 3 Reserved Bit 2 0 Command aborted 1 Command completed Bit 1 0 Track 000 found 1 Track 000 not found Bit 0 0 DAM found (CP-3002 is always 0) 1 DAM not found
01F1h	W	Hard disk 0 write precompensation register (MFM)
01F2h	R/W	Hard disk 0 sector count (MFM)
01F2h	R	Status of sector count (EIDE)
01F2h	W	Write sector count for command setup (EIDE)
01F3h	R/W	Hard disk 0 sector number (MFM)
01F3h	R	Location of starting sector (EIDE)
01F3h	W	Write sector start for command setup (EIDE)
01F4h	R/W	Hard disk 0 number of cylinders, low byte (MFM)
01F4h	R	Low byte of cylinder location (EIDE)
01F4h	W	Write low byte of cylinder location for command setup (EIDE)
01F5h	R/W	Hard disk 0 number of cylinders, high byte (MFM)
01F5h	R	High byte of cylinder location (EIDE)
01F5h	W	Write high byte of cylinder location for command setup (EIDE)
01F6h	R/W	Hard disk 0 drive/head register (MFM) Bit 4 Drive select 0 first hard disk drive 1 Second hard disk drive Bits 3–0 Head select bits
01F6h	R	Head and device selection (EIDE)
01F6h	W	Write device selection and head selection for command setup (EIDE)
01F7h	R	Hard disk 0 status register (MFM) Bit 7 1 Controller is executing a command Bit 6 1 Drive is ready Bit 5 1 Write fault Bit 4 1 Seek complete Bit 3 1 Sector buffer requires servicing Bit 2 1 Disk data read corrected Bit 1 An index. Set to 1 at each disk revolution Bit 0 1 Previous command ended with an error
01F7h	R	Hard disk drive 0 command register (MFM)

I/O Port	Read/ Write	Description
01F7h	R	Device status (EIDE)
01F7h	W	Device command (EIDE)
0200h – 020Fh	R/W	Game controller ports
0201h	R/W	Game port I/O data
020C – 020Dh		Reserved for special use by AMIBIOS.
021Fh		Reserved for special use by AMIBIOS.
0278 – 027Fh		Parallel port 2. See the descriptions of I/O ports 0378h–037Ah for the parallel port bit definitions.
02E8h – 02EFh		Serial port 4. See the descriptions of I/O ports 03F8h–03FFh for the serial port bit definitions.
02F8 – 02FFh		Serial port 2. See the descriptions of I/O ports 03F8h–03FFh for the serial port bit definitions.
0300 – 031Fh		Prototype card
0364 – 0367h		Reserved for special use by AMIBIOS.
036C – 036Fh		Reserved for special use by AMIBIOS.
I/O ports 0372h–0377h are used for the secondary floppy disk controller. See the definitions of I/O ports 03F2h–03F7h for the bit definitions.		
0372h	W	Floppy disk controller 2 digital output register
0374h	R	Floppy disk controller 2 status register
0375h	R/W	Floppy disk controller 2 data register
0376h	R/W	Floppy disk controller 2 control register
0377h	R	Floppy disk controller 2 digital input register
0377h	W	Select register for floppy disk data transfer rate
0378h	R/W	Parallel port 1 data port
0379h	R/W	Parallel port 1 status port Bit 7 0 Busy Bit 6 0 Acknowledge Bit 5 1 Out of paper Bit 4 1 Printer is selected Bit 3 0 Error Bit 2 0 IRQ occurred Bits 1–0 Reserved
037Ah	R/W	Parallel port 1 control port Bits 7–5 Reserved Bit 4 1 Enable IRQ Bit 3 1 Select printer Bit 2 0 Initialize printer Bit 1 1 Automatic line feed Bit 0 1 Strobe
037Bh	R/W	Hercules configuration switch registers Bits 7–2 Not used Bit 1 0 Disable upper 32 KB of graphics mode buffer 1 Enable upper 32 KB of graphics mode buffer at B800:0000h Bit 0 0 Disable graphics mode
03B0–03B3h	R/W	Video registers. See the Video I/O Port tables for additional information about video I/O ports.
03B4h	R/W	MDA CRTC index register
03B5h	R/W	MDA CRTC data registers
03B8h	R/W	MDA mode control register

I/O Port	Read/ Write	Description
03BCh – 03BFh		Parallel port 3. See the descriptions of I/O ports 0378h–037Ah for the parallel port bit definitions.
03C0 – 03CFh		EGA and VGA video subsystem
03C2h	R	CGA input status register
03C3h	R/W	Video subsystem enable
03C4h	R/W	CGA sequencer index register
03C5h	R/W	Other CGA sequencer registers
03CAh	R	CGA feature control register
03D4h	W	Video CRTC index register
03D5h	W	Other CRTC registers
03D8h	R/W	CGA mode control register
03D9h	R/W	CGA palette register
03E8 – 03EFh		Serial port 3. See the descriptions of I/O ports 03F8h–03FFh for the serial port bit definitions.
03F2h	W	Floppy disk controller digital output register Bits 7–6 Reserved. Should be zero. Bit 5 1 Enable motor on floppy drive 1 Bit 4 1 Enable motor on floppy drive 0 Bit 3 1 Enable DMA for floppy drives Bit 2 0 Controller reset Bit 1 Reserved. Should be zero. Bit 0 0 Select floppy drive 0 1 Select floppy drive 1
03F4h	R	Floppy disk controller status register Bit 7 1 data register is ready Bit 6 0 Transfer from the computer to the controller 1 Transfer from the controller to the computer Bit 5 1 Non-DMA mode Bit 4 1 Floppy disk controller busy Bits 3–2 Reserved Bit 1 1 Drive 1 is busy Bit 0 1 Drive 0 is busy
03F5h	R/W	Floppy disk controller data register
03F6h	R	Floppy disk controller control port Bits 7–4 Reserved Bit 3 0 Reduce write current 1 Head select enable Bit 2 0 Disable floppy disk reset 1 Enable floppy disk reset Bit 1 0 Enable floppy disk initialization 1 Disable floppy disk initialization Bit 0 Reserved
03F6h	R	Alternate status (EIDE)
03F6h	W	Device control (EIDE)
03F7h	R/W	Drive address (EIDE)
03F7h	R	Floppy disk controller input register. Bits 7–1 apply to the currently selected floppy drive. Bit 7 1 Floppy disk change line Bit 6 1 Write gate Bit 5 Head select 3/Reduced write current Bit 4 Head select 2 Bit 3 Head select 1 Bit 2 Head select 0 Bit 1 Select drive 1 Bit 0 Select drive 0

I/O Port	Read/ Write	Description
03F7h	W	Floppy disk controller select register for data transfer rate Bits 7–2 Reserved Bits 1–0 00 500 Kbs mode 01 300 Kbs mode 00 250 Kbs mode
03F8h	W	Serial Port 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.
03F8h	R	Serial Port 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.
03F8h	R/W	Serial Port 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.
03F9h	R/W	Divisor Latch, high byte. Bits 7–0 Bits 15–8 of data transmission rate divisor when DLAB is 1.
03F9h	R/W	Serial Port. Interrupt Enable Register. Permits the serial port controller interrupts to enable the chip interrupt output signal. Bits 7–4 Reserved Bit 3 Modem status interrupt enable if set. Bit 2 Receiver line status interrupt enable if set. Bit 1 Transmitter Holding register empty interrupt enable if set. Bit 0 Received data available interrupt enable when DLAB is 0 if set.
03FAh	R	Serial Port 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. Bits 7–3 Reserved Bits 2–1 The pending interrupt with the highest priority. 11 Receiver Line Status Interrupt, priority is the highest. 10 Received Data Available, second in priority. 01 Transmitter Holding Register Empty, third in priority. 00 Modem Status Interrupt, fourth in priority. Bit 0 Interrupt pending if set to logical 0. If logical 1, no interrupt is pending.

I/O Port	Read/ Write	Description
03FBh	R/W	Serial Port 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. 1 2 stop bits if 6, 7, or 8-bit word length 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.
03FCh	R/W	Serial Port Modem Control Register Bits 7–5 Reserved Bit 4 Loopback mode for diagnostic testing of serial port if set. 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 Force OUT2 interrupt if set. Bit 2 Force OUT1 active if set. Bit 1 Force Request To Send active if set. Bit 0 Force Data Terminal Ready active if set.
03FDh	R	Serial Port Line Status Register Bit 7 Reserved Bit 6 Transmitter shift and holding registers empty if set. Bit 5 Transmitter holding register empty if set. The controller is ready to accept a new character to send. Bit 4 Break interrupt if set. 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 Framing error if set. The stop bit that follows the last parity or data bit is zero. Bit 2 Parity error if set. The character has incorrect parity. Bit 1 Overrun error if set. 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 Data Ready if set. A complete incoming character has been received and sent to the receiver buffer register.

I/O Port	Read/ Write	Description
03FEh	R	Serial Port Modem Status Register Bit 7 Data Carrier Detect if set. Bit 6 Ring Indicator if set. Bit 5 Data Set Ready if set. Bit 4 Clear To Send if set. Bit 3 Delta Data Carrier Detect if set. Bit 2 Trailing Edge Ring Indicator if set. Bit 1 Delta Data Set Ready if set. Bit 0 Delta Clear To Send if set.
03FFh	R/W	Serial port 1 scratch register
I/O ports 0401h–04D6h are only used by EISA computers.		
0401h	R/W	DMA channel 0 word count byte 2, high byte
0403h	R/W	DMA channel 1 word count byte 2, high byte
0405h	R/W	DMA channel 2 word count byte 2, high byte
0407h	R/W	DMA channel 3 word count byte 2, high byte
040Ah	W	Extended DMA chaining mode register, channels 0–3 Bits 7–5 Reserved Bit 4 0 Generate IRQ13 1 Generate terminal count Bit 3 0 Do not start chaining 1 Programming complete Bit 2 0 Disable buffer chaining mode (default value) 1 Enable buffer chaining mode Bits 1–0 DMA channel select 00 Channel 0 01 Channel 1 10 Channel 2 11 Channel 3
040Ah	R	Channel interrupt (IRQ13) status register Bits 7–5 Interrupt on channels 7–5 Bit 4 Reserved Bits 3–0 Interrupt on channels 3–0
040Bh	W	DMA extended mode register for channels 0–3 Bit 7 0 Enable stop register Bit 6 0 Terminal count is an output for this channel (default) Bits 5–4 DMA cycle timing 00 ISA-compatible (default) 01 Type A timing mode 10 Type B timing mode 11 DMA burst mode Bits 3–2 Address mode 00 8-bit I/O, count by bytes (default) 01 16-bit I/O, count by words, address-shifted 10 32-bit I/O, count by bytes 11 16-bit I/O, count by bytes Bits 1–0 DMA channel select

I/O Port	Read/ Write	Description
0461h	R/W	Extended NMI status and control register Bit 7 1 NI pending from fail-safe timer (read only) Bit 6 1 NMI pending from bus timeout NMI status (read only) Bit 5 1 NMI pending (read only) Bit 4 Reserved Bit 3 1 Bus timeout NMI enable (R/W) Bit 2 1 Fail-safe NMI enable (R/W) Bit 1 1 NMI I/O port enable (R/W) Bit 0 RSTDRV. Bus reset (R/W) 0 Normal bus reset 1 Reset bus asserted
0462h	W	Software NMI register. Writing to this register causes an NMI if NMIs are enabled.
0464h	R	Bus master status latch enable register (slots 1–8). Identifies the last bus master to control the bus. Bit 7 0 Slot 8 Bit 6 0 Slot 7 Bit 5 0 Slot 6 Bit 4 0 Slot 5 Bit 3 0 Slot 4 Bit 2 0 Slot 3 Bit 1 0 Slot 2 Bit 0 0 Slot 1
0465h	R	Bus master status latch enable register (slots 9–16). Identifies the last bus master to control the bus. Bit 7 0 Slot 16 Bit 6 0 Slot 15 Bit 5 0 Slot 14 Bit 4 0 Slot 13 Bit 3 0 Slot 12 Bit 2 0 Slot 11 Bit 1 0 Slot 10 Bit 0 0 Slot 9
0481h	R/W	DMA channel 3 address byte 3, high byte
0483h	R/W	DMA channel 2 address byte 3, high byte
0485h	R/W	DMA channel 1 address byte 3, high byte
0487h	R/W	DMA channel 0 address byte 3, high byte
0489h	R/W	DMA channel 6 address byte 3, high byte
048Bh	R/W	DMA channel 7 address byte 3, high byte
048Dh	R/W	DMA channel 5 address byte 3, high byte
04C6h	R/W	DMA channel 5 word count byte 2, high byte
04CAh	R/W	DMA channel 6 word count byte 2, high byte
04CEh	R/W	DMA channel 7 word count byte 2, high byte
04D0h	W	IRQ0–IRQ7 interrupt edge/level registers Bit 7 1 IRQ7 is level-sensitive Bit 6 1 IRQ6 is level-sensitive Bit 5 1 IRQ5 is level-sensitive Bit 4 1 IRQ4 is level-sensitive Bit 3 1 IRQ3 is level-sensitive Bits 2–0 Reserved

I/O Port	Read/ Write	Description
04D1h	W	IRQ8–IRQ15 interrupt edge/level registers Bit 7 1 IRQ15 is level-sensitive Bit 6 1 IRQ14 is level-sensitive Bit 5 Reserved Bit 4 1 IRQ12 is level-sensitive Bit 3 1 IRQ11 is level-sensitive Bit 2 1 IRQ10 is level-sensitive Bit 1 1 IRQ9 is level-sensitive Bit 0 Reserved
04D4h	R	Chaining mode status register Bits 7–5 1 Enable Channels 7–5 Bit 4 Reserved Bits 3–0 1 Enable Channels 3–0
04D4h	W	Extended DMA chaining mode register, channels 4–7 Bits 7–5 Reserved Bit 4 0 Generate IRQ13 1 Generate terminal count Bit 3 0 Do not start chaining 1 Programming complete Bit 2 0 Disable buffer chaining mode (default value) Bits 1–0 Select DMA channel 00 Channel 4 01 Channel 5 10 Channel 6 11 Channel 7
04D6h	W	DMA extended mode register for channels 4–7. See I/O port 040Bh for the bit settings.
0500h-07FFh		A copy of all I/O port assignments from 0110h-03FFh is placed here in EISA computers.
0800h–08FFh	R/W	I/O port access registers for EISA CMOS RAM
0900h-0BFFh		A copy of all I/O port assignments from 0110h-03FFh is placed here in EISA computers.
0C00h	R/W	Page register to write to SRAM or I/O
0C80h	R/W	EISA motherboard ID Bit 7 Reserved. Should be zero. Bits 6–2 First letter of manufacturer code. Bits 1–0 First two bits of second letter of manufacturer code
0C81h	R/W	EISA motherboard ID Bits 7–5 Remaining 3 bits of second letter of manufacturer code. Bits 4–0 Third letter of manufacturer code.
0C82h	R/W	EISA motherboard ID Bits 7–0 Reserved for use by manufacturer. Often used for manufacturer's product number. American Megatrends EISA motherboards have the serial number in BCD.

I/O Port	Read/ Write	Description
0C83h	R/W	EISA motherboard ID Bits 7–3 Product Revision Number assigned by EISA motherboard manufacturer. Bits 2–0 EISA Bus Version (initial version is 001) 001 is currently the only standard value defined, but, in practice, EISA motherboard and adapter card manufacturers have been using this field for their own purposes. In American Megatrends EISA motherboards: Bits 7–4 EISA Configuration file revision number Bit 3 Reserved Bits 2–0 EISA bus version
0C84h	R/W	American Megatrends EISA Motherboard ID data Bits 7–3 Schematic release (for internal use only) Bits 2–0 PCB release (for internal use only)
0C85h	R/W	American Megatrends EISA Motherboard ID data CPU speed in MHz (in BCD)
0D00h-0FFFh		A copy of all I/O port assignments from 0110h-03FFh is placed here in EISA computers.
EISA Adapter Card Ports In the following rows, *n* can be 1h – Fh and represents EISA expansion slots 1 – 15.		
*n*000-*n*0FFh		EISA expansion slot *n*.
*n*100-*n*3FFh		A copy of all I/O port assignments from 0100h – 03FFh.
*n4*000-4FFh		EISA expansion slot *n*.
*n*500-*n*7FFh		A copy of all I/O port assignments from 0100h – 03FFh.
*n*800-*n*8FFh		EISA expansion slot *n*.
*n*900-BFFh		A copy of all I/O port assignments from 0100h – 03FFh.
*N*C00-CFFh		EISA expansion slot *n*.
*n*D00-FFFh		A copy of all I/O port assignments from 0100h – 03FFh.
*n*000-*n*0FFh		EISA expansion slot *n*.
*n*100-*n*3FFh		A copy of all I/O port assignments from 0100h – 03FFh.
*n4*00h-4FFh		EISA expansion slot *n*.
*n*500-*n*7FFh		A copy of all I/O port assignments from 0100h – 03FFh.
*n*800-*n*8FFh		EISA expansion slot *n*.
*n*900-BFFh		A copy of all I/O port assignments from 0100h – 03FFh.
*N*C00-CFFh		EISA expansion slot *n*.
*n*D00-FFFh		A copy of all I/O port assignments from 0100h – 03FFh.

EISA Adapter Card Compressed ID

The following I/O ports are used by EISA adapter cards in the slot number corresponding to *n (1h – Fh).*

I/O Port	Description of Contents
0*n*C80h	Bit 7 Reserved. Should be zero. Bits 6–2 First letter of manufacturer code. Bits 1–0 First two bits of second letter of manufacturer code.
0*n*C81h	Bits 7–5 Remaining 3 bits of second letter of manufacturer code. Bits 4–0 Third letter of manufacturer code.
0*n*C82h	Bits 7–4 First hex digit of product number. Bits 3–0 Second hex digit of product number.
0*n*C83h	Bits 7–4 First hex digit of revision number. Bits 3–0 Second hex digit of revision number.
0*n*C84h	EISA Adapter Card Control Register Bits 7–3 Reserved. Should be zero. Bit 2 IOCHKRST (write/only) 0 Normal Operation. 1 Reset the adapter after sending an Active High Pulse. Bit 1 IOCHKERR (read/only) 0 No I/O error pending. 1 I/O error detected by adapter card. Bit 0 ENABLE (Read/Write) 0 Adapter Card Disable. 1 Adapter Card Enable.

Monochrome Video I/O Ports

Some I/O devices (such as the video controllers) use system memory addresses as well as I/O port addresses. This technique (memory-mapped I/O) makes the CPU think the devices are part of system memory. Memory-mapped devices are easier to program because they permit more flexible memory instructions. The video ports used in the MDA and CGA video standard are listed below. EGA and VGA standards use I/O ports much more extensively, but since the system BIOS does not perform EGA, VGA, or XGA video, they are not discussed here.

MDA I/O Ports The 6845 CRTC index register is mapped to I/O port 03B4h. The index value in port 03B4h controls the register that appears at I/O port 03B5h. The 68345 mode control register is accessed directly through I/O port 03B8h.

I/O Port	Read/ Write	Description
03B4h	W	CRTC index register
03B5h	W	Index 00h Horizontal total Index 01h Horizontal displayed Index 02h Horizontal sync position Index 03h Horizontal sync pulse width Index 04h Vertical total Index 05h Vertical displayed Index 06h Vertical sync position Index 07h Vertical sync pulse width Index 08h Interlace mode Index 09h Maximum scan lines Index 0Ah Cursor start Index 0Bh Cursor end Index 0Ch Start address, high byte Index 0Dh Start address, low byte Index 0Eh Cursor location, high byte Index 0Fh Cursor location, low byte Index 10h Light pen, high byte Index 11h Light pen, low byte
03B8h	W	Mode control register
03BAh	R	CRT status register

CGA Video I/O Ports

CGA I/O Ports The 6845 CRTC index register is mapped to I/O port 03D4h. The value written to 03D4h controls the register that appears in I/O port 03D5h.

I/O Port	Read/ Write	Description
03D4h	W	CRTC index register
03D5h	W	Index 00h Horizontal total Index 01h Horizontal displayed Index 02h Horizontal sync position Index 03h Horizontal sync pulse width Index 04h Vertical total Index 05h Vertical displayed Index 06h Vertical sync position Index 07h Vertical sync pulse width Index 08h Interleaved mode Index 09h Maximum scan lines Index 0Ah Cursor start Index 0Bh Cursor end Index 0Ch Start address, high byte Index 0Dh Start address, low byte Index 0Eh Cursor location, high byte Index 0Fh Cursor location, low byte Index 10h Light pen, high byte Index 11h Light pen, low byte
03D8h	W	Mode control register
03D9h	W	Palette register
03DAh	R	CRT status register
03DBh	W	Clear light pen latch
03DCh	W	Preset light pen latch

7 Power On Self Test

The first BIOS routine that executes is called the Power On Self Test (POST). POST is performed every time the computer is powered on. POST performs diagnostic tests on system memory and key computer components. It also initializes BIOS configuration tables. It then boots the operating system.

Starting POST You can start POST in several different ways:

Starting POST	What the BIOS Does
Turn the computer on.	Jumps to the address pointed to by the processor reset vector (FFFF0h). All POST tests and initializations are then executed. If successful, POST calls INT 19h Bootstrap Loader.
Press the reset button.	Jumps to the address pointed to by the processor reset vector (FFFF0h). All POST tests and initializations are then executed. If successful, POST calls INT 19h Bootstrap Loader.
Press <Ctrl> <Alt> <Del>.	The INT 09h keyboard hardware interrupt service transfers control to POST. POST does not test memory above 64 KB, but all other tests and initializations are performed. POST then calls INT 19h Bootstrap Loader.

POST Phases

Before POST Enables NMIs NMI and I/O checks are disabled. Before NMIs are enabled, the BIOS POST:

Step	Description
1	Writes data in all motherboard and adapter card memory locations to establish that parity is good.
2	Enables the onboard and 32-bit slot memory parity checks by writing to I/O port 61h with data bit 2 set to zero.
3	Enables the I/O channel check signal by writing to I/O Port 61h with data bit 3 set to zero.

POST Functions

POST usually performs the following tests in the following order. In some AMIBIOS, some tests are performed in a slightly different sequence. The errors that can be generated by the test are listed below.

If the contents of 0Fh at CMOS RAM location 0Fh (Shutdown Byte) is 00h, POST performs all tests and initializations.

Processor Register Test The following values are loaded consecutively into all registers: 0555h, 0AAAh, 0CCCh, and 0F0Fh. If any register does not retain any of these values, Beep Code 5 is issued.

ROM BIOS Checksum Test A checksum is performed on the system BIOS. If it is incorrect, Beep Code 9 is issued.

Keyboard Controller Test The BIOS issues a keyboard controller BAT command. If the response is not 55h, Beep Code 6 is issued.

CMOS Shutdown Register Test The BIOS writes the values 55h, then AAh to 0Fh in CMOS RAM. If there is an error, *CMOS not operational* is displayed. The computer halts and must be rebooted. Replace the battery on error.

System Timer Test The *CH-1 timer error* or *CH-2 timer error* is displayed if this test fails on channel 1 or 2 of the timer. Beep Code 4 is issued if timer channel 1 does not work.

Memory Refresh Test Timer channels 0 and 1 are tested. Beep Code 1 is issued if either channel does not work.

Base 64 KB Test An address test, sequential read and write, and random read and write test are performed on the first 64 KB of RAM. Beep Code 3 is issued if any errors (including parity errors) occur.

Cont'd

POST Functions, Continued

Cache Memory Test Memory reads are performed with secondary cache memory enabled. Then memory tests are performed with cache disabled. If cache memory is not performing as expected, *Cache Memory bad, do not enable cache* is displayed.

CMOS RAM Battery Test POST reads the CMOS RAM status register (offset 8Dh) to see if the battery is on. It then reads CMOS diagnostic data and calculates CMOS RAM checksums. The following error messages may be generated: *CMOS battery state low*, *CMOS system option not set*, or *CMOS checksum error*.

Display Verification POST does a vertical and horizontal retrace and a sequential read and write of 4 KB in different display modes. Beep Code 8 is issued if there is any error. Other messages that may be issued are: *Display switch setting not proper* or *CMOS display type mismatch*.

Enter Protected Mode This test issues INT 15h Function 89h. *8042 Gate-A20 error, system halted* is displayed if it cannot get to protected mode.

Address Line Test A test pattern is written to both conventional and extended memory. Beep Code 3 is issued if an error occurs.

Conventional and Extended Memory Test Zeros are written to extended and conventional memory unless <Esc> is pressed. Sequential and random read and write tests are performed. The amount of memory tested is displayed. Beep Code 3 or 7 is issued if there are errors.

DMA Controller Test Several patterns are written to DMA page registers 80h through 8Fh and then DMA registers 0 through 7. The following errors may be generated: *DMA error, system halted, DMA #1 error, system halted,* or *DMA #2 error, system halted*.

Cont'd

Keyboard Test The keyboard self-test command is issued. A stuck key check is performed. The keyboard interface test is then performed. Possible errors are: *Keyboard error* or *KB/Interface error*.

System Configuration Verification The floppy and hard disk areas are initialized and a Seek is performed on drive A: or drive C:. Possible errors are: *FDD controller failure*, *HDD controller error, C: drive failure*, or *D: drive failure*. The memory size is verified. *CMOS memory size mismatch* is generated if there is an error. Option ROMs are checked and the timer data area is initialized. Possible errors: *CMOS time & date not set*. The parallel and serial ports are configured. *Keyboard is locked* is displayed if the keyboard is locked.

POST Error Handling

A primary POST function is to find any conditions that prevent proper operation. POST diagnostic routines look for system errors and report them in the following manner.

If...	Then...
the error occurs before the display is initialized.	a series of beeps sound. Beep codes indicate that a fatal error has occurred. AMIBIOS Beep Codes are described in Appendix A.
the error occurs after the display is initialized,	the error message is displayed. Displayed BIOS messages are described in Appendix A. A prompt to press <F1> also appears on the screen.

Beep Codes AMIBIOS beeps when it cannot successfully configure the system display monitor. Beeps indicate a serious problem. All errors except Beep Code 8 are fatal. Fatal errors halt the computer.

Displayed Errors If POST is able to configure the system display, it can display errors on the screen. In general, these errors are not as serious as beep codes.

POST Diagnostic Codes POST also produces a series of diagnostic codes that indicate specific milestones in POST. AMIBIOS POST codes are accessed via I/O Port 0080h.

POST Memory Test

Normally, the only visible POST routine is the memory test. A sample screen follows:

```
AMIBIOS (C) 1997 American Megatrends Inc.

xxxxx KB OK

Hit <DEL> if you want to run SETUP




(C) American Megatrends Inc.
XX-XXXX-XXXXXX-XXXXXXXX-XXXXXX-XXXX-X
```

BIOS Identification String An Identification String is displayed at the left bottom corner of the screen, below the copyright message. In some AMIBIOS, you can press <Ins> during system boot to display two additional BIOS Identification Strings. These strings display the options in AMIBIOS. Appendix B lists the contents of the Identification Strings.

Freezing the Screen Make sure *Wait for <F1> If any Error* in Advanced Setup is *Enabled* before using the following procedure. Freeze the screen by powering on the computer and holding a key down on the keyboard. Copy the BIOS Identification Strings on a piece of paper and refer to Appendix B. Press <F1> to continue the boot process.

POST Memory Test The following message is displayed after POST completes:

```
Hit <DEL> if you want to run SETUP
```

Press <Del> to access the Setup utility.

AMIBIOS Configuration Summary Screen

AMIBIOS displays the BIOS Configuration Summary screen (see the sample screen below) when the POST routines complete successfully. This screen may be slightly different in your computer.

```
System Configuration (C) Copyright 1985-1991 American Megatrends Inc.

Main Processor          : Pentium       Base Memory Size    : 640 KB
Numeric Coprocessor     : Present       Ext. Memory Size    : 7808 KB
Floppy Drive A:         : 1.2 MB 5¼     Pri Master          :
Floppy Drive B:         : 1.44 MB 3½    Pri Slave           : None
Display Type:           : VGA or EGA    Serial Port(s)      : 3F8, 3E8
ROM-BIOS Date:          : 7/15/95       Parallel Port(s)    : 378
```

ROM Extensions

An option ROM is an optional extension to the system BIOS. Extension ROMs can either replace existing ROM BIOS device service routines or add new service routines. Examples of ROM extensions include an ESDI hard disk drive BIOS or a SCSI BIOS. POST detects ROM extensions and allows them to initialize themselves and test and initialize the devices they control. The ROM extensions then return control to POST.

Handling ROM Extensions By convention, ROM extensions can appear on any 2 KB boundary between C0000h and FFFFFh. POST searches RAM from C0000h through FFFFFh in 2 KB increments for ROM extensions. A ROM extension at E0000h must be 64 KB long.

Identifying a ROM Extension ROM extensions must have a standard header. The information in the header indirectly identifies the type of device and its use. The following table lists the most important parts of a ROM extension header.

Offset	Contents	Description
0	55AAh	ROM extension header identifier.
1		Length code. The length in 512-byte (½K) units. A 64 KB ROM extension has a length code of 128.
2		Instruction. Normally, this field has a one-byte FAR RETURN instruction or a three-byte JMP instruction.
5	Varies	The header contains a number of other fields.
Last	00h	Usually 00h.

System Boot

The last action performed by AMIBIOS POST is to boot the operating system by invoking INT 19h, as follows:

If...	and...	then...
The boot sequence is 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 boot sequence is A:, C:,	drives A: and C: have no bootable disk,	INT 19h invokes INT 18h.
The boot sequence is A:, C:,	the floppy disk in drive A: is not bootable, but drive C: is bootable,	INT 19h reads the boot sector on drive A: and places its contents at 7C00h.
The boot sequence is C:, A:,	a boot sector is found on drive C:,	INT 19h reads the boot sector on drive C: and places its contents at 7C00h.
The boot sequence is C:, A:,	drive C: has no boot sector (the hard disk is not formatted for boot) but drive A: does,	INT 19h reads the boot sector on drive A: and places its contents at 7C00h.
The boot sequence is C:, A:,	neither drive C: or A: has a boot sector,	INT 19h invokes INT 18h.

INT 18h

INT 18h is called if INT 19h does not find a boot sector. INT 18h can also be vectored to a routine that takes over the boot process — to permit booting over a network, for example. INT 18h usually points to a routine that displays

```
No Boot Device Available
```

POST Checkpoint Codes

POST is performed by the BIOS when the computer is reset or rebooted. POST tests and initializes key components. When a POST routine completes, a code is written to I/O port address 80h. Display this code by attaching diagnostic equipment to I/O port address 0080h. The following POST checkpoint codes are valid for all AMIBIOS with a core BIOS date of 7/15/95 and a version number of 6.14 or later.

Uncompressed Initialization Codes The uncompressed initialization checkpoint codes (in order of execution) are:

Code	Description
D0h	The NMI is disabled. Power on delay is starting. Next, the initialization code checksum will be verified.
D1h	Initializing the DMA controller, performing the keyboard controller BAT test, starting memory refresh, and entering 4 GB flat mode next.
D3h	Starting memory sizing next.
D4h	Returning to real mode. Executing any OEM patches and setting the stack next.
D5h	Passing control to the uncompressed code in shadow RAM at E000:0000h.The initialization code is copied to segment 0 and control will be transferred to segment 0.
D6h	Control is in segment 0. Next, checking if <Ctrl> <Home> was pressed and verifying the system BIOS checksum. If either <Ctrl> <Home> was pressed or the system BIOS checksum is bad, next will go to checkpoint code E0h. Otherwise, going to checkpoint code D7h.
D7h	Passing control to the interface module next.
D8h	The main system BIOS runtime code will be decompressed next.
D9h	Passing control to the main system BIOS in shadow RAM next.

Cont'd

Bootblock Recovery Codes The bootblock recovery checkpoint codes are (in order of execution):

Code	Description
E0h	The onboard floppy controller if available is initialized. Next, beginning the base 512 KB memory test.
E1h	Initializing the interrupt vector table next.
E2h	Initializing the DMA and Interrupt controllers next.
E6h	Enabling the floppy drive controller and Timer IRQs. Enabling internal cache memory.
EDh	Initializing the floppy drive.
EEh	Looking for a floppy diskette in drive A:. Reading the first sector of the diskette.
EFh	A read error occurred while reading the floppy drive in drive A:.
F0h	Next, searching for the AMIBOOT.ROM file in the root directory.
F1h	The AMIBOOT.ROM file is not in the root directory.
F2h	Next, reading and analyzing the floppy diskette FAT to find the clusters occupied by the AMIBOOT.ROM file.
F3h	Next, reading the AMIBOOT.ROM file, cluster by cluster.
F4h	The AMIBOOT.ROM file is not the correct size.
F5h	Next, disabling internal cache memory.
FBh	Next, detecting the type of flash ROM.
FCh	Next, erasing the flash ROM.
FDh	Next, programming the flash ROM.
FFh	Flash ROM programming was successful. Next, restarting the system BIOS.

Cont’d

POST Checkpoint Codes, Continued

Uncompressed Initialization Codes The following runtime checkpoint codes are in order of execution. These codes are uncompressed in F0000h shadow RAM.

Code	Description
03h	The NMI is disabled. Next, checking for a soft reset or a power on condition.
05h	The BIOS stack has been built. Next, disabling cache memory.
06h	Uncompressing the POST code next.
07h	Next, initializing the CPU and the CPU data area.
08h	The CMOS checksum calculation is done next.
0Bh	Next, performing any required initialization before the keyboard BAT command is issued.
0Ch	The keyboard controller input buffer is free. Next, issuing the BAT command to the keyboard controller.
0Eh	The keyboard controller BAT command result has been verified. Next, performing any necessary initialization after the keyboard controller BAT command test.
0Fh	The initialization after the keyboard controller BAT command test is done. The keyboard command byte is written next.
10h	The keyboard controller command byte is written. Next, issuing the Pin 23 and 24 blocking and unblocking commands.
11h	Next, checking if the <End or <Ins> keys were pressed during power on. Initializing CMOS RAM if the *Initialize CMOS RAM in every boot* AMIBIOS POST option was set in AMIBCP or the <End> key was pressed.
12h	Next, disabling DMA controllers 1 and 2 and interrupt controllers 1 and 2.
13h	The video display has been disabled. Port B has been initialized. Next, initializing the chipset.
14h	The 8254 timer test will begin next.
19h	The 8254 timer test is over. Starting the memory refresh test next.
1Ah	The memory refresh line is toggling. Checking the 15 second on/off time next.
23h	Reading the 8042 input port and disabling the MEGAKEY Green PC feature next. Making the BIOS code segment writable and performing any necessary configuration before initializing the interrupt vectors.
24h	The configuration required before interrupt vector initialization has completed. Interrupt vector initialization is about to begin.
25h	Interrupt vector initialization is done. Clearing the password if the POST DIAG switch is on.
27h	Any initialization before setting video mode will be done next.
28h	Initialization before setting the video mode is complete. Configuring the monochrome mode and color mode settings next.
2Ah	Bus initialization system, static, output devices will be done next, if present.
2Bh	Passing control to the video ROM to perform any required configuration before the video ROM test.
2Ch	All necessary processing before passing control to the video ROM is done. Looking for the video ROM next and passing control to it.
2Dh	The video ROM has returned control to BIOS POST. Performing any required processing after the video ROM had control.
2Eh	Completed post-video ROM test processing. If the EGA/VGA controller is not found, performing the display memory read/write test next.
2Fh	The EGA/VGA controller was not found. The display memory read/write test is about to begin.
30h	The display memory read/write test passed. Look for retrace checking next.

Code	Description
31h	The display memory read/write test or retrace checking failed. Performing the alternate display memory read/write test next.
32h	The alternate display memory read/write test passed. Looking for alternate display retrace checking next.
34h	Video display checking is over. Setting the display mode next.
37h	The display mode is set. Displaying the power on message next.
38h	Initializing the bus input, IPL, and general devices next, if present.
39h	Displaying bus initialization error messages.
3Ah	The new cursor position has been read and saved. Displaying the *Hit <DEL>* message next.
40h	Preparing the descriptor tables next.
42h	The descriptor tables are prepared. Entering protected mode for the memory test next.
43h	Entered protected mode. Enabling interrupts for diagnostics mode next.
44h	Interrupts enabled if the diagnostics switch is on. Initializing data to check memory wraparound at 0:0 next.
45h	Data initialized. Checking for memory wraparound at 0:0 and finding the total system memory size next.
46h	The memory wraparound test has completed. The memory size calculation has been completed. Writing patterns to test memory next.
47h	The memory pattern has been written to extended memory. Writing patterns to the base 640 KB memory next.
48h	Patterns written in base memory. Determining the amount of memory below 1 MB next.
49h	The amount of memory below 1 MB has been found and verified. Determining the amount of memory above 1 MB memory next.
4Bh	The amount of memory above 1 MB has been found and verified. Checking for a soft reset and clearing the memory below 1 MB for the soft reset next. If this is a power on situation, going to checkpoint 4Eh next.
4Ch	The memory below 1 MB has been cleared via a soft reset. Clearing the memory above 1 MB next.
4Dh	The memory above 1 MB has been cleared via a soft reset. Saving the memory size next. Going to checkpoint 52h next.
4Eh	The memory test started, but not as the result of a soft reset. Displaying the first 64 KB memory size next.
4Fh	The memory size display has started. The display is updated during the memory test. Performing the sequential and random memory test next.
50h	The memory below 1 MB has been tested and initialized. Adjusting the displayed memory size for relocation and shadowing next.
51h	The memory size display was adjusted for relocation and shadowing. Testing the memory above 1 MB next.
52h	The memory above 1 MB has been tested and initialized. Saving the memory size information next.
53h	The memory size information and the CPU registers are saved. Entering real mode next.
54h	Shutdown was successful. The CPU is in real mode. Disabling the Gate A20 line, parity, and the NMI next.
57h	The A20 address line, parity, and the NMI are disabled. Adjusting the memory size depending on relocation and shadowing next.
58h	The memory size was adjusted for relocation and shadowing. Clearing the *Hit <DEL>* message next.
59h	The *Hit <DEL>* message is cleared. The *<WAIT...>* message is displayed. Starting the DMA and interrupt controller test next.
60h	The DMA page register test passed. Performing the DMA Controller 1 base register test next.
62h	The DMA controller 1 base register test passed. Performing the DMA controller 2 base register test next.

Code	Description
65h	The DMA controller 2 base register test passed. Programming DMA controllers 1 and 2 next.
66h	Completed programming DMA controllers 1 and 2. Initializing the 8259 interrupt controller next.
7Fh	Extended NMI source enabling is in progress.
80h	The keyboard test has started. Clearing the output buffer and checking for stuck keys. Issuing the keyboard reset command next.
81h	A keyboard reset error or stuck key was found. Issuing the keyboard controller interface test command next.
82h	The keyboard controller interface test completed. Writing the command byte and initializing the circular buffer next.
83h	The command byte was written and global data initialization has completed. Checking for a locked key next.
84h	Locked key checking is over. Checking for a memory size mismatch with CMOS RAM data next.
85h	The memory size check is done. Displaying a soft error and checking for a password or bypassing WINBIOS Setup next.
86h	The password was checked. Performing any required programming before WINBIOS Setup next.
87h	The programming before WINBIOS Setup has completed. Uncompressing the WINBIOS Setup code and executing the AMIBIOS Setup or WINBIOS Setup utility next.
88h	Returned from WINBIOS Setup and cleared the screen. Performing any necessary programming after WINBIOS Setup next.
89h	The programming after WINBIOS Setup has completed. Displaying the power on screen message next.
8Bh	The first screen message has been displayed. The <*WAIT...*> message is displayed. Performing the PS/2 mouse check and extended BIOS data area allocation check next.
8Ch	Programming the WINBIOS Setup options next.
8Dh	The WINBIOS Setup options are programmed. Resetting the hard disk controller next.
8Fh	The hard disk controller has been reset. Configuring the floppy drive controller next.
91h	The floppy drive controller has been configured. Configuring the hard disk drive controller next.
95h	Initializing the bus option ROMs from C800 next.
96h	Initializing before passing control to the option ROM at C800.
97h	Initialization before the C800 option ROM gains control has completed. The option ROM check is next.
98h	The option ROM had control and has now returned control to BIOS POST. Performing any required processing after the option ROM returned control.
99h	Any initialization required after the option ROM test has completed. Configuring the timer data area and printer base address next.
9Ah	Set the timer and printer base addresses. Setting the RS-232 base address next.
9Bh	Returned after setting the RS-232 base address. Performing any required initialization before the Coprocessor test next.
9Ch	Required initialization before the Coprocessor test is over. Initializing the Coprocessor next.
9Dh	Coprocessor initialized. Performing any required initialization after the Coprocessor test next.
9Eh	Initialization after the Coprocessor test is complete. Checking the extended keyboard, keyboard ID, and Num Lock key next. Issuing the keyboard ID command next.
A2h	Displaying any soft errors next.

Code	Description
A3h	The soft error display has completed. Setting the keyboard typematic rate next.
A4h	The keyboard typematic rate is set. Programming the memory wait states next.
A5h	Memory wait state programming is over. Clearing the screen and enabling parity and the NMI next.
A7h	NMI and parity enabled. Performing any initialization required before passing control to the option ROM at E000 next.
A8h	Initialization before passing control to the option ROM at E000h completed. Passing control to the option ROM at E000h next.
A9h	Returned from option ROM at E000h control. Performing any initialization required after the E000 option ROM had control next.
AAh	Initialization after E000 option ROM control has completed. Displaying the system configuration next.
ABh	Building the multiprocessor table, if necessary.
ACh	Uncompressing the DMI data and initializing DMI POST next.
B0h	The system configuration is displayed.
B1h	Copying any code to specific areas.
00h	Code copying to specific areas is done. Passing control to INT 19h boot loader next.

Bus Checkpoint Codes The system BIOS passes control to buses at:

Code	Description
2Ah	Initializing the different bus system, static, and output devices, if present.
38h	Initialized bus input, IPL, and general devices, if present.
39h	Displaying bus initialization error messages, if any.
95h	Initializing bus option ROMs from C8000h through D8000h.

Additional Bus Checkpoints While control is in the bus routines, additional checkpoints are output to I/O port address 0080h as word to identify the routines being executed. These are word checkpoints. The low byte of checkpoint is the system BIOS checkpoint where control is passed to bus routines. The high byte of the bus checkpoints includes:

Bits	Description
7-4	0000 Function 0. Disable all devices on the bus. 0001 Function 1. Initialize static devices on the bus. 0010 Function 2. Initialize output devices on the bus. 0011 Function 3. Initialize input devices on the bus. 0100 Function 4. Initialize IPL devices on the bus. 0101 Function 5. Initiate general devices on the bus. 0110 Function 6. Initialize error reporting on the bus. 0111 Function 7. Initialize add-on ROMs for all buses.
3-0	Specify the bus 0 Generic DIM Device Initialization Manager. 1 Onboard System devices. 2 ISA devices. 3 EISA devices. 4 ISA PnP devices. 5 PCI devices.

8 Using Interrupts

Overview

The interrupt is the method used in ISA and EISA computers to access BIOS services. Both software programs and peripheral devices use interrupts:

- hardware peripheral devices use interrupts to report an event or request that an action be performed.
- software programs use the INT mnemonic to request certain action from a peripheral device.

What an Interrupt Does An interrupt essentially stops other CPU operations. The number specified with software interrupts instructs the BIOS to perform an operation using a specific peripheral device.

Requesting a Software Interrupt Invoke a software interrupt from any assembler language program. Place the interrupt number after the assembler mnemonic INT. The CPU executes the instructions identified by the interrupt number when it finds an INT mnemonic. These instructions make up an interrupt service routine (or device service routine).

CPU Interrupt Handling When the CPU receives an interrupt signal (if it is a software interrupt, it also contains an interrupt number), it usually stops all other activity and activates a subroutine stored in system memory. These subroutines are either interrupt service routines (ISRs) or device service routines (DSRs). The ISR or DSR is keyed to the interrupt number (either software or hardware). The ISR or DSR contains the code that executes the task or routine requested by the INT mnemonic and interrupt number.

Using Registers to Further Define the Interrupt Before a software interrupt is invoked, special prespecified codes and parameters are moved into CPU registers to specify the operation the interrupt routine is to perform. The interrupt routine output is usually returned in the CPU registers or flags.

Types of Interrupts

Interrupt Type	Description	Range
Processor	Generated or processed by the CPU.	00h – 04h
Hardware	Generated by hardware devices. Eight of these are hardwired to either the CPU or the motherboard. IRQs 2, 8, 9, and 11 – 15 are reserved.	08h – 0Fh 70h – 77h
Software	Handled by the BIOS. 05h, 10h through 1Ah, and 40h, 41h, 42h, 43h, 46h, and 4Ah are reserved. The rest are available for revectoring to programmer-written routines.	05h – 07h 10h – 1Fh 40h – 5Fh
DOS	Only available when DOS is active. INTs 20h through 3Fh are reserved for DOS.	20h – 3Fh
BASIC	Reserved	80h – BFh
User	INT 67h is used for EMS. All others available for revectoring to user-written routines.	60h – 6Fh

Interrupts Not Discussed In this manual, we are concerned with only the following types of interrupts. And only certain software interrupts are discussed in detail.

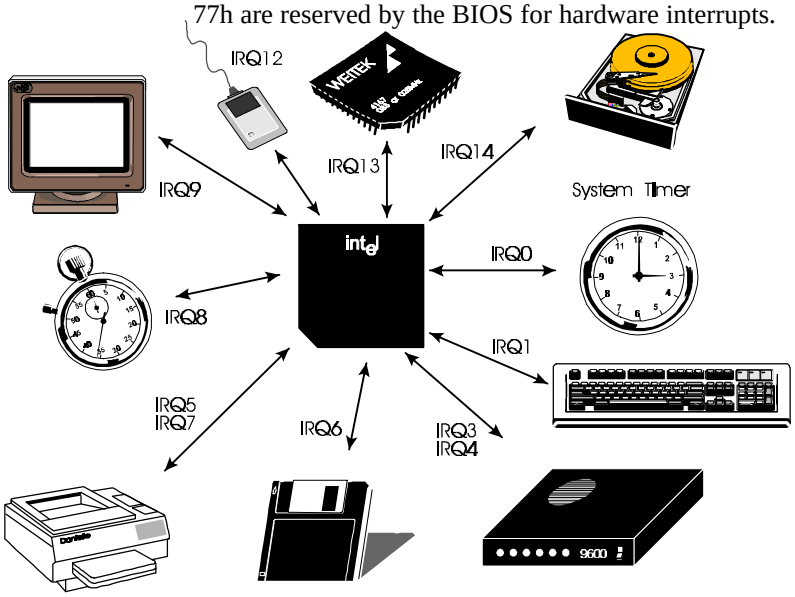
Interrupt Type	Description
Processor (Logical)	Generated by the CPU as a result of an unusual program result, such as a divide by zero or a nonmaskable interrupt.
Hardware	Generated by the computer circuits and managed by the Programmable Interrupt Controller (the Intel 8259).
Software	BIOS interrupts invoked by a system software or application software program or used internally by the BIOS.

Processor Interrupts These interrupts are invoked by the processor as a result of an unusual program result. Interrupts 00h – 04h are reserved for the processor. For example, INT 00h means a program tried to divide a value by 0. The processor generates this interrupt and halts processing.

Cont'd

Types of Interrupts, Continued

Hardware Interrupts When a hardware device or program needs the processor, it sends a signal or instruction to the processor, requesting a certain service or task. After the interrupt handler performs its task, computer activities continue at the same point the interrupt occurred. Hardware interrupts are invoked by peripheral devices by setting an assigned Interrupt Request (IRQ) line. In EISA systems, IRQs can be assigned via the EISA Configuration Utility (ECU). For example, every time a key is pressed on the keyboard, the keyboard hardware generates a hardware interrupt (IRQ). Hardware interrupts are vectored to Interrupt Service Routines (ISRs) that generally reside in the BIOS. INTs 08h – 0Fh and 70h – 77h are reserved by the BIOS for hardware interrupts.



Hardware Interrupt Priorities The hardware interrupt priority is: NMI, IRQ0, IRQ1, IRQ2, IRQ8, IRQ9, IRQ10, IRQ11, IRQ12, IRQ13, IRQ14, IRQ15, IRQ3, IRQ4, IRQ5, IRQ6, and IRQ7.

Cont'd

Types of Interrupts, Continued

Hardware Interrupts The hardware interrupts that actually interface with the BIOS are IRQs 0, 1, 3, 4, 5, 6, 7, 13 and 14.

System Interrupt	Interrupt Request Line	Description
08h	IRQ0	System Timer (Timer Channel 0 output)
09h	IRQ1	Keyboard Controller Output Buffer Full Interrupt
0Ah	IRQ2	Cascade from Second Programmable Interrupt Controller
0Bh	IRQ3	COM2 or COM 4
0Ch	IRQ4	COM1 or COM 3
0Dh	IRQ5	LPT2 Parallel Port
0Eh	IRQ6	Floppy disk drives
0Fh	IRQ7	LPT1, Parallel Port
70h	IRQ8	Real Time Clock Interrupt
71h	IRQ9	Vertical Retrace Interrupt from Video Display Adapter
72h	IRQ10	Reserved
73h	IRQ11	Reserved
74h	IRQ12	Reserved in ISA. Mouse controller in EISA.
75h	IRQ13	Math coprocessor interrupt on any error
76h	IRQ14	IDE or MFM hard disk drive controller
77h	IRQ15	Second IDE hard disk drive controller. Sometimes used for power management.

Software Interrupts Software Interrupts are invoked via the x86 assembly language INT mnemonic. Most software interrupts are vectored to device service routines (DSRs) in the ROM BIOS or operating system.

Exceptions

- BIOS software interrupts 1Dh, 1Eh, 1Fh, 41h, 42h, 43h, and 46h do not service a specific device. They return ROM-resident hardware parameter tables.
- INTs 20h – 3Fh are software interrupts that are only to be used by the operating system (by convention).
- INTs 05h, 10h – 1Ah, 1Dh – 1Fh, 40h, 41h, 42h, 43h, 44h, and 46h can only be used by the system BIOS.

Interrupt Numbers

Every interrupt in the Intel x86 architecture is assigned a unique interrupt number, whether it comes from the CPU, hardware, or software.

Restricted Interrupt Numbers To maintain IBM compatibility, certain ranges of interrupts are reserved for special use.

- INTs 60 – 67h are used for user software interrupts.
- INTs 20h – 3Fh is reserved for the operating system.

Interrupt Numbers and Interrupt Vectors Each interrupt number is associated with a specific interrupt vector. An interrupt vector is in the doubleword segment:offset format of the routine assigned to an interrupt number. Interrupt vectors are stored in a table beginning at address 0:0000h. The vector for INT 00h is stored at address 0:0h through 0:03h. The vector for INT 02h is stored at 0:08h to 0:0Bh, the vector for INT 03h is stored at 0:0Ch to 0:0Fh, and so on.
POST writes the interrupt vectors to low memory and initializes the vector address of all interrupts used by the BIOS. When the operating system boots, it initializes all operating system-specific interrupt vectors.

Applications programs that add their own interrupt routines are responsible for initializing the interrupt vectors for their own interrupts.

Interrupt Vector Table

The originator of the interrupt does not need to know the memory address of the required interrupt handler. It only needs to know the interrupt number. The interrupt number points to the interrupt vector table, a table in low memory that contains the segmented address of the interrupt handling subroutine.

The interrupt handler's address is called the interrupt vector and the table is the interrupt vector table.

The interrupt vector table is normally supervised by the BIOS and DOS. When new interrupt handling routines are created, the programmer either uses an existing interrupt number and vector, or assigns a new one..

Interrupts Return to Next Instruction after Executing Interrupts automatically save the contents of CS and IP on the stack, so the computer can return to where it was after the interrupt is processed. The IRET instruction performs this function. IRET also restores the flags, CS, and EIP.

The interrupt process also saves the flag register on the stack and clears the interrupt flag (IF), temporarily preventing additional interrupts. It is a convention when writing interrupt routines to turn interrupts back on as soon as possible.

Available Interrupts Software interrupts 40h through 6Fh are defined, used, and possibly made available through the operating system or a memory manager. You can use INTs 60h – 6Fh (except 67h) to add a new BIOS service.

Nonmaskable Interrupt

The Nonmaskable interrupt (NMI) is a special case. It is generated by hardware devices and is used to demand immediate attention from the CPU. It often signals an emergency, such as a low voltage condition or a memory error.

The BIOS generates INT 02h to handle NMIs. See the INT02h description for additional information about the NMI.

Unexpected Interrupt Handler

AMIBIOS initializes unused interrupt vectors to the BIOS unexpected interrupt handler. The unexpected interrupt handler routine processes all interrupts that are either: user-defined processes (INT 1Ch and INT 4Ah, for example), or not meaningful to the BIOS (INT 73h or INT 7Fh, for example).

If an unexpected interrupt occurs, AMIBIOS either: returns to the caller with CF set to 1 and all registers preserved, or revectors the interrupt to a caller-supplied interrupt processing routine.

BIOS Register Conventions

Input to Function Calls The standard input registers are:

Register	Conventional Use
CS and IP	Automatically loaded, reserved, and restored as part of the interrupt process.
DS and ES	Preserved by the interrupt services.
SS	Not changed by the interrupt services.
SP	Preserved because, by coding convention, all BIOS device service routines leave the stack clean before returning.
AX	May be changed by the BIOS service. You cannot be sure the contents are the same after returning from a BIOS service.
BX	May be changed by the BIOS service. You cannot be sure the contents are the same after returning from a BIOS service.
CX	May be changed by the BIOS service. You cannot be sure the contents are the same after returning from a BIOS service.
DX	May be changed by the BIOS service. You cannot be sure the contents are the same after returning from a BIOS service.
SI and DI	May be changed by the BIOS service.

Output from Function Calls The standard output registers are:

Register	Conventional Use
AH	Used to return error information.
AL	Sometimes used to return error information.
CF	The Carry Flag (CF) is set if an error occurred when the interrupt request was processed.
FLAG Bits	All flag bits might be changed by the BIOS service. You cannot depend on any bit to be the same.

9 System BIOS Interrupts

System interrupts are routines used to access I/O devices without directly accessing the hardware. Interrupts supported by AMIBIOS are described in this chapter. Interrupts are mainly associated with specific peripheral devices. You can select an interrupt function by placing a value in a CPU register. The functions specify the activity to be performed by the interrupt service. BIOS functions are described under each interrupt heading. The actions that the BIOS interrupt service performs and required input and output values are listed. BIOS interrupts are software interrupts. The BIOS function calls for PCI, Socket Services, and Plug and Play are all invoked through the INT 1Ah interrupt service routine. The EISA and APM (Advanced Power Management) functions are invoked through INT 15h. The flash ROM function calls are invoked via INT 16h.

Interrupt Vector Table

Each time the BIOS initializes the computer, it creates a table of interrupt vectors (pointers to the location of the interrupt service routine entry point at 0:0000h).

Byte	Description
First	Least significant byte of offset (OLSB).
Second	Most significant byte of offset (OMSB).
Third	Least significant byte of segment (SLSB).
Fourth	Most significant byte of segment (SMSB).

Vector Example If the four-byte entry for an interrupt is stored in the interrupt vector table as 7D EA 00 F0, the interrupt entry point address is F000:EA7Dh.

Using the Interrupt Vector Table By replacing the existing vector in an Interrupt Vector table entry with a pointer to your own BIOS routine, you can add a new BIOS service or replace an existing service. An entry for each BIOS interrupt number from 00h through BFh appears in the Interrupt Vector Table. Counting in hex by fours, you can determine the address of the interrupt vector table to be added or replaced. For example, the interrupt vector table entry for INT 10h is at 0:0040h (4 times 10h). The interrupt vector table entry for INT 47h (available for use), is 0:011Ch.

BIOS Interrupt Summary

INT Code	Type	Function
00h	Processor	Divide by Zero
01h	Processor	Single Step
02h	Processor	Nonmaskable Interrupt (NMI)
03h	Processor	Breakpoint
04h	Processor	Arithmetic Overflow
05h	Software	Print Screen
06h	Processor	Invalid Op Code
07h	Processor	Coprocessor Not Available
08h	Hardware	Timer Tick
09h	Hardware	Keyboard Device
0Ah	Hardware	IRQ2 Cascade from Interrupt Controller 2
0Bh	Hardware	Serial Port (COM2)
0Ch	Hardware	Serial Port (COM1)
0Dh	Hardware	Parallel Printer (LPT2)
0Eh	Hardware	IRQ6 Floppy Disk
0Fh	Hardware	IRQ7 Parallel Printer (LPT1)
10h	Software	Video Service
11h	Software	Equipment List Service
12h	Software	Memory Size Service
13h	Software	Hard Disk Service
13h	Software	Floppy Disk Service
14h	Software	Serial Communications Service
15h	Software	Systems Services
16h	Software	Keyboard Service
17h	Software	Parallel Printer Service
18h	Software	ROM Basic
19h	Software	Bootstrap Loader
1Ah	Software	Time of Day Service
1Bh	Software	<Ctrl><Break>
1Ch	Software	User Timer Tick Service
1Dh	Software	Video Control Parameter Table
1Eh	Software	Floppy Disk Base Table
1Fh	Software	Video Graphics Table
20h – 3Fh	Software	DOS interrupts
40h	Software	Floppy Disk Revector
41h	Software	Hard Disk C: Parameter Table
42h	Software	EGA Default Video Driver
43h	Software	Video Graphics Characters
44h	Software	Novell NetWare API
46h	Software	Hard Disk D: Parameter Table
45h, 47h–49h	Software	Available
4Ah	Software	User Alarm
4Bh–63h, 65h, 66h	Software	Available
64h	Software	IPX (Novell NetWare)
67h	Software	EMS
68h – 6Fh	Software	Available
70h	Hardware	Real Time Clock
71h	Hardware	Redirect Interrupt Cascade
72h – 74h	Hardware	Reserved. Do not use.
75h	Hardware	Math Coprocessor Exception
76h	Hardware	Hard Disk
77h	Hardware	Suspend Request
7Ah	Software	Novell NetWare API

INT Code	Type	Function
78h, 79h, 7Bh–BFh	Software	Available

BIOS Stack Area

AMIBIOS uses the location at 30:0000h – 30:00FFh as a stack area. This area (the BIOS Stack Area) is used primarily for BIOS calculations and temporary storage.

Following the interrupt vector entry format shown above, the addresses for INTs C0h through FFh should occur in this space. For this reason, INTs C0h through FFh are not supported in AMIBIOS.

INT 00h through INT 07h

The first eight interrupts (00h through 07h) are called by the processor directly, but they can also be called via any software program using the INT instruction. See the INT 05h example below. All processors in the Intel x86 family handle the INT mnemonic.

INT 00h Divide by Zero

Input: None

Output: None

Description INT 00h is a logical or processor interrupt. INT 00h is generated by the CPU to handle any division operation that has a denominator value of zero. The exact behavior is dependent on the operating system or application program in use when the interrupt occurs. Most programs display an error message, such as "Divide By Zero" and then terminate.

INT 01h Single Stepping

Input: Trap bit = 1

Output: None

Description INT 01h is a logical or processor interrupt. INT 01h traces the execution of each instruction in a software program. Most debugging utility programs use this interrupt.

INT 02h Nonmaskable Interrupt (NMI)

Input: None

Output: None

Description An NMI is a hardware interrupt. The BIOS generates INT 02h, an interrupt service routine that handles NMIs in response to a hardware NMI. The hardware NMI is used primarily to halt the computer when memory errors occur. The status bits (I/O port 61h) indicate whether the NMI was caused by an memory parity check or I/O check. You can prevent the execution of all software interrupts by invoking CLI, with the exception of INT 02h, which handles NMIs. The NMI cannot be masked by CLI, but NMIs can be turned off, as described on the next page.

When the operating system boots the computer, it resets the interrupt vector that corresponds to the NMI to its own routine. When an error that causes an NMI occurs, the operating system NMI routine calls the BIOS INT 02h NMI handling routine. The BIOS NMI handling routine displays an error message that describes the type of hardware error that caused the NMI.

Disabling NMIs The NMI is disabled by writing to I/O port 70h with data bit 7 set and enabled by writing to port 70h with data bit 7 reset.

How NMIs Occur Any of the following conditions can cause an NMI:

- an onboard dynamic RAM parity failure,
- a 32-bit adapter card memory parity failure,
- an error reported by the I/O channel adapter card through the I/O channel check (-IOCHCK) signal,
- a bus timeout on an EISA slot,
- when a program sets bit 7 of I/O Port 462h (EISA only),
- a fail-safe timer NMI (EISA only), or
- when an EISA card is enabled or disabled.

INT 03h Breakpoint

Input: None

Output: None

Description INT 03h is a logical or processor interrupt. It provides a single-byte instruction (CCh) that halts the execution of a program so the programmer can evaluate the CPU registers and other areas of memory. INT 03h is useful in debugging. It is called by many commercial debugging programs.

INT 04h Overflow Error

Input: Overflow bit of FLAGS register = 1

Output: None

Description INT 04h is a logical or processor interrupt. When a numeric overflow occurs following a mathematical operation, the Overflow bit in the FLAGS register is set. The INTO instruction calls INT 04h when executed afterwards. If INT 04h is invoked, the Overflow bit is not read. INT 04h is not used often, so most operating systems set it to an IRET.

INT 05h Print Screen

Input: None

Output: None

Description INT 05h is a software interrupt. The contents of the screen are moved to a printer when the <Print Screen> key is pressed. Use this INT to accomplish the same task. INT 05h only works in text modes. It does not print graphics screens.

INT 06h Invalid Op Code

Input: None

Output: None

Description INT 06h is a logical or processor interrupt that is called after an Invalid Op Code Exception Error is generated by the processor. AMIBIOS allows you to replace the interrupt vector table entry for INT 06h with a routine.

INT 07h Coprocessor Not Available

Input: None

Output: None

Description INT 07h is a logical or processor interrupt. INT 07h is called by the processor when the emulation bit (EM) in the CPU control register is set and an ESC instruction is encountered.

Programs that use coprocessor emulation can trap this interrupt and provide another routine to be executed when this interrupt occurs.

Interrupts 08h Through 0Fh

INTs 08h – 0Fh are generated by the interrupt controller and correspond to IRQs 0 – 7. They are vectors to handle IRQs 08h – 0Fh. Since these interrupts are not generated by the CPU directly, the interrupt controller sets the corresponding IRQ line to request that the CPU generate the appropriate interrupt (08h – 0Fh) when invoked.

IRQs usually have a fixed priority scheme: NMI, 0, 1, 2, 8, 9, 10, 11, 12, 13, 14, 15, 3, 4, 5, 6, and 7. The interrupt controller determines which interrupt request (IRQ) has the highest priority and then forwards that request to the CPU. All ISA and EISA computers have two interrupt controllers and sixteen IRQ lines. The IRQ 2 line cascades the second interrupt controller to the first interrupt controller. IRQ 2 is not used by any other devices and gives IRQs 8 – 15 a higher priority than IRQs 3 – 7.

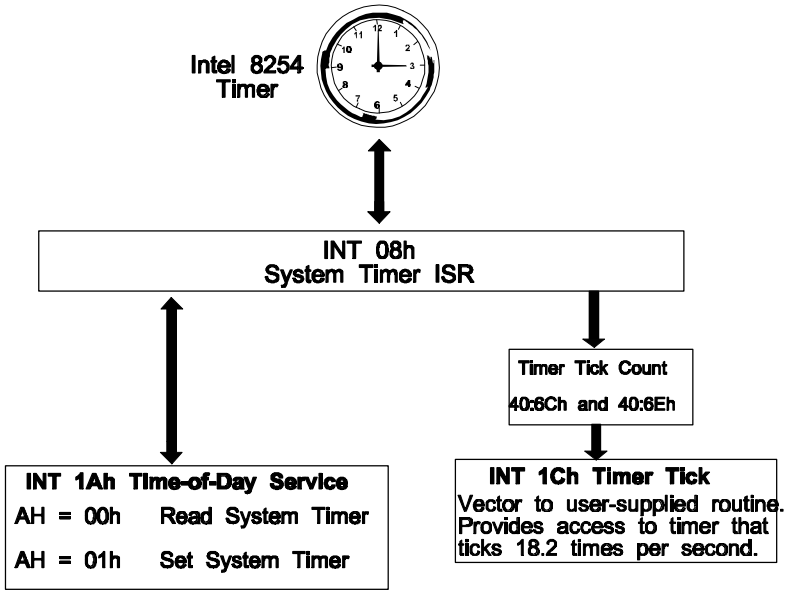
INT 08h Timer Interrupt (IRQ0)

Input: None

Output: None

Description INT 08h can be used to measure time increments independent of the system clock frequency. INT 08h is called approximately 18.2 times per second. INT 08h increments the system time count at location 40:6Ch through 40:6Eh every time it is called. If the system time count (40:6Ch) exceeds 24 hours, the Timer Overflow Flag (40:70h) is set, the date is incremented by the BIOS, and the system time count is reset to 0. INT 08h also decrements the floppy disk motor count at 40:40h. When the count reaches 0, INT 08h turns the floppy drive motor off.

INT 08h also issues an INT 1Ch Timer Tick interrupt every time it is called. Programmers can revector INT 1Ch to their own routines and use the clock feature for timed events. The following graphic illustrates how the system timer is used with the BIOS.

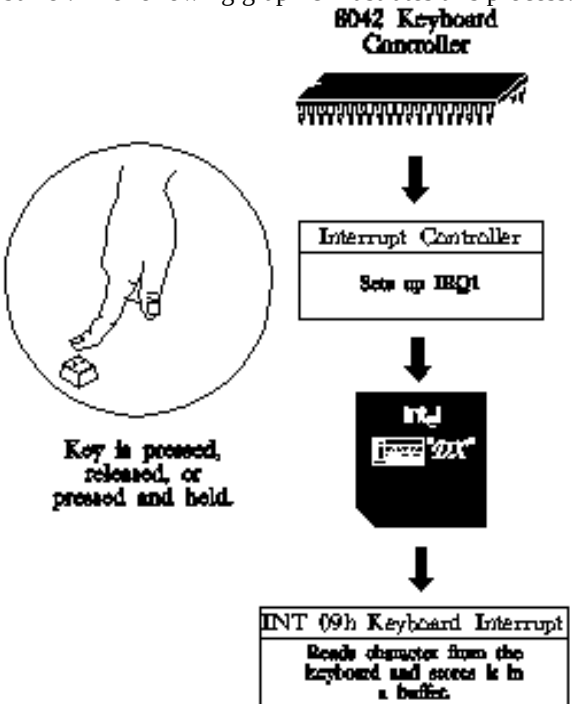


INT 09h Keyboard Interrupt (IRQ1)

Input: None

Output: None

Description: ISA computers generally use 8042 processors to control the keyboard and keyboard registers. If a key is pressed, released, or pressed and held, the 8042 issues an interrupt signal to the interrupt controller. The interrupt controller sets IRQ line 1 so the CPU can issue an interrupt. The BIOS INT 09h routine is then called. INT 09h reads the character from the keyboard and stores it in a buffer. The following graphic illustrates this process.



Cont'd

Keyboard Key Processing

When...	the BIOS...
The BIOS receives the ASCII Scan code for any key.	The ASCII scan code is read from port 60h and is placed in the 32-byte keyboard buffer (40:1Eh).
<Ctrl>, <Shift>, or <Alt> is pressed.	The Keyboard Control flags (40:17h and 40:18h) and the Keyboard Mode State and Type flag (40:96h) are updated.
the <Ctrl><Alt><Del> keychord is pressed.	The reset flag (40:72h) is set to 1234h and the routine jumps to the POST tests, followed by a reboot. This allows POST to skip the memory test it usually performs.
<Pause> is pressed.	The computer enters a wait loop until a valid ASCII character key is pressed.
<Print Screen> is pressed.	The BIOS issues an INT 05h call.
The <Ctrl><Break> keychord is pressed.	The BIOS issues an INT 1Bh call.
<SysReq> is pressed.	INT 15h Function 85h is called. This routine is initialized by the BIOS to do nothing. Other software programs can trap this interrupt and provide an interrupt handler for <SysReq>.

Testing for any Keystroke After reading the scan code from I/O port 60h, an INT 15h Function 4Fh is issued. This function is initialized by the BIOS to do nothing. Other software programs can trap this interrupt and provide an interrupt handler.

INTs 0Ah Through 0Fh Miscellaneous Interrupts

Input: None

Output: None

Description These interrupts are defined by other external peripheral devices attached to the computer.

In ISA computers, INTs 0Ah – 0Fh are often attached to the IRQ lines shown in the following table. When these interrupts are invoked, the corresponding IRQ line is enabled, alerting the CPU that the attached device must be serviced.

Interrupt	Common Hardware Interface
0Ah	IRQ2 Cascade to second interrupt controller.
0Bh	IRQ3 Serial port 2 or 4.
0Ch	IRQ4 Serial port 1 or 3.
0Dh	IRQ5 Parallel port 2.
0Eh	IRQ6 Floppy disk drive.
0Fh	IRQ7 Parallel port 1.

IRQ assignments are not fixed. They can vary. AMIDiag Version 4.0 or later lists the hardware interrupt assignments for your computer.

EISA computers are even more flexible. The EISA Configuration Utility (ECU) allows you to assign IRQs in any order to any EISA adapter card, with few restrictions.

The above table is probably not accurate for most EISA computers.

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’d

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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’d

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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’d

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'd

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'd

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'd

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'd

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’d

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'd

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'd

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'd

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–FFh Non-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’d

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'd

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’d

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'd

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’d

INT 13h Floppy Disk Service, Continued

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'd

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'd

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'd

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'd

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'd

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'd

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.
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.

I/O Port	R/W	Description
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.
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.

I/O Port	R/W	Description
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'd

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'd

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'd

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 01h COM2
= 02h COM 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 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'd

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'd

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'd

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 7Fh APM 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'd

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'd

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'd

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’d

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 3 1	BIOS Power Management is disabled (v1.2).
		Bits 30-21	Reserved
		Bit 2 0	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’d

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'd

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'd

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

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'd

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'd

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

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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 µseconds has elapsed. Bit 7 of the first byte at ES:BX is set after the wait has expired. The µseconds 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'd

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 µseconds.

Cont'd

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'd

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'd

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–FFh (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–FFh (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'd

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'd

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'd

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'd

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'd

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'd

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'd

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’d

Function C2h Subfunction 07h Set Mouse Handler Address

Input:

AH	=	C2h	
AL	=	07h	
ES:BX	=	Address of programmer's 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'd

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'd

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'd

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'd

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’d

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

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'd

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).
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.

Offset	Description
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
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.
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.

Offset	Description
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
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'd

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'd

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.
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

Offset	Description
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 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
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'd

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 “SMAP”. 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	=	“SMAP” 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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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’d

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

Function 02h Read EPP Status

Input:	AH	=	02h
	BX	=	5050h (“PP”)
	CH	=	45h (“E”)
	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 “PPE”. EPP v3.0+ BIOS is installed 4550:50h “EPP”. 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'd

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.

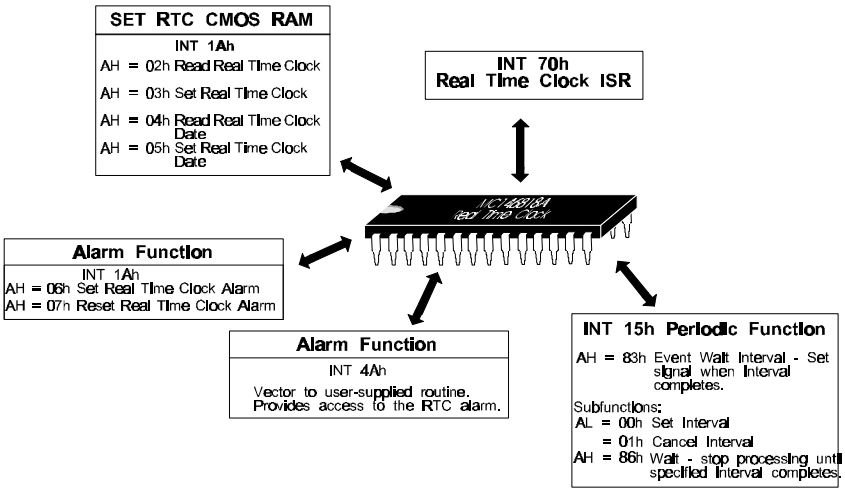
INT 1Ah Functions

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
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

Function	Service	Title
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’d

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.

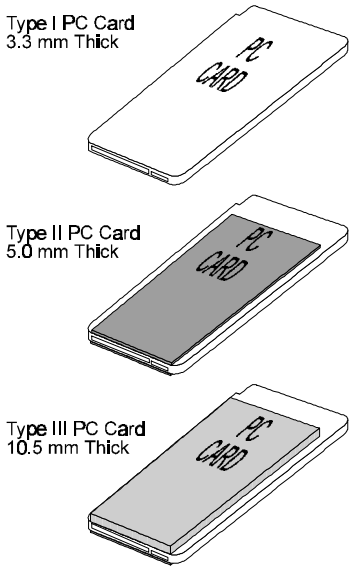
Socket Services provides the software interface to the hardware controlling PCMCIA-compatible cards (memory and I/O) in sockets. Socket Services provides the lowest level access to PCMCIA cards but does not interpret the content of the cards.

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'd

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'd

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.

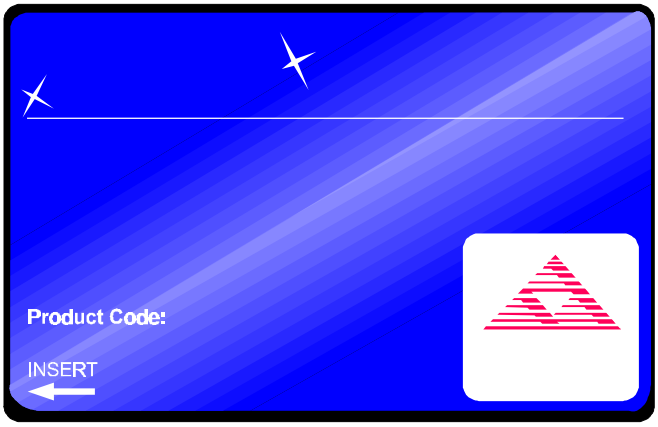
PCMCIA Software The primary architectural software building blocks required by PCMCIA PC Cards are Socket Services and Card Services.

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'd

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'd

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'd

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'd

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).

INT 1Ah Socket Services

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’d

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'd

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'd

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'd

Function 84h Inquire Adapter, Cont'd

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'd

Function 84h Inquire Adapter, Cont'd

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'd

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'd

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’d

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’d

Function 87h Inquire Window, cont'd

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.*
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.

Flag	Description
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'd

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.
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.

Flag	Description
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’d

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'd

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’d

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'd

Function 8Ah Get Page, cont'd

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'd

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'd

Function 8Bh Set Page, Cont'd

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'd

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'd

Function 8Ch Inquire Socket, cont'd

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'd

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'd

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’d

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’d

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’d

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'd

Function 95h Inquire EDC, cont'd

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'd

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'd

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'd

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'd

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'd

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’d

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'd

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'd

Function A0h Get and Set SS Address, cont'd

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'd

Function A0h Get and Set SS Address, cont'd

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'd

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'd

Function A1h Get Access Offsets, cont'd

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'd

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’d

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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'd

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’d

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’d

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’d

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'd

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 359.

Cont'd

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’d

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'd

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’d

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'd

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’d

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’d

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'd

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'd

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'd

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'd

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'd

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'd

SelectPrimaryBootDevices 07, cont'd

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'd

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'd

Plug and Play Option ROMs, Continued

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'd

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'd

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'd

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’d

Function 50h Get Number of DMI Structures, cont'd

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'd

Function 51h Get DMI Structure, cont'd

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'd

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’d

Function 53h Get DMI Event Information, Cont'd

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'd

Function 53h Get DMI Event Information, Cont'd

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'd

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'd

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'd

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'd

Function 57h Write General Purpose NVRAM Data, Cont'd

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.

10 Power Management AMIBIOS

Power management is the coordination and manipulation of power-consuming computer components to minimize power consumption and maximize battery life. Power management techniques include turning power off to a specific device and slowing or stopping the device's clock. Power management features are used in laptops, notebook, and handheld computers. Since many of these computers run on battery power, they must conserve power to permit the computer can run as long as possible. Some desktop computers also incorporate power management features.

System BIOS is a Logical Place to Start Because it directly controls the hardware, the system BIOS is the logical place to implement power management. The American Megatrends Power Management AMIBIOS does just that, it adheres to the APM specification.

APM

AMIBIOS adheres to the Advanced Power Management (APM) Specifications developed jointly by Intel and Microsoft. APM is a layered approach that defines a cooperative environment where the BIOS, operating system, and applications programs work together to reduce power consumption and conserve battery power. APM takes a system-wide view of power management. The BIOS, operating system, and software applications programs all play a role. The operating system can provide precise power management information to the BIOS, permitting the BIOS to intelligently conserve power use.

Cont’d

APM Features

- APM can be implemented in any operating system. Microsoft offers APM support in MS-DOS® 5.0 and above and Microsoft Windows 3.1 and above. APM is compatible with applications that are not aware of APM.
- APM is an open platform-independent specification that can be implemented on any Intel x86-based CPU. Intel and Microsoft made APM an open specification for all BIOSes.
- APM is simple for PC users. Microsoft has shipped APM drivers for MS-DOS 5.0 and Windows 3.1. Once configured, the end user does not have to configure or adjust any parameters.

APM Power States The APM specification power states are:

Power State	affects...
Ready	applies to both individual components and the computer as a whole.
Standby	applies to both individual components and the computer as a whole.
Suspend	a low power condition that applies to the computer as a whole but not to individual components.
Off	applies to both individual components and to the computer as a whole.

Cont'd

Ready State In the Ready State, the computer or device is fully powered up and ready for use. The computer can be active or idle.

Standby State The Standby State is an intermediate system-dependent state that tries to conserve power. This state is entered when the CPU is idle and no device activity occurs for a prespecified length of time. The computer does not return to the Ready State until:

- a device raises a hardware interrupt, or
- any controlled device is accessed.

All data and operational parameters are preserved when the computer is in Standby mode.

Suspend State Suspend is the lowest level of power consumption available that still preserves operational data and parameters. This state can be initiated either by AMIBIOS or software one layer above the BIOS.

AMIBIOS can place the computer in Suspend mode with no notification if it detects a situation that requires an immediate response, for example, when the battery power becomes critically low.

When the computer is in Suspend mode, no computation is performed until normal activity is resumed and the computer leaves this state. Activity cannot resume unless signaled by an external event, such as a key press, a Real Time Clock alarm, or a modem Ring signal through the serial port.

Cont'd

Off State The computer is powered down and inactive in Off state. Data and operational parameters may or may not be preserved in this state.

Power Management State Changes The computer and devices in the computer can change from one power state to another by explicit command or automatically, based on APM parameter settings and activity.

Power capabilities differ from device to device. Some devices may not be able to enter all states. Some devices may have built-in automatic power management features invisible to the computer. These devices are outside the scope of this manual.

Power Consumption Power management features control the power consumption of many components. Almost every facet of power consumption is monitored. When the computer is idle for an end user-specified period of time, the computer automatically enters Power Down Mode. The end user can also power down the computer by pressing an externally-mounted *Power Down* switch.

IDLE Mode In IDLE Mode, the CPU receives a very low clock frequency and all other clocks except the DRAM refresh are stopped. The clock can also be stopped for a static CPU. Pressing the externally-mounted *IDLE* switch exits Idle Mode.

Cont'd

Power Management Interrupt INT 77h is the Software Power Management Interrupt (SPMI) in AMIBIOS. The request to change the state of the machine to Power Down Mode comes to the BIOS via OEM-specified sequence microcode.

OS/2, Unix, and Xenix Support All power management features work with all operating systems.

Modes Power management is implemented in different levels. Each level saves more power than the previous level and each level can be accessed directly or incrementally. The levels are:

Mode	Description
Full On	This is full power mode. A computer built on a power management chipset initially powers on in this mode. The LCD and hard disk drive are powered off in inactive for a set length of time. The timeout values are set via WINBIOS or AMIBIOS Setup and AMIBCP. When AMIBIOS determines that the computer does not need maximum power, it enters Idle mode.
Idle	This mode is entered when the CPU has been idle for a specified length of time. AMIBIOS automatically enters this mode. AMIBIOS returns to Full On mode when additional power is required.
Sleep	AMIBIOS determines if the performance has dropped to a level such that the computer can function efficiently in Sleep mode. Sleep mode can only be entered from Idle mode. The length of time that the BIOS waits before entering Sleep mode is set in WINBIOS Setup, AMIBIOS Setup, or AMIBCP.
Suspend	This mode uses the least amount of power necessary for the computer to function. Suspend mode is entered from Sleep mode. The computer can go from Idle mode directly to Suspend mode via timers configured by WINBIOS Setup, AMIBIOS Setup, or AMIBCP. If an external switch is pressed, the computer can go to Suspend mode from any other mode. Pressing the switch again returns the computer to Full On mode.

Cont'd

AMIBIOS 98 fully supports all Advanced Power Management (APM) specifications and the power management provisions of the ACPI specification.

11EISA Overview

This chapter describes the EISA bus and the interaction between the EISA AMIBIOS, American Megatrends EISA Configuration Utility (ECU), System Configuration Utility (SCU), and EISA computers.

EISA

EISA is an acronym for Extended Industry Standard Architecture. EISA is basically a superset of the Industry Standard Architecture (ISA), based on the original IBM AT specifications. The EISA specifications allow 32-bit memory addressing to be used by the CPU, DMA devices, and bus mastering devices. EISA devices can also perform either 16-bit or 32-bit data transfers.

EISA Features

The EISA specification introduced the following features:

- bus mastering, with an arbitration scheme to prioritize bus access and use,
- 32-bit burst mode DMA and three additional DMA transfer modes,
- 16 additional data lines, allowing 32-bit data transfers,
- eight additional address lines, allowing up to 4 GB of address space,
- complete compatibility with XT and ISA standards, and
- both level-triggered and edge-triggered interrupts.

EISA and ISA Differences

The most important difference between ISA and EISA is that EISA system configuration is done through software, rather than the hardware switches in an ISA computer. I/O ports, ROM addresses, IRQ lines, and DMA lines for EISA motherboards and expansion cards are configured using an EISA Configuration Utility (ECU), where an ISA card uses DIP switches and jumpers. The American Megatrends System Configuration Utility (SCU) not only configures EISA computers, but also configures PnP and PCI computers.

EISA Bus Specifications

EISA computers have 32-bit expansion slots that are fully compatible with 8-bit and 16-bit ISA expansion slots.

EISA expansion slots have 188 pins. The upper 98 pins are exactly the same as the standard ISA pinouts. The lower 90 pins are used for EISA bus signals.

Data can flow on the EISA bus much faster than on the ISA Bus. Not only does EISA provide a wider 32-bit bus, it also provides a maximum 33 MB per second bus transfer rate. An ISA bus can transfer data at a maximum rate of only 8 MB per second.

EISA achieves this higher throughput via high-speed burst transfers, which use only one clock cycle. Normal EISA (ISA-compatible) transfers use two clock cycles.

EISA Bus and the ISA Bus

Attribute	EISA Bus	ISA Bus
Number of Data Lines	32	16
Number of Address Lines	32	24
Bus Clock Rate	about 8 MHz	about 8 MHz
Bus Modes	8-, 16-, and 32-bit	8- and 16-bit
Burst Transfer Rate	33 MB per second	8 MB per second at 0 wait states
DMA characteristics	Supports 8, 16, and 32-bit DMA	Supports 8 and 16-bit DMA
Normal DMA Transfer Rate	5.3 Mbs	1.2 – 1.6 Mbs
Maximum DMA Transfer Rate	33 Mbs	4 Mbs
DMA Cycle Time	0.12 – 1.0 µseconds	1.25 – 1.67 µseconds
Adapter Card Pin Count	188 pins	98 pins
Bus Master	Multiple intelligent 8-, 16-, and 32-bit bus masters	Limited bus mastering
Configuring Adapter Cards	Autoconfiguration through ECU. DIP switch and Jumper setting still available.	Only DIP switch and jumper setting available.

EISA 32-bit Memory Addressing

32 memory address lines are available in EISA computers. EISA computers use Intel x86 CPUs that allow 32-bit memory addressing. Up to 4 GB of physical RAM can be configured in an EISA computer.

An ISA card used in an EISA computer can address only up to 16 MB of RAM (because it uses 24-bit ISA memory addressing) but EISA adapter cards can use all available RAM.

EISA Bus Masters

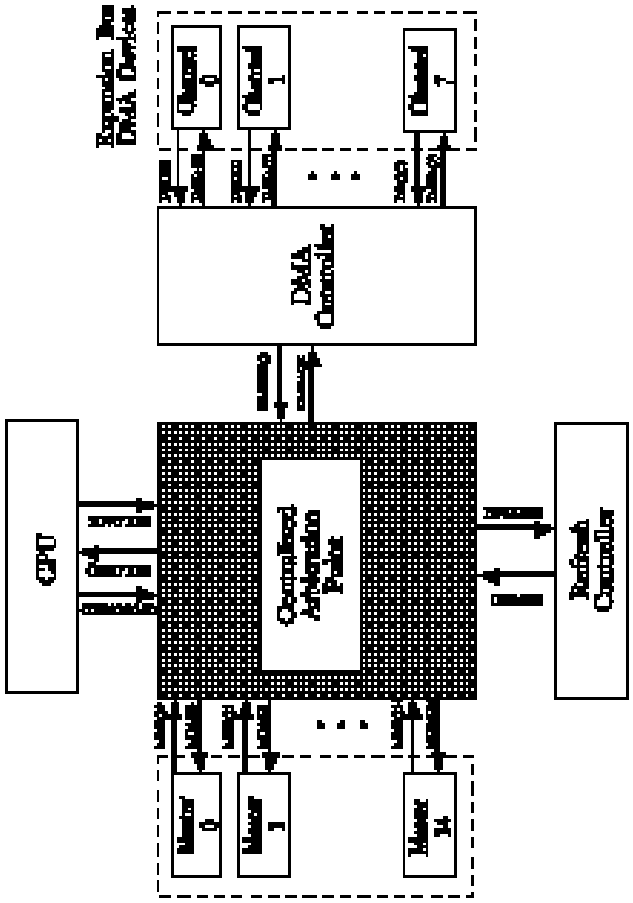
A bus master is a device that takes control of the bus during data transfers supervised by the bus master. The EISA specification permits up to 15 intelligent bus mastering devices. Although it is possible to add bus mastering to an ISA computer, it can only be done on a limited scale and the bus master cannot be intelligent.

With an EISA bus master, the CPU does not have to monitor all data transfers. An intelligent EISA bus mastering device uses a dedicated I/O processor and local memory to facilitate and manage data transfers on the EISA bus.

EISA Bus Master Arbitration Memory refresh, DMA, and each EISA slot has a preassigned priority level. Each priority level has its own line to the central arbitration point. EISA bus arbitration allows the latency for each device on the bus to be determined. The EISA bus master then knows how much response time to allocate for all devices on the bus. Several I/O processors can run concurrently on an EISA bus.

Cont’d

EISA Bus Arbitration Priority Memory refresh and DMA have the highest priority. The assignment of arbitration levels 1 – 15 to bus master expansion slots is defined by the EISA motherboard manufacturer. The following block diagram shows the EISA components involved in arbitration. It identifies 15 bus masters, which is the limit in the EISA Specification.



Cont'd

Common Clock Signal EISA devices can synchronize data transfers to a common clock signal generated and optimized by the EISA motherboard.

Type of Data Transfer	Rate
Standard transfers	two clock cycles
EISA burst transfers	one clock cycle
Bus masters slave devices	1.5 clock cycles

Bus Master Components An EISA bus master includes a dedicated I/O processor and local memory. The I/O processor is a specialized processor that drives the address, data, and control signals for intelligent peripherals, which become slave devices, during a bus cycle. Bus masters improve performance by taking on simple tasks that would otherwise be the responsibility of the host processor.

Bus Master Uses The EISA bus master is designed for I/O peripherals that need optimum performance or advanced memory access functions (such as non-ordered scatter/gather data operations). EISA can use 32-bit devices that contain dedicated I/O processors and require fast bus data transfer rates using a fast burst transfer mode.

EISA DMA

EISA DMA devices have seven channels, just like ISA, but DMA transfer is much faster and supports 8-, 16-, and 32-bit data transfers.

EISA Data Transfer Cycles Four cycle control sequences for transferring data between the DMA device and memory are available. These cycles are:

EISA Cycle Type	Execution Rate
AT-compatible	Executes one transfer in eight bus cycles. Two additional bus cycles are added for each wait state. ISA DMA devices can use this cycle to transfer data to or from 8-, 16-, or 32-bit memory.
Type A	Executes one transfer cycle in six bus cycles (longer if the transferred data requires data size translation). Supports 8-, 16-, and 32-bit DMA devices. Data size translation is performed automatically for transfers to mismatched memory.
Type B	Executes one transfer in four bus cycles (longer if the transferred data requires data size translation). Supports 8-, 16-, and 32-bit DMA devices and perform automatic data-size translation for transfers to mismatched memory. Transfer time can be cut in half in some ISA devices by using this type of transfer.
Type C (Burst DMA)	Executes one transfer cycle in one bus cycle. Adds one cycle for each simultaneous transfer and each additional wait state. Supports 8-, 16-, and 32-bit DMA devices and perform automatic data-size translation for transfers to mismatched memory.

Cont'd

Using Type A and B Faster than ISA-Compatible Most ISA-compatible DMA devices can transfer data about 120% faster by programming the EISA DMA controller to use Type A and B transfers instead of ISA-compatible timing.

EISA DMA Modes

DMA Mode	Description
ISA-compatible	DMA request and acknowledge cycles are performed during each DMA transfer cycle.
Block Transfer	The peripheral device that requires service makes a DMA request. The DMA controller performs a DMA acknowledge cycle and executes DMA transfer cycles continuously until the DMA request is removed or the terminal count is reached. Devices that use ISA-compatible timing should not use this mode.
Demand Transfer	The peripheral device that requires service makes a DMA request. The DMA controller performs a DMA acknowledge cycle. Bus transfers continue until the terminal count register value is reached. Devices that use ISA-compatible timing should not use this mode.
Cascade	A bus mastering device that wants bus ownership asserts a DMA request on the channel. The DMA controller performs a DMA acknowledge cycle. Bus ownership is transferred to the ISA bus-mastering requester. DMA channel 4 uses this mode to cascade DMA channels 0–3 Controller Block to the DMA Channels 4–7 Controller Block. A DMA channel can be programmed in cascade mode for use with external 16-bit bus masters.

Cont’d

Benefits of Arbitration Arbitration provides increased efficiency and performance. Arbitration manages the time between the DMA device request and the grant events.

Arbitration does not decrease ISA compatibility. Existing hardware and software can take advantage of arbitration without modification, so it may actually improve compatibility with older computers.

EISA DMA Cycle Types

DMA Cycle Type	Size of Transfer	Maximum Transfer Rate (MBs)	Compatibility
ISA-Compatible	8-bit	1.0	All ISA
ISA-Compatible	16-bit	2.0	All ISA
Type A	8-bit	1.3	Mostly ISA
Type A	16-bit	2.6	Mostly ISA
Type A	32-bit	5.3	EISA Only
Type B	8-bit	2.0	Some ISA
Type B	16-bit	4.0	Some ISA
Type B	32-bit	8.0	EISA Only
Burst DMA (Type C)	8-bit	8.2	EISA only
Burst DMA (Type C)	16-bit	16.5	EISA Only
Burst DMA (Type C)	32-bit	33.0	EISA Only

EISA Interrupt Handling

The original PC and ISA buses use edge-triggered interrupts. Edge-triggered interrupts are easy to implement but are also susceptible to false triggering and cannot be shared with other interrupts. Edge-triggered interrupts are signaled by the rising edge of the interrupt signal wave form. Other than the line that the signal came from, there is no way for the computer to identify an edge-triggered interrupt. Edge-triggered interrupts cannot be shared.
EISA supports edge-triggered interrupts to maintain ISA compatibility, but also provides level-triggered interrupts. Level-triggered interrupts are less susceptible to noise and allow multiple peripherals to share the same interrupt level. Level-triggered interrupts are signaled by a continuous logic-level voltage, permitting interrupt sharing.

EISA System Configuration

EISA permits automatic configuration of system resources and adapter cards and lowers dependence on switches and jumpers.

EISA specifies a product identification mechanism for motherboards and adapter cards. The computer automatically interrogates each adapter card during POST for the product identifier, compares it with the ID stored in EISA Extended CMOS RAM, and configures the adapter card accordingly.

Cont’d

EISA Configuration Files EISA adapter cards come with a configuration file (CFG file). EISA motherboards come with both a CFG file and an ECU. The ECU configures adapter cards that have EISA .CFG files and store configuration information in EISA CMOS RAM. Using an ECU guarantees that conflicts or contention issues between adapter cards are minimized. The ECU controls the assignment of all necessary system resources.

Configuration Characteristics EISA configuration consists of the following elements:

- an ECU for motherboard and adapter card configuration,
- CFG files for the motherboard and adapter cards,
- EISA CMOS RAM to store configuration parameters,
- a mechanism to save and restore a backup copy of the configuration parameters,
- BIOS routines to read from and write to CMOS RAM, and
- automatic detection and initialization of adapter cards by the BIOS during BIOS Power-On Self Test (POST).

Adapter Cards and EISA Slot Numbers Each adapter card (EISA or ISA) is installed in an expansion slot on the EISA motherboard. The slots are numbered from 1 to 15. The EISA motherboard is always slot 0.

Each EISA expansion slot has a unique I/O address space of 1024 bytes (1 KB). The BIOS uses these registers and the information written to EISA CMOS RAM to initialize the EISA adapter cards during POST.

If the slot is occupied by an ISA adapter card, the I/O space is limited to 00100h – 003FFh. The ECU can display the proper switch and jumper settings for the ISA adapter card or device if a CFG file is provided with the ISA adapter card or device.

EISA Configuration Utility

The function of the ECU is to read and write the system configuration and adapter card parameters such that a conflict-free environment is established. American Megatrends provides both an ECU for EISA computers and the System Configuration Utility (SCU) to configure EISA, PnP, and PCI computers.

CFG Files The ECU or SCU reads the CFG files that are provided by the manufacturer of every EISA adapter card. The CFG files contain the product ID, the product's system resource requirements, and the product's initialization information.

Configuration Data Stored in EISA CMOS RAM Initialization information is read by the ECU and stored in EISA CMOS RAM. A backup copy of the CMOS RAM configuration data is also stored on disk. The BIOS reads CMOS RAM and executes the initialization instructions during EISA BIOS POST when the computer is rebooted. Initialization usually consists of reading and writing to I/O ports assigned to the device.

EISA System Resources EISA system resources include:

- DMA channels,
- memory,
- interrupt request lines (IRQs), and
- I/O ports.

The ECU verifies that the resources requested by the slot device are not already assigned to another device and then allocates them. The allocation information is stored in EISA CMOS RAM and is accessed by the BIOS during POST.

EISA Configuration Overlay Files Manufacturers may not be able to perform all initializations in the framework of a CFG file. Features and resources may be specific to the adapter card and may not be configurable by the ECU. For these situations, the EISA specification permits CFG file extensions, or overlay files.

EISA Product ID

I/O port addresses 0zC80 – 0zC83h (z = the slot number) store the EISA four-byte compressed product ID number. The I/O port information differs for motherboards and adapter cards.

For an EISA motherboard

I/O Port	Description
0C80h	Bit 7 Reserved. Should be 0. Bits 6–2 First letter of the manufacturer code. Bits 1–0 First two bits of the second letter of the manufacturer code.
0C81h	Bits 7–5 Remaining bits of the second letter of the manufacturer code. Bits 4–0 Third letter of the manufacturer code.
0C82h	Bits 7–0 Manufacturer's product number.
0C83h	Bits 7–0 Product revision number.

For an EISA Adapter Card

I/O Port	Description
*z*C80h	Bit 7 Reserved. Should be 0. Bits 6–2 First letter of the manufacturer code. Bits 1–0 First two bits of the second letter of the manufacturer code.
*z*C81h	Bits 7–5 Remaining bits of the second letter of the manufacturer code. Bits 4–0 Third letter of the manufacturer code.
*z*C82h	Bits 7–4 Second hex digit of the product number. Bits 3–0 First hex digit of the product number.
*z*C83h	Bits 7–4 Product revision level. Bits 3–0 Third hex digit of the product number.

EISA CFG Filenames

CFG filenames must consist of an exclamation point, the product ID, and the DOS file extension .CFG.

File Naming Example !AMI16B4.CFG is the sample CFG file. The following table identifies the components of this CFG file name.

Code	Description
!	Identifies a CFG file.
AMI	Manufacturer ID.
16	Product Number.
B4	Product Revision Level.

Duplicate File Names The ECU or SCU renames CFG files when it finds duplicate CFG filenames. The ECU/SCU changes the exclamation point in the filename to the number of the duplicate.

For example, if the ECU/SCU finds multiple CFG files for AMI16B4, the first CFG file is named !AMI16B4.CFG, the next 1AMI16B4.CFG, the next 2AMI16B4.CFG, and so on.

EISA System AMIBIOS

The EISA System BIOS works with the ECU. The EISA BIOS POST routines use the information stored in EISA CMOS RAM to initialize the computer. The EISA BIOS provides software routines to read and write this information to and from CMOS RAM. These routines can be called using a software INT instruction.

EISA BIOS Interrupts

There are two BIOS INT 15h Function D8h routines used by the ECU to manipulate the information in CMOS RAM, briefly described below. Detailed explanations of these routines are in the INT 15h section.

Clear EISA CMOS RAM Call INT 15h with AH = D8h and AL = 02h (16-bit addressing) or AL = 82h (32-bit addressing).

Write EISA CMOS RAM Call INT 15h with AH = D8h and AL = 03h (16-bit addressing) or AL = 83h (32-bit addressing).

Reading EISA CMOS RAM The BIOS POST routines and other software drivers use:

- INT 15h Function D8h subfunction 00h/80h *Read Slot Configuration Information*, and
- INT 15h Function D8h subfunction 01h/81h *Read Function Configuration Information*

to read CMOS RAM data. Detailed explanations of these routines are in the INT 15h section

12 DMI Data Structures

The DMI (Desktop Management Interface) specifies a number of data structures that are used when interfacing with the DMI-aware AMIBIOS system BIOS.

Every DMI structure has a formatted section and an optional unformatted section.

DMI Section Type	Description
Formatted	The formatted section begins with an optional four-byte header. The remaining data in the formatted section and the length of the section is determined by the structure type.
Unformatted	The unformatted section is used to pass variable data, such as text strings.

DMI Structured Header

Every DMI BIOS structure begins with a four-byte header in the following format:

Offset	Name	Length	Description
00h	Type	Byte	This field specifies the structure. Types 0 - 127 are defined in the DMI specification. Types 128 - 256 are reserved for system- and OEM-specific information.
01h	Length	Byte	This field specifies the length of the formatted area of the structure. The length of the structure's stringset is not included. The structure begins with the *Type* field.
02h	Handle	Word	This field specifies the structure handle, a unique 16-bit number between zero and the total number of structures minus one. The handle is used with Function 51h Get DMI Structure to retrieve a specific structure.

Accessing DMI Structures

Field	Description	Used in DMI function
NumStructures	Number of DMI structures	50h Get DMI Structure
dmiStrucBuffer	Text strings associated with a specified DMI structure	51h Get DMI Structure

DMI Text Strings

Text strings associated with a specific DMI structure are returned in the buffer pointed to by the value in dmiStrucBuffer. Text strings are appended directly to the formatted part of the DMI structure. Each text string is terminated by a null byte (00h). Each set of text strings is terminated by an additional null byte (00h).

When the formatted part of the DMI structure references a text string, it specifies a non-zero string number that is part of the structure's string set.

Each text string is limited to 64 significant characters.

DMI BIOS Information

Offset	Name	Size	Value	Description
00h	Type	Byte	0	BIOS information indicator
01h	Length	Byte	13h	Length in bytes
02h	Handle	Word	Varies	
04h	Vendor	Byte	Varies	String number of the BIOS vendor name
05h	BIOS Version	Byte	Varies	String number of the BIOS version
06h	BIOS base address segment	Word	Varies	Segment location of the BIOS starting address. The size of the runtime BIOS image is computed by subtracting the starting address segment from FFFFh and multiplying the result by 16.
08h	BIOS release date	Byte	Varies	String number of the BIOS release date

Offset	Name	Size	Value	Description
09h	BIOS characteristics	Qword	Bit field	Specifies BIOS functions Bits 63-48 System vendor Bits 47-31 BIOS vendor Bit 30 1 INT 10h CGA Bit 29 1 INT 17h Parallel Port supported. Bit 28 1 INT 14h Serial Port Bit 27 1 INT 09h Keyboard Service supported. Bit 26 1 INT 05h Print Screen supported. Bit 25 1 INT 13h 3½" 2.88 MB floppy. Bit 24 1 INT 13h 3½" 720 KB floppy. Bit 23 1 INT 13h 5¼" 1.2 MB floppy. Bit 22 1 INT 13h 5¼" 360 KB floppy. Bit 21 1 INT 13h Toshiba 3½" 1.2 MB (1 KB /sector, 360RPM). Bit 20 1 INT 13h NEC 9800 3½" 1.2 MB (1KB /sector, 360 RPM) Bit 19 1 EDD (Enhanced Disk Drive) Bit 18 1 Boot from PC Card Bit 17 1 socket BIOS ROM Bit 16 1 Selectable boot Bit 15 1 Boot from CD-ROM Bit 14 1 ESCD supported. Bit 13 1 VESA™ VL-Bus™ Bit 12 1 BIOS shadowing Bit 11 1 BIOS flash ROM Bit 10 1 APM is supported. Bit 9 1 PnP Bit 8 1 PCMCIA Bit 7 1 PCI Bit 6 1 EISA Bit 5 1 MCA Bit 4 1 ISA Bit 3 1 BIOS characteristics not supported. Bit 2 1 Unknown
11h	BIOS size	Word	Varies	The size is 16 KB times (n+1)

DMI System Information (Type 1)

The information in this structure defines the system attributes. This table is associated with the Component ID group of the MIF in the computer.

Offset	Name	Size	Value	Description
00h	Type	Byte	01h	Component ID information indicator
01h	Length	Byte	08h	Length in bytes
02h	Handle	Word	Varies	
04h	Manufacturer	Byte	Varies	Number of null-terminated string
05h	Product name	Byte	Varies	Number of null-terminated string
06h	Version	Byte	Varies	Number of null-terminated string
07h	Serial number	Byte	Varies	Number of null-terminated string

DMI Motherboard Information (Type 2)

The information in this structure defines the motherboard attributes.

Offset	Name	Size	Value	Description
00h	Type	Byte	02h	Motherboard information indicator
01h	Length	Byte	09h	
02h	Handle	Word	Varies	
04h	Manufacturer	Byte	Varies	Number of null-terminated string
05h	Product name	Byte	Varies	Number of null-terminated string
06h	Version	Byte	Varies	Number of null-terminated string
07h	Serial number	Byte	Varies	Number of null-terminated string
08h	POST error status	Word	Bit field	Specifies the POST errors: Bits 15-9 Reserved. Must be zero. Bit 8 1 RAM test failed. Bit 7 1 Keyboard test failed. Bit 6 1 Power management initialization failed. Bit 5 1 Video BIOS failed. Bit 4 1 CMOS checksum failed. Bit 3 1 BIOS ROM checksum failed. Bit 2 1 Error reporting not supported. Bit 1 Unknown Bit 0 Other

DMI Chassis Information (Type 3)

The information in this DMI structure defines the computer's mechanical enclosure. If a computer has a separate enclosure for peripheral devices, two Type 3 DMI structures are returned: one for the main computer and one for the peripheral enclosure.

Offset	Name	Size	Value	Description
00h	Type	Byte	03h	System enclosure indicator
01h	Length	Byte	09h	
02h	Handle	Word	Varies	
04h	Manufacturer	Byte	Varies	Number of null-terminated string
05h	Type	Byte	Varies	Bit 7 1 Chassis lock present Bits 6-0 Chassis type 01h Other 02h Unknown 03h Desktop 04h Low profile desktop 05h Pizza box 06h Minitower 07h Tower 08h Portable 09h Laptop 0Ah Notebook 0Bh Handheld 0Ch Docking station 0Dh All-in-one 0Eh Subnotebook 0Fh Space-saving 10h Lunchbox 11h Main server chassis 12h Expansion chassis 13h Subchassis 14h Bus expansion chassis 15h Peripheral chassis 16h RAID chassis 17h Rack-mount chassis
06h	Version	Byte	Varies	Number of null-terminated string
07h	Serial number	Byte	Varies	Number of null-terminated string
08h	Asset tag number	Byte	Varies	Number of null-terminated string

DMI Processor Information (Type 4)

The information in this structure defines the attributes of a single processor. If this structure is used for a multiprocessor computer, a separate DMI structure is provided for each processor in the computer. A second DMI structure is provided for any coprocessor.

Offset	Name	Size	Value	Description
00h	Type	Byte	04h	Processor information indicator
01h	Length	Byte	1Ah	
02h	Handle	Word	Varies	
04h	Socket Designation	Byte	Varies	String number for reference designation
05h	Processor Type	Byte	ENUM	01h Other 02h Unknown 03h Central processor 04h Math coprocessor 05h DSP processor 06h Video processor
06h	Processor family	Byte	ENUM	01h Other 02h Unknown 03h Intel 86 04h Intel 286 05h Intel 386 06h Intel 486 07h Intel 8087 08h Intel 287 09h Intel 387 0Ah Intel 487 0Bh Intel Pentium family 0Ch Intel Pentium Pro family 0Dh Intel OverDrive family (New CPU core technologies adapted to existing sockets). 0Eh Intel Pentium II 20h PowerPC family
07h	Processor manufacturer	Byte	Varies	String number of processor manufacturer
08h	Processor ID	Qword	Varies	Raw processor identification data. For the Intel x86 CPUs, the format of this field depends on the processor support for the CPUID instruction. If CPUID is supported, this field contains two doubleword-formatted values The first value (offset 08h-0Bh) is the EAX value returned by a CPUID instruction with EAX set to 1. The second value (offset 0Ch - 0Fh) is the EDX value returned by CPUID.
10h	Processor version	Byte	Varies	String number that describes the processor

Offset	Name	Size	Value	Description
11h	Voltage	Byte	Varies	Bits 7-4 Reserved. Must be zero. Bit 3 Reserved. Must be zero. Bit 2 1 2.9V CPU Bit 1 1 3.3V CPU Bit 0 1 5V CPU The CPU socket voltage is configurable if more than one of the bits from 2-0 are set.
12h	External clock	Word	Varies	The external clock frequency. Zero if the value is unknown.
14h	Maximum speed	Word	Varies	99h for a 99 MHz processor. Zero if the value is unknown.
16h	Current speed	Word	Varies	99h for a 99 MHz processor. Zero if the value is unknown.
18h	Status	Byte	Varies	Bit 7 Reserved. Must be zero. Bit 6 CPU socket populated 0 CPU socket filled 1 CPU socket empty. Bits 5-3 Reserved. Must be zero. Bits 2-0 CPU status 000 Unknown 001 CPU enabled 010 CPU disabled by BIOS Setup Utility 011 CPU disabled, POST 100 CPU idle. Waiting to be enabled.
19h	Processor upgrade	Byte	ENUM	01h Other 02h Unknown 03h Daughterboard 04h ZIF socket 05h Replaceable piggyback 06h None

DMI Memory Controller Information (Type 5)

The information in this DMI structure defines the memory controller and memory module attributes. Memory modules not controlled by the memory controller are not included.

Offset	Name	Size	Value	Description
00h	Type	Byte	05h	Memory controller indicator
01h	Length	Byte	Varies	14 + (2 times the number of the associated memory slot), offset 0Dh.
02h	Handle	Word	Varies	
04h	Error detecting method	Byte	ENUM	01h Other 02h Unknown 03h None 04h 8-bit parity 05h 32-bit ECC 06h 64-bit ECC 07h 128-bit ECC
05h	Error correcting capability	Byte	Bit field	Bit 5 1 Error scrubbing. Bit 4 1 Double bit error correcting Bit 3 1 Single bit error correcting Bit 2 1 None Bit 1 1 Unknown Bit 0 1 Other
06h	Supported interleave	Byte	ENUM	01h Other 02h Unknown 03h One way interleave 04h Two way interleave 05h Four way interleave 06h Eight way interleave 07h Sixteen way interleave
07h	Current Interleave	Byte	ENUM	01h Other 02h Unknown 03h One way interleave 04h Two way interleave 05h Four way interleave 06h Eight way interleave 07h Sixteen way interleave
08h	maximum memory module size	Byte	Varies (n)	The size of the largest memory module supported (per slot). Specifies as n, where 2**n is the maximum size in MB. The maximum amount of memory supported by this controller is 2**n times the number of slots, as specified in 0Dh Number of associated memory slots.
09h	Supported speeds	Byte	Bit field	Bits 7-5 Reserved. Must be zero. Bit 4 50 ns Bit 3 60 ns Bit 2 70 ns Bit 1 Unknown Bit 0 Other

Offset	Name	Size	Value	Description
0Ah	Supported memory types	Word	Bit field	Bits 15-9 Reserved. Must be zero. Bit 8 DIMM Bit 7 SIMM Bit 6 ECC Bit 5 Parity Bit 4 EDO Bit 3 Fast page mode Bit 2 Standard Bit 1 Unknown Bit 0 Other
0Ch	Memory module voltage	Byte	Varies	This field specifies the required voltages for each memory socket controlled by this controller Bits 7-3 Reserved. Must be zero. Bit 2 1 2.9V Bit 1 1 3.3V Bit 0 1 5V The voltages for each socket are configurable if more than one bit of bits 2-0 are set.
0Dh	Number of associated memory slots	Byte	Varies	This field defines the number of memory module information blocks controlled by this memory controller.
0Eh	Memory module configuration handle	Word	Varies	This field is a memory information structure index controlled by this controller. The value in *offset 0Dh Number of associated memory slots* defines the count.

DMI Memory Module Information (Type 6)

The information in this DMI structure defines memory module attributes. One of these DMI structures is included for each memory module socket in the computer. This structure describes the speed, type, size, and error status of each system memory module. The supported attributes of each module are described in the DMI memory controller information structure that owns this module.

Offset	Name	Size	Value	Description
00h	Type	Byte	06h	Memory module configuration indicator
01h	Length	Byte	0Dh	
02h	Handle	Word	Varies	
05h	Socket designation	Byte	Varies	Bits 7-4 Memory bank (RAS#) connection. Bits 3-0 Different memory bank (RAS#) connection. 0Fh No memory bank connection. For example: if memory banks 1 and 3 (RAS#1 and RAS#3) were connected to SIMM sockets, the value in this field would be 13h. If only bank2 (RAS#2) is connected, this field would contain 2Fh.
06h	Current speed	Byte	Bit field	Bits 7-5 Reserved. Must be zero. Bit 4 50 ns Bit 3 60 ns Bit 2 70 ns Bit 1 Unknown Bit 0 Other
07h	Upgrade speed	Byte	Bit field	Bits 7-5 Reserved. Must be zero. Bit 4 50 ns Bit 3 60 ns Bit 2 70 ns Bit 1 Unknown Bit 0 Other
08h	Current Memory Type	Word	Bit field	Bits 15-9 Reserved. Must be zero. Bit 8 DIMM Bit 7 SIMM Bit 6 ECC Bit 5 Parity Bit 4 EDO Bit 3 Fast page mode Bit 2 Standard Bit 1 Unknown Bit 0 Other

Offset	Name	Size	Value	Description
0Ah	Installed size	Byte	Varies (n)	This field specifies the size of the memory module installed in the memory socket. If the computer cannot detect memory, this field is set to 7Dh. Bit 7 Type of connector 0 Single bank 1 Double bank Bits 6-0 Size(n), where 2**n is the size in MB. There are three special cases: 7Dh Memory cannot be determined. 7Eh Memory module is installed, but not enabled. 7Fh Not installed.
0Bh	Enabled Size	Byte	Varies	The amount of memory currently enabled. If a memory module has been installed in the socket but all memory in the module has been disabled, this field is set to 7Eh. Bit 7 Type of connector 0 Single bank 1 Double bank Bits 6-0 Size(n), where 2**n is the size in MB. The special cases: 7Eh Memory module installed but not enabled. 7Fh Not installed.
0Ch	Error status	Byte	Varies	Bits 7-2 Reserved. Must be set to zero. Bit 1 1 Correctable errors received for the module Bit 0 Uncorrectable errors received for the module. All or part of the module has been disabled.

DMI Cache Information Structure (Type 7)

The information in this structure defines the cache memory attributes. One of these DMI structures is specified for each cache memory device in the computer, no matter if the cache memory is internal or external to the CPU module. Cache modules can be associated with a processor structure.

Offset	Name	Size	Value	Description
00h	Type	Byte	07h	Cache memory information indicator
01h	Length	Byte	0Fh	
02h	Handle	Word	Varies	
04h	Socket designation	Byte	Varies	This field contains a string number for reference designation.
05h	Cache configuration	Word	Varies	Bits 15-10 Reserved. Must be zero. Bits 9-8 Operational mode 00 Write-through 01 Write-back 10 Varies with memory address 11 Unknown Bit 7 Status at boot time 0 Disabled 1 Enabled Bits 6-5 Location relative to CPU module 00 Internal 01 External 10 Reserved 11 Unknown Bit 4 Reserved. Must be zero. Bit 3 Cache socketed 0 Not socketed 1 Socketed Bits 2-0 Cache Level 000 Level 1 001 Level 2 010 Level 3 011 Level 4 100 Level 5 101 Level 6 110 Level 7 111 Level 8
07h	Maximum cache size	Word	Varies	This field contains the maximum cache memory that can be installed. Bit 15 Granularity 0 1 KB 1 64 KB Bits 14-0 Maximum size in units equal to the cache memory granularity.
09h	Installed size	Word	Varies	This field contains the installed cache memory size. Bit 15 Granularity 0 1 KB 1 64 KB Bits 14-0 Maximum size in units equal to the cache memory granularity.

Offset	Name	Size	Value	Description
0Bh	Supported SRAM type	Word	Bit field	Bits 15-7 Reserved. Must be zero. Bit 6 Asynchronous Bit 5 Synchronous Bit 4 Pipeline burst Bit 3 Burst Bit 2 Non-burst Bit 1 Unknown Bit 0 Other
0Dh	Current SRAM type	Word	Bit field	Bits 15-7 Reserved. Must be zero. Bit 6 Asynchronous Bit 5 Synchronous Bit 4 Pipeline burst Bit 3 Burst Bit 2 Non-burst Bit 1 Unknown Bit 0 Other

DMI Port Connector Information (Type 8)

The information in this structure defines the attributes of a serial, parallel, mouse, or keyboard port connector. One of these DMI structures is present for each port in the computer. For example: if there are three serial ports, there will be three separate DMI structures, one for each serial port.

Offset	Name	Size	Value	Description
00h	Type	Byte	08h	Connector information indicator
01h	Length	Byte	07h	
02h	Handle	Word	Varies	
04h	Reference Designator	Byte	Varies	This field contains a string number for reference designation, for example: SERIAL1.
05h	Connector type	Byte	ENUM	Connector type 01h Centronics 02h Minicentronics 03h Proprietary 04h Male DB25 05h Female DB25 06h Male DB15 07h Female DB15 08h Male DB9 09h Female DB9 0Ah RJ11 0Bh RJ45 0Ch 50-pin miniSCSI 0Dh MiniDIN 0Eh MicroDIN 0Fh PS/2 10h Infrared 11h HP-HIL 12h Access bus 13h SSA SCSI 14h Male circular 8-pin DIN 15h Female circular 8-pin DIN

Offset	Name	Size	Value	Description
06h	Port type	Byte	ENUM	Port type 00h None 01h XT/AT-compatible parallel port 02h PS/2 Parallel Port 03h ECP parallel port 04h EPP parallel port 05h ECP/EPP parallel port 06h XT/AT-compatible serial port 07h 16450-compatible serial port 08h 16550-compatible serial port 09h 16550A-compatible serial port 0Ah SCSI port 0Bh MIDI port 0Ch Joystick port 0Dh Keyboard port 0Eh Mouse port 0Fh SSA SCSI 10h USB 11h Firewire (IEEE P1394) 12h PCMCIA Type I PC Card 13h PCMCIA Type II PC Card 14h PCMCIA Type III PC Card 15h CardBus 16h Access bus port

DMI System Slots (Type 9)

The information in this structure defines the computer expansion slot attributes. One DMI structure is provided for each expansion slot in the computer.

Offset	Name	Size	Value	Description
00h	Type	Byte	09h	Expansion slot structure indicator
01h	Length	Byte	09h	
02h	Handle	Word	Varies	
04h	Slot Designation	Byte	Varies	This field contains a string number for reference designation, for example: PCI-1.
05h	Slot type	Byte	ENUM	Slot type 01h Other 02h Unknown 03h ISA 04h MCA 05h EISA 06h PCI 07h PCMCIA 08h VESA VL-Bus 09h Proprietary 0Ah Processor card slot 0Bh Proprietary memory card slot 0Ch I/O riser card slot 0Dh NuBus
06h	Slot data bus width	Byte	ENUM	Slot data bus width 01h Other 02h Unknown 03h 8-bit 04h 16-bit 05h 32-bit 06h 64-bit 07h 128-bit
07h	Current usage	Byte	Byte	Expansion slot current usage 01h Other 02h Unknown 03h Available 04h In use
08h	Slot length	Byte	ENUM	Expansion slot length 01h Other 02h Unknown 03h Half-length 04h Full-length

DMI Onboard Device Information (Type 10)

The information in this structure defines the onboard device attributes. Many devices (such as video, SCSI, IDE, network) are mounted on the motherboard and thus are onboard devices.

Offset	Name	Size	Value	Description
00h	Type	Byte	10h	Onboard device information indicator
01h	Length	Byte	Varies	4 + (number of devices times 2).
02h	Handle	Word	Varies	
04h	Device 1 type	Byte	Varies	Bit 7 Device 1 status 0 Device disabled 1 Device enabled Bits 6-0 Device Type 01h Other 02h Unknown 03h Video 04h SCSI 05h Ethernet 06h Token ring 07h Sound
05h	Description string	Byte	Varies	This field contains the number of string that describes Device 1.
06h	Device 2 type	Byte	ENUM	Bit 7 Device 2 status 0 Device disabled 1 Device enabled Bits 6-0 Device Type 01h Other 02h Unknown 03h Video 04h SCSI 05h Ethernet 06h Token ring 07h Sound
07h	Description string	Byte	Varies	This field contains the number of string that describes Device 2.

DMI OEM Strings (Type 11)

Offset	Name	Size	Value	Description
00h	Type	Byte	11h	OEM string information indicator.
01h	Length	Byte	05h	
02h	Handle	Word	Varies	
04h	Count	Byte	Varies	Number of strings

DMI System Configuration Options (Type 12)

Offset	Name	Size	Value	Description
00h	Type	Byte	12h	Configuration information indicator
01h	Length	Byte	05h	
02h	Handle	Word	Varies	
04h	Count	Byte	Varies	Number of strings

DMI BIOS Language Information (Type 13)

Offset	Name	Size	Value	Description
00h	Type	Byte	12h	Language information indicator
01h	Length	Byte	06h	
02h	Handle	Word	Varies	
04h	Installable language	Byte	Varies	This field contains the number of languages available. Each available language has a description string. This field contains the number of strings that follow.
05h	Current language	Byte	Varies	This field contains a string number of the currently installed language.

DMI Group Associations (Type 14)

Offset	Name	Size	Value	Description
00h	Type	Byte	14h	Group association indicator
01h	Length	Byte	Varies	5 + (Three bytes for each item in the group).
02h	Handle	Word	Varies	
04h	Group name	Byte	Varies	This field contains the string number of the text string that describes the group.
05h	Item type	Byte	Varies	This field contains the item (structure type) of this member.
06h	Item handle	Word	Varies	This field contains the handle that corresponds to the structure.

DMI System Event Log (Type 15)

If a DMI System Event Log is included in the DMI data, the computer supports an event log. An event log is a fixed-length area in NVRAM. The event log begins with a vendor-specific fixed-length header record, followed by one or more variable-length log records.

Offset	Name	Size	Value	Description
00h	Type	Byte	15h	Event log type indicator
01h	Length	Byte	14h	This field contains the length of this structure, including the Type and Length fields.
02h	Handle	Word	Varies	This field contains the handle (instance number) associated with the record.
04h	Log area length	Word	Varies	This field contains the length (in bytes) of the total event log area, from the first byte of the header through the last byte of data.
06h	Log header start offset	Word	Varies	This field specifies the starting offset (or index) of the event log header in NVRAM. The Access Method Address field at offset 0Ah in this DMI structure defines the method used to access the event log area. The most significant byte of the start offset is set to 00h for single-byte indexed I/O accesses. This field is set to 00h if the event log area has no header.

Offset	Name	Size	Value	Description
08h	Log data start offset	Word	Varies	This field specifies the starting offset (or index) of the first data byte in the event log in NVRAM. The Access Method Address field at offset 0Ah in this DMI structure defines the method used to access the event log area. The most significant byte of the start offset is set to 00h for single-byte indexed I/O accesses. The data directly follows the header information. The header length is determined by subtracting the Log Header Start Offset value (in offset 06h) from the Log Data Start Offset value (at offset 08h) in this DMI structure.
0Ah	Access Method	Byte	Varies	This field specifies the method used to access the event log area: 00h Indexed I/O using a one-byte index and a one-byte data field. The Access Method Address field at offset 10h contains the one-byte index I/O address followed by the one-byte data I/O address. 01h Indexed I/O using a two-byte index and a one-byte data field. The Access Method Address field at offset 10h contains the two-byte index I/O address (in Intel Word format) followed by the one-byte data I/O address. 02h Memory-mapped physical 32-bit address. The Access Method Address field at offset 10h contains the four-byte index I/O address (in Intel Dword format) starting physical address. 03h Available via Function 55h Get General Purpose NVRAM Information. 04h-FFh Reserved for future use.
0Bh	Log Status	Byte	Varies	The bit field specifies the current status of the system event log. Bits 7-2 Reserved. Must be zero. Bit 1 1 Log area full Bit 0 1 Log area valid
0Ch	Reserved for future use	Qword	Zeros	These four bytes are reserved for future use.

Offset	Name	Size	Value	Description
10h	Access Method Address	Dword	Varies	This field specifies the address associated with the access method. The data present depends on the value in the Access Method field at offset 0Ah in this DMI structure. The format of the area is described by the following packed C union: `union { struct { char Index; char Data; } IOSingleByte; struct { int Index; char Data; } IODoubleByte; long PhysicalAddr32; } AccessMethodAddress;`

DMI System Event Log

DMI Event Log Organization The event log includes an optional implementation-specific fixed-length header and one or more variable-length event records, as shown below.

The format of the event log header can change. The size of the total event log area can also change. All event log area fields will be consistent across all applications on all computers. The event log area operates as a write-once clear-all log.

Optional Log Header (eight bytes)								
Type	**Length**	**Year**	**Month**	**Day**	**Hour**	**Minute**	**Second**	**Log Variable Data**

All fields are required except the Log Variable Data field.

Cont'd

DMI Event Log Record Header Each event log record includes a required fixed-length header followed by optional data defined by the event type. The fixed-length log record header is in the first eight bytes of every log record. The log header:

Offset	Field	Format	Description
00h	Event type	Byte	The event log types are: 00h Reserved 01h Single-bit ECC memory error 02h Multi-bit ECC memory error 03h Parity memory error 04h Bus timed out 05h I/O channel check 06h Software NMI 07h POST memory resize 08h POST error 09h PCI parity error 0Ah PCI system error 0Bh CPU failure 0Ch Fail-Safe timer timed out 0Dh Correctable memory log disabled 0Eh Error logging disabled 0Fh Reserved 10h System limit exceeded 11h Asynchronous system reset 12h System configuration information 13h Hard disk information 14h System reconfigured 15h Uncorrectable CPU complex error 16h Log area reset and cleared 17h System boot 18-7Fh Reserved for future use. 80-FFh Available for system- and OEM-specific assignments.
01h	Length	Byte	The byte length of the event record, including the record type and length fields. If the most significant bit of the field is 0, the record has been read. If it is 1, the record has not been read. If the field has been read, it likely y has been processed by a higher-level software layer.
02h-07h	Date and time fields	Byte	The BCD representation of the date and time of the most recent occurrence of the logged event as read from NVRAM. The format of this field is: year, month, day, hour, minute, and second.
08h Plus	Log variable data	Varies	Optional event-specific additional status information.

13 AMIBIOS Multiprocessor Support

AMIBIOS with a BIOS date of 10/10/94 or later support multiprocessor computers that use the ISA, EISA, VL-Bus, or PCI buses.

APIC Support

Interrupt handling in a multiprocessor ISA computer cannot be handled properly by the standard Intel 8259A Programmable Interrupt Controllers. A multiprocessor system needs the Intel 62489DX Advanced Programmable Interrupt Controller (APIC), or a compatible chip. APIC features include:

- 8259-compatible operation,
- Interprocessor Interrupts (IPIs),
- multiprocessor interrupt management using either static or dynamic distribution schemes,
- dynamic distribution via interrupt routing to the lowest priority CPU,
- support for several addressing schemes, and
- system-wide control functions (NMI, INIT, SMI, Start-Up Interprocessor Interrupt).

APIC Components The APIC architecture components are:

- a local APIC,
- an I/O APIC, and
- the Interrupt Controller Communications (ICC) bus.

Local APIC The local APIC dispatches interrupts to the processors. The Local APIC is programmed to accept the correct interrupts on the APIC bus. It provides interrupt queuing, nesting, and masking. It handles delivery protocols with its local processor and controls access to the APIC registers. The local APIC also manages Interprocessor interrupts and remote APIC register reads. Each local APIC includes a built-in timer and local interrupt pins that can handle processor-specific interrupts. The local APIC can be disabled in hardware or software. It can be used with a standard 8259A Programmable Interrupt Controller.

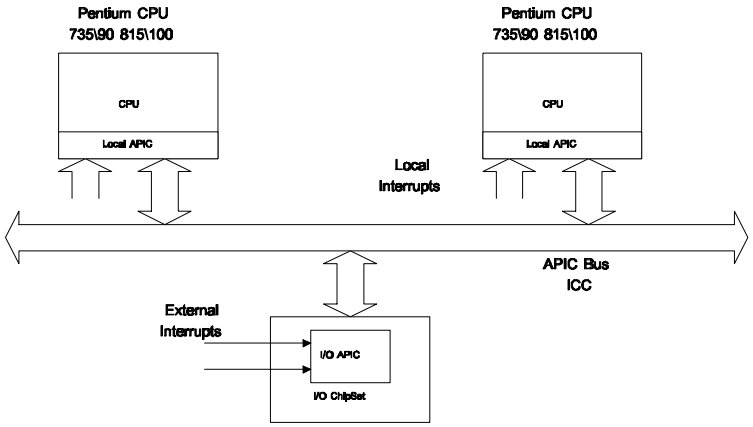
Cont'd

I/O APIC The I/O APIC captures interrupts from I/O devices. The I/O APIC enables the individual interrupt pins to be programmed by specifying:

- a vector and an implied priority,
- the sensitivity to be recognized as an edge- or level-triggered interrupt, and
- the set of processors that service the interrupt.

ICC Bus The Interrupt Controller Communications bus provides a pathway for interrupts between the CPUs and the APIC.

APIC Block Diagram The following diagram illustrates the APIC architecture in a dual Pentium computer:



Addresses The Local APIC and I/O APIC are memory-mapped as follows:

Device	Mapped to
Local APIC	FEE0:0000h to FEEF:FFFFh
I/O APIC	FEC0:0000h to FECF:FFFFh.

Number of APICs There must be one Local APIC per CPU. At least one I/O APIC is required. These requirements are essential for an APIC-based Symmetric Multiprocessing (SMP) computer.

Cont'd

APIC Support, Continued

Interrupt Modes

Mode	Description
PIC Mode	This mode effectively bypasses all APIC components and forces the computer to operate in single-processor mode.
Virtual Wire Mode	This mode uses an APIC as a virtual wire but otherwise operates like PIC mode.
Symmetric I/O Mode	This mode allows the computer to operate with multiple processors.

Intel Multiprocessor Specification

The Intel Multiprocessor Specification (MPS) defines an enhancement to the ISA standard. A multiprocessing operating system can run on an MPC-compliant computer with no customization. MPS applies to IBM PC/AT®-compatible (ISA) multiple CPU designs based on the Intel x86 processor and APIC architectures.

Compatibility MPS maintains ISA software compatibility and provides a way for a multiprocessing operating system to utilize the resources of more than one CPU.

Floating Point Table The MPS Floating Pointer structure must be implemented. See the Intel MPS Specification Rev 1.1 for information on implementing this AMIBIOS table. A multiprocessing operating system searches for this structure in predefined locations. This structure specifies if the Multiprocessor Configuration Table is present.

Cont'd

Intel Multiprocessor Specification, Continued

MPS Features MPS features include:

Feature	Description
A multiprocessor extension to ISA that runs all existing single processor shrink-wrapped binaries and all multiprocessor binaries.	MPS extends the performance of the ISA platform. MPS defines a way for a multiprocessor bus to coexist with ISA and EISA buses and provide hooks in the firmware for multiprocessing operating systems.
Symmetric multiprocessing support for one or more Intel x86 (I486 and up) CPUs.	The MPS specification supports all x86 binary-compliant CPUs.
Can use a BIOS with minimal multiprocessing-specific support.	The BIOS as minimal extension to support multiprocessor computers that do not conflict with single processor system BIOS functions.
Optional multiprocessing configuration table to communicate configuration information to a multiprocessing operating system.	MPS defines the following configuration tables so that an operating system can access resource information: The Multiprocessor Floating Point Structure. This table is required for MPS compliance. The Multiprocessor Configuration Table.
Incorporates ISA buses in multiprocessor computers.	The multiprocessor bus is a system bus that can handle more than one processor. ISA and EISA buses are attached to the multiprocessor bus via a bridge.
Secondary cache and memory bus implementations are transparent to software.	Other non-MPS multiprocessor operating systems require special cache and memory considerations to make a computer transparent to software.

Default Configuration The Default Configuration is defined in MPS V1.1. It is a two-processor computer that can also have an Integrated or Discrete APIC and an ISA bus.

Non-Default Configuration The Non Default configuration includes all other multiprocessor computers. The multiprocessor Configuration Table must be implemented in a Non-Default Configuration because the multiprocessor Floating Point Structure cannot transmit the non-default configuration information.

MPS System BIOS Multiprocessor Support

The AMIBIOS system BIOS programming guidelines for MPS V1.1 are:

- implement a Multiprocessor Floating Point Structure,
- implement a Multiprocessor Configuration Table if a non-default MPC configuration is used, and
- initialize the APIC to Virtual Wire Mode.

Dual Processor Issues Dual processor computers are defined as Default Configurations.

Bus Support The bus configuration in a multiprocessor computer that meets the MPC specifications can be: PCI and ISA, or PCI and EISA.

Floating Point Structure AMIBIOS first builds the MPC Floating Point Structure. The table must be available in the BIOS runtime after the operating system boots. For EISA and PCI computer, the table structure is:

Offset	Default	Description
00h	_MP_	MPC signature.
01h	0000h	Offset part of Multiprocessing Configuration Table address.
03h	0000h	Segment part of Multiprocessing Configuration Table address.
05h	01h	Length of this table in 16-byte blocks.
06h	01h	MPC Specification version Number.
07h	9Dh	Structure checksum. (9Eh for PCI ISA dual CPU configuration).
08h	06h	MPC Feature Info Byte 1 06h PCI and EISA bus with dual CPU configuration. 05h PCI and ISA bus with dual CPU configuration.
09h	00h	MPC Feature Info Byte 2
0Ah	00h	MPC Feature Info Byte 3
0Bh	00h	MPC Feature Info Byte 4
0Ch	00h	MPC Feature Info Byte 5
0Dh	00h	Reserved for future use.

Cont'd

The 735/90 and 815/100 Pentium CPUs support a private interface that acts like a multiprocessing bus to facilitates the design of dual processor computers that use the P54C and P54CM CPUs.

The chipsets that support dual processor computers include the:

- Intel Neptune and the
- VLSI 590.

No special BIOS considerations are necessary for chipset programming. In the VLSI 590 chipset, the I/O APIC is on the chipset. The I/O APIC can be disabled by programming a register. You cannot disable the I/O APIC on the Intel Neptune EISA or ISA chipsets.

Initialize Local APIC The BIOS must initialize the Local APIC on the primary processor to Virtual Wire Mode. When the Local APIC is set to Virtual Wire Mode, the computer is in 8259 compatible mode. If necessary, the operating system programs the modes in the APICs in both processors.

Warning

The Intel MPS V1.1 specification says that it is not necessary for the Local APIC to be programmed to Virtual Wire Mode, but if the APIC has been enabled in hardware, it must be initialized to Virtual Wire Mode or the computer hangs as soon as interrupt-oriented I/O begins or interrupts are enabled. There is a workaround with the P54x B stepping CPUs. Call American Megatrends for additional information.

Cont'd

Detecting Multiple Processors AMIBIOS uses the APIC Interprocessor Interrupt (IPI) to detect two processors. The APIC supports the IPI so that the CPUs in the multiprocessor computer can interrupt each other.

> ***Warning***
>
> After initializing the dual processor, system software must issue an INIT IPI to the dual processor to return it to Reset state. Otherwise, Windows NT V3.5 will hang.

Non-Default Configurations Non-Default configurations include all multiprocessor computers not included under Default Configurations. For Example: the LSI Logic chipset can support up to six P54C CPUs on their proprietary MPI multiprocessor bus. For the Hydra chipset, AMIBIOS must create a Multiprocessor Configuration Table. This table includes entries for all processors, buses, I/O APICs, and interrupt assignments read by a multiprocessing operating system.
At reset, AMIBIOS might be required to arbitrate for the bootstrap processor and place all other processors in Sleep mode. Only one processor should be executing the system BIOS.

Cont'd

Corollary C-Bus AMIBIOS also supports other multiprocessor specifications.

Other vendors introduced successful ISA DOS multiprocessor computers before Intel defined the MPS. Corollary specified the C-Bus multiprocessor bus and designed their own multiprocessor-oriented interrupt architecture. The C-Bus specifications are proprietary. Corollary had an i486-based multiprocessor computer. Corollary and The Santa Cruz Operation developed the SCO MPX multiprocessing kernel.

C-Bus II is the latest version. It supports the Intel P54C and PCI local bus. AMIBIOS has been ported to the Intergraph Eagle C-Bus II computer. This specification supports computers computer with up to six Intel P54C CPUs on the PCI and EISA buses and up to 512 MB of DRAM.

AAMIBIOS Error Messages and Beep Codes

Beep Codes

Errors can occur during POST (Power On Self Test), which is performed every time the computer is powered on. Fatal errors (see below) are communicated through a series of audible beeps. All errors except Beep Code 8 are fatal errors. Fatal errors do not allow the computer to continue the boot process. Most displayed errors allow the computer to continue the boot process.

Beeps	Error message	Description
1	Refresh Failure	The memory refresh circuitry on the motherboard is faulty.
2	Parity Error	Parity error in the first 64 KB of memory.
3	Base 64 KB Memory Failure	Memory failure in first 64 KB.
4	Timer Not Operational	Memory failure in the first 64 KB of memory, or Timer 1 on the motherboard is not functioning.
5	Processor Error	The CPU (Central Processing Unit) on the motherboard generated an error.
6	8042 – Gate A20 Failure	The keyboard controller (8042) may be bad. The BIOS cannot switch to protected mode.
7	Processor Exception Interrupt Error	The CPU generated an exception interrupt.
8	Display Memory Read/Write Error	The system video adapter is either missing or its memory is faulty. This is not a fatal error.
9	ROM Checksum Error	The ROM checksum value does not match the value encoded in the BIOS.
10	CMOS Shutdown Register Read/Write Error	The shutdown register for CMOS RAM failed.
11	Cache Error/ External Cache Bad	The external cache is faulty.

Displayed Fatal Error Messages

The computer halts after the following messages and cannot be rebooted until a physical change is made.

- CMOS not operational
- 8042-Gate A20 error
- DMA error
- DMA #1 error
- DMA #2 error
- FDD Controller failure
- HDD Controller failure
- INTR #1 error
- INTR #2 error
- On Board parity error
- On Board Parity error

AMIBIOS POST Error Message Format

```
ERROR Message Line 1
ERROR Message Line 2
Press <F1> to RESUME
```

The <F1> prompt message is not displayed if *Wait for <F1> If Any Error* in Advanced Setup has been set to *Disabled.* For most displayed error messages, there is only one message. If a second message appears, it is

```
RUN SETUP
```

press <F1> to run AMIBIOS Setup.

Error Message	Explanation
8042 Gate – A20 Error	Gate A20 on the keyboard controller (8042) is not working. Replace the 8042.
Address Line Short!	Error in the address decoding circuitry on the motherboard.
C: Drive Error	Hard disk drive C: does not respond. Run AMIDiag to correct this problem. Also, check the C: hard disk type in Standard Setup to make sure that the hard disk drive type is correct.
C: Drive Failure	Hard disk drive C: does not respond. Replace the hard disk drive.
Cache Memory Bad, Do Not Enable Cache!	Cache memory is defective. Replace it.
CH-2 Timer Error	Most ISA computers include two timers. There is an error in timer 2.
CMOS Battery State Low	CMOS RAM is powered by a battery. The battery power is low. Replace the battery.
CMOS Checksum Failure	After CMOS RAM values are saved, a checksum value is generated for error checking. The previous value is different from the current value. Run AMIBIOS Setup.
CMOS System Options Not Set	The values stored in CMOS RAM are either corrupt or nonexistent. Run AMIBIOS Setup.
CMOS Display Type Mismatch	The video type in CMOS RAM does not match the type detected by the BIOS. Run AMIBIOS Setup.
CMOS Memory Size Mismatch	The amount of memory on the motherboard is different than the amount in CMOS RAM. Run AMIBIOS Setup.
CMOS Time and Date Not Set	Run Standard Setup to set the date and time.

Error Message	Explanation
D: Drive Error	Hard disk drive D: does not respond. Run AMIDiag. Also check the D: hard disk type in Standard Setup to make sure that the hard disk drive type is correct.
D: drive failure	Hard disk drive D: does not respond. Replace the hard disk drive.
Diskette Boot Failure	The boot disk in floppy drive A: is corrupt. It cannot be used to boot the computer. Use another boot disk and follow the screen instructions.
Display Switch Not Proper	Some computers require a video switch on the motherboard be set to either color or monochrome. Turn the computer off, set the switch, then power on.
DMA Error	Error in the DMA controller.
DMA #1 Error	Error in the first DMA channel.
DMA #2 Error	Error in the second DMA channel.
FDD Controller Failure	The BIOS cannot communicate with the floppy disk drive controller. Check all appropriate connections after the computer is powered down.
HDD Controller Failure	The BIOS cannot communicate with the hard disk drive controller. Check all appropriate connections after the computer is powered down.
INTR #1 Error	Interrupt channel 1 failed POST.
INTR #2 Error	Interrupt channel 2 failed POST.
Invalid Boot Diskette	The BIOS can read the disk in floppy drive A:, but cannot boot the computer. Use another boot disk.
Keyboard Is Locked...Unlock It	The keyboard lock on the computer is engaged. The computer must be unlocked to continue.
Keyboard Error	Keyboard timing problem. Set the *Keyboard* option in Standard Setup to *Not Installed* to skip the keyboard POST routines.
KB/Interface Error	There is an error in the keyboard connector.
No ROM BASIC	Cannot find a bootable sector on either disk drive A: or hard disk drive C:. The BIOS calls INT 18h which generates this message. Use a bootable disk.
Off Board Parity Error	Parity error in memory installed in an expansion slot. The format is: OFF BOARD PARITY ERROR ADDR (HEX) = (XXXX) *XXXX* is the hex address where the error occurred. Run AMIDiag to find and correct memory problems.
On Board Parity Error	Parity error in motherboard memory. The format is: ON BOARD PARITY ERROR ADDR (HEX) = (XXXX) *XXXX* is the hex address where the error occurred. Run AMIDiag to find and correct memory problems.
Parity Error ????	Parity error in system memory at an unknown address. Run AMIDiag to find and correct memory problems.

EISA BIOS Error Messages

Error Message	Explanation
EISA CMOS Checksum Failure	The EISA CMOS checksum is incorrect. The battery for EISA CMOS RAM must be replaced.
EISA CMOS not operational	A Read or Write error occurred in extended CMOS RAM. The battery must be replaced.
Expansion Board not ready at Slot *X, Y, Z*	Cannot find the adapter card in Slot *X*, *Y*, or *Z*. Make sure the card is in the correct slot and is properly seated.
Fail-Safe Timer NMI Inoperational	Devices that depend on the fail-safe NMI timer cannot operate correctly.
ID information mismatch for Slot *X, Y, Z*	The EISA adapter card ID in Slot *X*, *Y*, or *Z* does not match the ID in CMOS RAM. Run the ECU.
Invalid Configuration Information for Slot *X, Y, Z*	The configuration data for EISA adapter cards *X*, *Y*, or *Z* is incorrect. Run the ECU.
Software Port NMI Inoperational	The software port NMI is not working. You can continue but the computer halts when NMIs occur.

ISA NMI BIOS Messages

ISA NMI Message	Explanation
Memory Parity Error at *xxxxx*	Memory failed. If the memory location can be determined, it is displayed as *xxxxx*. If not, the message is *Memory Parity Error ????*.
I/O Card Parity Error at *xxxxx*	An expansion card failed. If the address can be determined, it is displayed as *xxxxx*. If not, the message is *I/O Card Parity Error ????*.
DMA Bus Time-out	A device has driven the bus signal for more than 7.8 microseconds.

EISA NMI BIOS Error Messages

EISA AMIBIOS can generate additional EISA-specific NMI messages. They are:

EISA NMI Message	Explanation
BUS Timeout NMI at Slot *n*	There was a Bus Timeout NMI at Slot *n*.
(E)nable (D)isable Expansion Board?	Type E to enable the adapter card that had an NMI or D to disable it.
Expansion Board Disabled at Slot *n*	The EISA adapter card in Slot *n* has been disabled.
Expansion Board NMI at Slot *n*	An adapter card NMI was generated from Slot *n*.
Fail-Safe Timer NMI	A fail-safe timer NMI has been generated.
Software Port NMI	A software port NMI has been generated.

B AMIBIOS Identification Strings

AMIBIOS has three BIOS Identification strings. Only String 1 appears at the bottom of the screen during boot-up. Press <Ins> to display Strings 2 and 3. The bytes of String 1 are numbered consecutively from left to right.

Byte	Description
1	Processor Type 0 8086 or 8088 2 80286 3 80386 4 80486 5 Pentium 6 Pentium Pro 7 Pentium II
2	Size of BIOS 0 64K BIOS 1 128K BIOS
4–5	Major Version Number
6–7	Minor Version Number
9–14	Reference Number
16	Halt on POST Error. Set to 1 if On.
17	Initialize CMOS in every boot. Set to 1 if On.
18	Block pins 22 and 23 of the keyboard controller. 0 Off 1 On
19	Mouse support in BIOS/keyboard controller. Set to 1 if On.
20	Wait for <F1> if error found. Set to 1 if On.
21	Display Floppy error during POST. Set to 1 if On.
22	Display Video error during POST. Set to 1 if On.
23	Display Keyboard error during POST. Set to 1 if On.
25–26	BIOS Date. Month (1-12).
27–28	BIOS Date. Date (1-31).
29–30	BIOS Date. Year (0-99).
32–39	Chipset Identification. BIOS Name.
41	Keyboard controller version number.

AMIBIOS Identification String Line 2

Byte	Description
1–2	Pin number for clock switching through keyboard controller.
3	Indicates High signal on pin switches clock to High(H) or Low (L).
5	Clock switching through chipset registers 0 No clock switching through chipset registers. 1 Clock switching through chipset registers.
7–10	Port address to switch clock high through special port.
12–13	Data value to switch clock high through special port.
15–16	Mask value to switch clock high through special port.
18–21	Port Address to switch clock low through special port.
23–24	Data value to switch clock low through special port.
26–27	Mask value to switch clock low through special port.
29–31	Pin number for Turbo Switch Input Pin

AMIBIOS Identification String Line 3

Byte	Description
1–2	Keyboard Controller Pin number for cache control.
3	Keyboard Controller Pin number for cache control. Indicates if High signal on the pin enables (H) or disable (L) cache.
5	1 The High signal is used on the Keyboard Controller pin.
7–9	Cache Control through Chipset Registers: 0 Cache Control off 1 Cache Control on
11–12	Port Address to enable cache through special port.
14–15	Data value to enable cache through special port.
17–20	Mask value to enable cache through special port.
22–23	Port Address to disable cache through special port.
25–26	Data value to disable cache through special port.
28–29	Mask value to disable cache through special port.
31	Pin number for resetting the memory controller.
33	BIOS Modified Flag. Incremented each time AMIBIOS is modified, from 1 to 9, then from A to Z, and then reset to 1. 0 AMIBIOS has not yet been modified. 1 AMIBIOS has been modified.

AMI BIOS and AMI BIOS Plus Identification Strings

AMI BIOS and AMI BIOS Plus were sold from 1986 through 1990. The general format of the BIOS Reference string in this type of AMI BIOS is:

```
Ref. TTTT-XXXX-042088-Kn
```

TTTT BIOS type
XXXX Customer number
042088 The BIOS release date
K*n* The keyboard BIOS version number. If *n* is 0, it is not an American Megatrends keyboard controller BIOS.

Index

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W