



**Personal System/2  
and Personal Computer  
BIOS Interface  
Technical Reference**

## **First Edition (September 1991)**

**The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.**

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Requests for technical information about IBM products should be made to your IBM Authorized Dealer or your IBM Marketing Representative.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

**© Copyright International Business Machines Corporation 1991. All rights reserved.**

**Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.**

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights or other legally protectible rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

**IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.**

The following terms, denoted by an asterisk (\*) in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

IBM  
Micro Channel  
Personal Computer  
PCjr  
Personal Computer XT  
PC/XT  
Personal Computer AT  
AT  
Personal System/2  
PS/2  
Gearbox  
Operating System/2

**The following term, denoted by a double asterisk (\*\*) in this publication, is a trademark of another company:**


**Intel** **Intel Corporation**





---

# Preface


 This technical reference provides information about the Basic Input/Output System (BIOS) and Advanced BIOS (ABIOS) interfaces. It is intended for developers who provide hardware or software products to operate with IBM\* products.

You should understand the concepts of computer architecture and programming before using this publication.

**Warning:** The term "Reserved" is used to describe certain signals, bits, and registers. Use of reserved areas can cause compatibility problems, loss of data, or permanent damage to the hardware.

This technical reference is divided into two parts: **BIOS** and **Advanced BIOS**.

**BIOS** contains the following:

 Section 1, "Introduction to BIOS," provides an overview of BIOS, interrupts, parameter passing, data areas, and read-only memory (ROM) tables. It also describes how to determine the system-BIOS version date.

Section 2, "Interrupts," contains detailed information about how interrupts function across the IBM Personal System/2\* and Personal Computer\* product lines. Exceptions between products are noted.

Section 3, "BIOS Data Areas," contains detailed information about regular data areas.

Section 4, "ROM Tables," contains detailed information about ROM tables for system- and adapter-ROM BIOS.

Section 5, "Additional Information," contains information about sharing interrupts in IBM Personal System/2 and Personal Computer products. It also contains information about adapter-ROM calls, video compatibility, and multitasking provisions.

 Section 6, "System Identification," contains information about system-identification bytes.

---

\* IBM, Personal System/2, and Personal Computer are trademarks of the International Business Machines Corporation.

Section 7, "Scan Code/Character Code Combinations," contains information about keyboard keys and scan code/character code combinations.

**Advanced BIOS** contains the following sections:

Section 1, "Introduction to Advanced BIOS," provides an overview of Advanced BIOS, data structures, initialization, transfer conventions, interrupt processing, and extending ABIOS.

Section 2, "Data Structures," contains detailed information on the common data area, function transfer tables, device blocks, and how these ABIOS data structures are used.

Section 3, "Initialization," describes the ABIOS steps and the operating-system steps that are necessary to make the ABIOS interface operational.

Section 4, "Transfer Conventions," describes the methods that are used to transfer control to ABIOS-device routines. The request block, the ABIOS transfer convention, and the operating-system transfer convention are described.

Section 5, "Additional Information," contains detailed information on interrupt processing; adding, patching, extending, and replacing ABIOS; and operating-system implementation considerations.

Section 6, "Interfaces," describes the interfaces that are supported by ABIOS.

System-specific hardware- and software-interface information for IBM systems and for IBM diskette drives, fixed disk drives, adapters, and other options is contained in separate technical reference publications.

# Contents—BIOS

<b>Section 1. Introduction to BIOS</b> .....	1-1
Interrupts .....	1-3
Parameter Passing .....	1-5
Data Areas and ROM Tables .....	1-5
BIOS-Level Determination .....	1-6
System Groups .....	1-6
 <b>Section 2. Interrupts</b> .....	2-1
Interrupt 02H—Nonmaskable Interrupt (NMI) .....	2-IN02-1
Interrupt 05H—Print Screen .....	2-IN05-1
Interrupt 08H—System Timer .....	2-IN08-1
Interrupt 09H—Keyboard .....	2-IN09-1
Interrupt 10H—Video .....	2-IN10-1
Interrupt 11H—Equipment Determination .....	2-IN11-1
Interrupt 12H—Memory Size Determination .....	2-IN12-1
Interrupt 13H—Diskette .....	2-IN13D-1
Interrupt 13H—Fixed Disk .....	2-IN13F-1
Interrupt 14H—Asynchronous Communication .....	2-IN14-1
Interrupt 15H—System Services .....	2-IN15-1
Interrupt 16H—Keyboard .....	2-IN16-1
Interrupt 17H—Printer .....	2-IN17-1
Interrupt 19H—Bootstrap Loader .....	2-IN19-1
Interrupt 1AH—System-Timer and Real-Time Clock Services .....	2-IN1A-1
Interrupt 4BH—Advanced Services .....	2-IN4B-1
Interrupt 70H—Real-Time Clock Interrupt .....	2-IN70-1
 <b>Section 3. BIOS Data Areas</b> .....	3-1
Extended BIOS Data Areas .....	3-21
 <b>Section 4. ROM Tables</b> .....	4-1
Fixed Disk Drive Parameter Table .....	4-1
Diskette Drive Parameter Table .....	4-8
 <b>Section 5. Additional Information</b> .....	5-1
Interrupt Sharing .....	5-3
Considerations .....	5-3
Interrupt Request (IRQ n) Reset .....	5-4
Interrupt-Sharing Software Requirements .....	5-4
Interrupt-Sharing Chaining Structure and Signature .....	5-6
ROM Considerations .....	5-7
Implementation Information .....	5-7
Adapter ROM .....	5-11


Video-Function Compatibility .....	5-13
Video-Presence Test .....	5-13
Video-Mode Switching .....	5-14
Multitasking Provisions .....	5-15
Application Guidelines .....	5-17
Math-Coprocessor Testing .....	5-17
Hardware Interrupts .....	5-17
Programming Considerations .....	5-18
BIOS and Operating-System Function Calls .....	5-19
 <b>Section 6. System Identification .....</b>	 <b>6-1</b>
 <b>Section 7. Scan Code/Character Code Combinations .....</b>	 <b>7-1</b>

---

# Section 1. Introduction to BIOS

Interrupts	1-3
Parameter Passing	1-5
Data Areas and ROM Tables	1-5
BIOS-Level Determination	1-6
System Groups	1-6

**Notes:**




The Basic Input/Output System (BIOS) for IBM Personal System/2 and Personal Computer products is a software interface or "layer" that isolates operating systems and application programs from specific hardware devices. BIOS routines allow assembly-language programmers to perform block- and character-level operations without concern for device addresses or hardware operating characteristics. BIOS also provides system services, such as time-of-day and memory-size determination.

Operating systems and application programs should make functional requests to BIOS instead of directly manipulating I/O ports and control words of the hardware. Hardware design and timing changes then become less critical, and software compatibility across systems and features is enhanced.

---

## **Interrupts**

BIOS is accessed through software interrupts; each BIOS entry point is available through its own interrupt. Within each interrupt, the AH register indicates the specific function that is being executed.



The table on the following page lists each interrupt, its function, and whether a detailed description of the interrupt is included in this technical reference manual.

<b>Interrupt Number</b>	<b>Interrupt Function</b>	<b>In This Technical Reference?</b>
00H	Divide by 0	
01H	Single Step	
02H	Nonmaskable Interrupt (NMI)	Yes
03H	Breakpoint	
04H	Overflow	
05H	Print Screen	Yes
06H to 07H	Reserved	
08H	System Timer	Yes
09H	Keyboard	Yes
0AH to 0DH	Reserved	
0EH	Diskette	
0FH	Reserved	
10H	Video	Yes
11H	Equipment Determination	Yes
12H	Memory Size Determination	Yes
13H	Fixed Disk/Diskette	Yes
14H	Asynchronous Communication	Yes
15H	System Services	Yes
16H	Keyboard	Yes
17H	Printer	Yes
18H	Resident BASIC	
19H	Bootstrap Loader	Yes
1AH	System-Timer and Real-Time Clock Services	Yes
1BH	Keyboard Break	
1CH	User Timer Tick	
1DH	Video Parameters	
1EH	Diskette Parameters	
1FH	Video Graphics Characters	
20H to 3FH	Diskette BIOS Revector	
41H	Fixed Disk Parameters	
42H to 45H	Reserved	
46H	Fixed Disk Parameters	
47H to 49H	Reserved	
4AH	User Alarm	
4BH	Advanced Services	Yes
4CH to 5FH	Reserved	
60H to 67H	Reserved for User-Program Interrupts	
68H to 6FH	Reserved	
70H	Real-Time Clock	Yes
71H to 74H	Reserved	
75H	Redirect to NMI	
76H to 7FH	Reserved	
80H to 85H	Reserved	
86H to F0H	Used by BASIC Interpreter When Running BASIC	
F1H to FFH	Reserved for User-Program Interrupts	

**Figure 1-1. Interrupts**



---

## Parameter Passing

All parameters that are passed to and from the BIOS routines go through the microprocessor registers. Each BIOS interrupt routine indicates which registers are used on the call and on the return. In general, if a BIOS routine has several possible functions, (AH) is used to select the desired function. For example, to set the time, the following code is required:

```
MOV AH,1           ;Function is to set time of day.
MOV CX,HIGH_COUNT  ;Establish the current time.
MOV DX,LOW_COUNT   ;
INT 1AH            ;Set the time.
```

To read the time, the following code is required:

```
MOV AH,0           ;Function is to read time of day.
INT 1AH            ;Read the timer.
```

| The BIOS interrupt handlers save all registers (including the  
| extended portion) except (AX), the flags, and registers that return  
| values to the caller. In some cases, other registers are modified.  
See the "Interrupts" section for additional information.

All parameters are 1-based (that is, the count starts with 1, not with 0), unless they are noted as 0-based.

---

## Data Areas and ROM Tables

Data areas are the memory locations that are allocated specifically to system BIOS and adapter BIOS to be used as work areas. Read-only memory (ROM) tables are used by BIOS to define the characteristics of hardware devices that are supported by a particular system BIOS or adapter BIOS.

See the "Data Areas" section and the "ROM Tables" section for additional information.

---

## BIOS-Level Determination

BIOS is contained in ROM modules on the system boards of Personal System/2 and Personal Computer products. It is also contained in ROM modules on some optional features (usually adapters) to provide device-level control of the features.

BIOS has been amended several times since it was first developed. All BIOS versions are dated. In this technical reference, BIOS-version dates are used when necessary to indicate interface differences in similar systems.

To determine the BIOS-version date of a system, run the following BASIC program. The date that is displayed is the version date of BIOS for that system.

```
10 DEF SEG=&HF000
20 FOR X=&HFFF5 TO &HFFFC
30 PRINT CHR$(PEEK(X));
40 NEXT
RUN
```

See "System Identification" in the "Additional Information" section for a list of IBM products and their BIOS-version dates. To access this information, see Interrupt 15H, Return System Configuration Parameters function ((AH)=C0H).

---

## System Groups

In this technical reference, IBM systems are categorized into groups of systems that have similar BIOS interfaces. When these groups are referred to, any exceptions are noted. These groups include:

- Personal System/2 products, all models
- Personal Computer XT\* products, including the Portable Personal Computer.
- Personal Computer AT\* products, all models.

---

\* Personal Computer XT and Personal Computer AT are trademarks of the International Business Machines Corporation.

---

# Section 2. Interrupts

## Notes:



## Interrupt 02H—Nonmaskable Interrupt (NMI)

The nonmaskable interrupt (NMI) handler is used to process severe errors. In most cases, the fault is in the hardware, but it is also possible for a software error to force an NMI to occur.

When an NMI occurs, the interrupt handler displays the error code that is associated with the error and halts the processor. The error codes are as follows.

Value	Description
Parity check 1	System-board memory failure
Parity check 2	I/O-channel parity error
110	System-board memory failure
111	Channel check activated (assumes channel memory)
112	Watchdog time-out
113	Direct memory access (DMA) bus time-out
210	System-board memory failure
211	Channel check activated (assumes channel memory)
00021000	System-board memory failure
000210xy	System-board memory failure in memory module x, which is of type y
00021100	Channel check activated (assumes channel memory)
000211x0	Channel check activated in slot x
0xxx1200	Watchdog time-out xxx = 129 - Processor card = 001 - System board
0xxx1300	Direct memory access (DMA) bus time-out xxx = 129 - Processor card = 001 - System board
01290400	Level 2 cache memory failure

**Note:** Not all codes apply to every system.

**Figure 2-1. Nonmaskable Interrupt Error Codes**

When a system-board memory failure (parity error) occurs, the NMI handler attempts to find the storage location that contains the bad parity. If the location is found, the segment address is displayed. If no parity error is found, "?????" appears in place of the address, indicating an intermittent read problem. When performing this test, the NMI handler does not check memory above 640KB, so it is possible for a recurring problem to exist that the NMI handler cannot find. Some of the Personal System/2 Model 90 and 95 systems display the socket number of the memory module that contains the error, and they display the type of the memory module.

| When a channel check occurs, the NMI handler assumes that the error was caused by an error in a channel adapter card. The Personal System/2 Model 90 and 95 systems attempt to determine which card caused the error and display the slot number along with the error code.

When the watchdog timer is enabled and a timer interrupt (IRQ 0) is missing, the system generates an NMI.

When a DMA-driven device uses the bus longer than the allowable time period (7.8 microseconds) after it is preempted by another device, the central arbitration control point generates an NMI.

When an NMI occurs, the central arbitration control point is implicitly disabled. The NMI handler explicitly reenables the central arbitration control point by writing hex 0 to port hex 90.

A system that has a math coprocessor must direct math coprocessor errors to this interrupt. An 8087 math coprocessor error on 8088- or 8086-based systems drives the NMI of the 8088 or 8086, respectively. An 80287 or 80387 math coprocessor error on 80286-, 80386-, and 80486-based systems drives the IRQ 13 line. The IRQ 13 interrupt handler issues a software Interrupt 02H to be compatible with software that expects the NMI to occur. For all systems, the math coprocessor application that points the NMI vector to itself must be sensitive to NMI errors. If an NMI occurs because of an NMI error, control should be transferred to the system NMI handler.

#### **Notes:**

1. For PCjr\*, the NMI is attached to the keyboard interrupt.
2. For PC Convertible, the NMI is attached to the keyboard, the diskette, the real-time clock, and the system-suspend interrupts. The NMI is activated by an I/O channel check.

---

\* PCjr is a trademark of the International Business Machines Corporation.

## Interrupt 05H—Print Screen

This interrupt handler prints the screen to printer port 0. When Interrupt 05H is invoked, the cursor position is saved and is restored when the interrupt is completed. Interrupt 05H runs with interrupts enabled; however, additional print-screen requests are ignored when a print-screen operation is already in progress. An initial status error from the printer ends the print-screen request. The byte at address hex 50:00 contains the status of the print-screen operation.

The following figure lists the status indications for the print-screen status byte (address hex 50:00).

Value	Description
00H	Print screen not called or, on return, operation successfully completed
01H	Print screen in progress, ignore request
FFH	Error encountered during printing

*Figure 2-2. Print Screen Status*

For PC Convertible, an initial status error ends the print-screen request and also sounds a “beep.” The Ctrl-Break sequence ends the print-screen operation.

**Notes:**





---

## Interrupt 08H—System Timer

This interrupt handler services the timer interrupt from channel 0 of the system timer. This interrupt occurs approximately 18.2 times per second.

The interrupt handler:

- Maintains a count of interrupts at data area address hex 40:6C (timer counter) since power-on that can be used to establish the time of day. After 24 hours of operation, address hex 40:70 (timer overflow) is increased (made nonzero). When the time counter crosses a day boundary, address hex 40:CE (day counter) is increased.
- Decreases the value at address hex 40:40 (motor-off counter of the diskette drive) and, when the count reaches 0, turns the diskette-drive motor off and resets the motor-running flags in address hex 40:3F (motor status).
- Issues a software Interrupt 1CH.

For PC Convertible, this interrupt handler calls a user routine through software Interrupt 4AH when an alarm interrupt occurs.

**Notes:**



---

## Interrupt 09H—Keyboard

This interrupt handler is issued on the make or break of every keystroke.

For ASCII keys, when a make code is read from port hex 60, the character code and scan code are put into the 32-byte keyboard buffer that begins at address hex 40:1E in the BIOS data area, at the address pointed to by the keyboard-buffer tail pointer (hex 40:1C). The keyboard-buffer tail pointer is then increased by 2, unless it extends past the end of the buffer. In this case, it is reinitialized to the start of the buffer.

For every Ctrl, Alt, or Shift key make or break, BIOS data area addresses hex 40:17 and hex 40:18 (keyboard control) and hex 40:96 (keyboard flags) are updated.

The Ctrl-Alt-Delete sequence causes the interrupt handler to set the reset flag (hex 40:72) to hex 1234 (bypass memory test) then jump to the power-on self-test (POST). POST checks the reset flag (hex 40:72); if the reset flag is set to hex 1234, POST does not retest the memory. For the PC Convertible, a processor reset is performed instead of a jump to POST, causing POST to be executed.

Pressing the Pause key causes the interrupt handler to loop until a valid ASCII keystroke occurs. The PC Convertible issues Interrupt 15H, Wait for External Event function ((AH)=41H) to wait for a valid ASCII keystroke.

Pressing the Print Screen key causes Interrupt 05H (Print Screen) to be issued.

The Ctrl-Break sequence causes Interrupt 1BH (Keyboard Break) to be issued.

For PC/XT\* BIOS dated 1/10/86 and later, AT\*, PC/XT Model 286, PC Convertible, and Personal System/2 products, pressing the SysRq key causes the interrupt handler to issue Interrupt 15H, System Request Key Pressed function ((AH)=85H) to inform the system of a SysRq key make or break operation. Also, the keyboard interrupt issues

---

\* PC/XT and AT are trademarks of the International Business Machines Corporation.

Interrupt 15H, Interrupt Complete function ((AH)=91H) with (AL)=02H (Type=Keyboard) to indicate that a keystroke is available.

For AT BIOS dated 6/10/85 and later, PC/XT Model 286, PC Convertible, and Personal System/2 products, Interrupt 15H, Keyboard Intercept function ((AH)=4FH) is issued after the scan code is read from port hex 60. This allows the system to replace or absorb the scan code. An End of Interrupt (EOI) instruction is processed by the Interrupt 09H handler after control is returned from the keyboard intercept handler.

| For Personal System/2 Model 56, Model 57, Model 90, and Model 95,  
| the following procedure causes the system to be booted from the  
| system partition on the fixed disk:

- | 1. Perform a Ctrl-Alt-Delete sequence.
- | 2. Wait for the cursor to appear in the upper-right corner of the display.
- | 3. Perform a Ctrl-Alt-Insert sequence.

## Interrupt 10H—Video

The following is a summary of the video functions of Interrupt 10H.

(AH) = 00H	— Set Mode
(AH) = 01H	— Set Cursor Type
(AH) = 02H	— Set Cursor Position
(AH) = 03H	— Read Cursor Position
(AH) = 04H	— Read Light-Pen Position
(AH) = 05H	— Select Active Display Page
(AH) = 06H	— Scroll Active Page Up
(AH) = 07H	— Scroll Active Page Down
(AH) = 08H	— Read Attribute/Character at Current Cursor Position
(AH) = 09H	— Write Attribute/Character at Current Cursor Position
(AH) = 0AH	— Write Character at Current Cursor Position
(AH) = 0BH	— Set Color Palette
(AH) = 0CH	— Write Dot
(AH) = 0DH	— Read Dot
(AH) = 0EH	— Write Teletype to Active Page
(AH) = 0FH	— Read Current Video State
(AH) = 10H	— Set Palette Registers
(AH) = 11H	— Character Generator
(AH) = 12H	— Alternative Selection
(AH) = 13H	— Write String
(AH) = 14H	— Load LCD Character Font/Set LCD High-Intensity Substitute
(AH) = 15H	— Return Physical-Display Parameters for Active Display
(AH) = 16H to 19H	— Reserved
(AH) = 1AH	— Read/Write Display-Combination Code
(AH) = 1BH	— Return Functionality/State Information
(AH) = 1CH	— Save/Restore Video State
(AH) = 1DH to FFH	— Reserved

**Figure 2-3.** INT 10H Video Functions

**Note:** All reserved input fields must be set to 0.

## **(AH) = 00H—Set Mode**

(AL) - Requested video mode

The following table lists the supported video modes.

<b>Mode (Hex)</b>	<b>Type</b>	<b>Maximum Colors</b>	<b>Alphanumeric Format</b>	<b>Buffer Start</b>
0, 1	A/N	16	40x25	B8000H
2, 3	A/N	16	80x25	B8000H
4, 5	APA	4	40x25	B8000H
6	APA	2	80x25	B8000H
7	A/N	Monochrome	80x25	B0000H
8	APA	16	20x25	B0000H
9	APA	16	40x25	B0000H
A	APA	4	80x25	B0000H
B, C	Reserved			
D	APA	16	40x25	A0000H
E	APA	16	80x25	A0000H
F	APA	Monochrome	80x25	A0000H
10	APA	16	80x25	A0000H
11	APA	2	80x30	A0000H
12	APA	16	80x30	A0000H
13	APA	256	40x25	A0000H
14	A/N	16	132x25	B8000H

APA = All points addressable (graphics)  
A/N = Alphanumeric (text)

**Figure 2-4. Video Modes**

The following table lists the characteristics of each video subsystem for each of the video modes.

Mode (Hex)	Display Size	Box Size	Supporting IBM Products	Maximum Pages
0, 1	320x200	8x8	PCjr, Color/Graphics Monitor Adapter (CGA), Enhanced Graphics Adapter (EGA), PC Convertible, and Personal System/2 products except Model 25 and Model 30	8
	320x350	8x14	EGA and Personal System/2 products except Model 25 and Model 30	8
	320x400	8x16	Personal System/2 Model 25 and Model 30	8
	360x400	9x16	Personal System/2 products except Model 25 and Model 30	8
2, 3	640x200	8x8	PCjr, CGA, and PC Convertible	4
	640x200	8x8	EGA and Personal System/2 products except Model 25 and Model 30	8
	640x350	8x14	EGA and Personal System/2 products except Model 25 and Model 30	8
	640x400	8x16	Personal System/2 Model 25 and Model 30	8
	720x400	9x16	Personal System/2 products except Model 25 and Model 30	8
4, 5	320x200	8x8	PCjr, CGA, EGA, and Personal System/2 products	1
6	640x200	8x8	PCjr, CGA, EGA, and Personal System/2 products	1
7	720x350	9x14	Monochrome Display and Printer Adapter (MDPA) and PC Convertible	1
	720x350	9x14	EGA and Personal System/2 products except Model 25 and Model 30	8
	720x400	9x16	Personal System/2 products except Model 25 and Model 30	8
	640x200	8x8	PC Convertible	4
8	160x200	8x8	PCjr	1
9	320x200	8x8	PCjr	1
A	640x200	8x8	PCjr	1
B, C	Reserved			

Figure 2-5 (Part 1 of 2). Video Mode Characteristics

Mode (Hex)	Display Size	Box Size	Supporting IBM Products	Maximum Pages
D	320x200	8x8	EGA and Personal System/2 products except Model 25 and Model 30	8
E	640x200	8x8	EGA and Personal System/2 products except Model 25 and Model 30	4
F, 10	640x350	8x14	EGA and Personal System/2 products except Model 25 and Model 30	2
11	640x480	8x16	Personal System/2 products	1
12	640x480	8x16	Personal System/2 products except Model 25 and Model 30	1
13	320x200	8x8	Personal System/2 products	1
14	1056x200	8x8	Personal System/2 Model 90 and Model 95 with IML update, Model 35 LS, Model 35 SX, Model 40 SX, and Model 57 SX	4
	1056x350	8x14	Personal System/2 Model 90 and Model 95 with IML update, Model 35 LS, Model 35 SX, Model 40 SX, and Model 57 SX	4
	1056x400	8x16	Personal System/2 Model 90 and Model 95 with IML update, Model 35 LS, Model 35 SX, Model 40 SX, and Model 57 SX	4

**Figure 2-5 (Part 2 of 2). Video Mode Characteristics**

**Notes:**

**1. For PCjr and IBM Color/Graphics Monitor Adapter (CGA):**

- The cursor is not displayed in graphics (APA) modes.
- Modes 0, 2, and 5 are identical to modes 1, 3, and 4, except that color burst is not enabled. Color-burst-on enables color information on composite displays; color-burst-off disables it. RGB displays are not affected by the state of color burst.
- For PCjr during mode setting, if bit 7 of (AL) is set to 1, the video buffer is not cleared.

**2. For IBM Enhanced Graphics Adapter (EGA):**

- The cursor is not displayed in graphics (APA) modes.
- Modes 0, 2, and 5 are identical to modes 1, 3, and 4, except that color burst is not enabled. Color-burst-on enables color information on composite displays; color-burst-off disables it. RGB displays are not affected by the state of color burst.
- The power-on default mode is determined by switch settings on the adapter.



- During mode setting, if bit 7 of (AL) is set to 1, the video buffer is not cleared.

See BIOS data area address hex 40:A8 in "BIOS Data Areas" for save-pointer dynamic overrides.

### 3. For PC Convertible:

- The cursor is not displayed in graphics (APA) modes.
- Modes 0, 2, and 5 are identical to modes 1, 3, and 4, except that color burst is not enabled. Color-burst-on enables color information on composite displays; color-burst-off disables it. RGB displays are not affected by the state of color burst.
- The power-on default mode for the color/graphics mode is 2.
- The power-on default mode for the monochrome mode is 7.
- During mode setting, if bit 7 of (AL) is set to 1, the video buffer is not cleared.
- Mode 7 (640x200) is used for a liquid crystal display (LCD) as monochrome.
- Mode 7 (720x350) is used for a monochrome display.

### 4. For Personal System/2 Model 25 and Model 30:

- The cursor is not displayed in graphics (APA) modes.
- Modes 0, 2, and 5 are identical to modes 1, 3, and 4.
- The power-on default mode is 3.
- During mode setting, if bit 7 of (AL) is set to 1, the video buffer is not cleared.
- In all modes except mode hex 13, the first 16 color registers are initialized, and the values in the remaining 240 color registers are undefined.
- In mode hex 13, 248 color registers are loaded.

### 5. For Personal System/2 products except Model 25 and Model 30:

- The cursor is not displayed in graphics (APA) modes.
- Modes 0, 2, and 5 are identical to modes 1, 3, and 4.
- When a color display is attached, the power-on default mode is 3. When a monochrome display is attached, the power-on default mode is 7.
- During mode setting, if bit 7 of (AL) is set to 1, the video buffer is not cleared.
- In all modes except mode hex 13, the first 64 color registers are initialized, and the values in the remaining 192 color registers are undefined.
- Refer to (AH) = 12H, (BL) = 30H on page 2-IN10-22 to select scan lines (200, 350, or 400) for alphanumeric modes.

See BIOS data area address hex 40:A8 in "BIOS Data Areas" for save-pointer dynamic overrides.

## **(AH) = 01H—Set Cursor Type**

- (CH) - Top line for cursor (bits 4 to 0)  
(Hardware causes blinking cursor;  
setting bit 6 or 5 causes erratic  
blinking or no cursor.)
- (CL) - Bottom line for cursor (bits 4 to 0)

### **Notes:**

1. BIOS maintains only one cursor type for all video pages.
2. For Personal System/2 Model 25 and Model 30, before the hardware video ports are written to, (CH) is multiplied by 2, and (CL) is multiplied by 2 and increased by 1.

## **(AH) = 02H—Set Cursor Position**

- (BH) - Page number (0 based); see Figure 2-5 on page 2-IN10-3 for maximum pages
- (DH,DL) - Row, column (0,0 is upper left)

## **(AH) = 03H—Read Cursor Position**

- (BH) - Page number (0 based); see Figure 2-5 on page 2-IN10-3 for maximum pages

### **On Return:**

- (CH,CL) - Cursor type currently set
- (DH,DL) - Row, column of current cursor for requested page

## **(AH) = 04H—Read Light-Pen Position**

For PC Convertible and Personal System/2 products, this function is not supported:

### **On Return:**

- (AH) = 00H - Light pen is not supported
- (BX), (CX), and (DX) are altered on return.

For all others:

### **On Return:**

- (AH) = 00H - Light-pen switch not activated
- (BX), (CX), and (DX) are altered on return.

(AH) = 01H - Valid light-pen value in registers  
(BX) - Pel column (from 0 to 319 639)  
(CH) - Raster line (from 0 to 199)  
(CX) - Raster line (from 0 to *nnn*), new graphics modes  
(DH,DL) - Row, column of character

### **(AH) = 05H—Select Active Display Page**

For PCjr:

(AL) = 80H - Read microprocessor- and display-page registers

(AL) = 81H - Set microprocessor-page register  
(BL) - Microprocessor-page register

(AL) = 82H - Set display-page register  
(BH) - Display-page register

(AL) = 83H - Set microprocessor- and display-page registers  
(BH) - Display-page register  
(BL) - Microprocessor-page register

On Return:

(BH) - Display-page register  
(BL) - Microprocessor-page register

For all others:

(AL) - New page number (0 based); see Figure 2-5 on page 2-IN10-3 for maximum pages

### **(AH) = 06H—Scroll Active Page Up**

(AL) - Number of lines to be blanked at bottom of window  
= 00H - Blank entire window

(BH) - Attribute to be used on blank line  
(CH,CL) - Row, column of upper-left corner of scroll  
(DH,DL) - Row, column of lower-right corner of scroll

### **(AH) = 07H—Scroll Active Page Down**

(AL) - Number of input lines to be blanked at top of window  
= 00H - Blank entire window

(BH) - Attribute to be used on blank line  
(CH,CL) - Row, column of upper-left corner of scroll  
(DH,DL) - Row, column of lower-right corner of scroll

## **(AH) = 08H—Read Attribute/Character at Current Cursor Position**

(BH) - Page number (0 based); see Figure 2-5 on page 2-IN10-3 for maximum pages

On Return:

(AH) - Attribute of character that was read (alphanumeric modes only)

(AL) - Character that was read

## **(AH) = 09H—Write Attribute/Character at Current Cursor Position**

For the read- and write-character interface in graphics modes 4, 5, and 6, the characters are formed from a character generator that is maintained in the system ROM, which contains only the first 128 characters. To read or write the second 128 characters, initialize the pointer at interrupt vector hex 1F (address hex 0007C) to point to the 1KB table that contains the code points for the second 128 characters (128 to 255).

In all other graphics modes, 256 graphics characters are supplied in the system ROM.

For the write-character interface in graphics mode, the character count that is contained in (CX) produces valid results for characters in the same row only. Continuation to succeeding rows produces invalid results.

(AL) - Character to be written

(BH) - Page number (0 based); see Figure 2-5 on page 2-IN10-3 for maximum pages

(BL) - Attribute of character (alphanumeric) or color of character (graphics)

(CX) - Count of characters to be written

### **Notes:**

1. The Write Attribute/Character function ((AH)=09H) and the Write Character function ((AH)=0AH) are similar. Use the Write Attribute/Character function ((AH)=09H) for graphics modes.
2. In graphics modes, if bit 7 of (BL) is set to 1, the color value is exclusively ORed with the current video memory (except in mode hex 13).
3. In mode hex 13, the value that is passed in (BH) is used as the background color.

## **(AH) = 0AH—Write Character at Current Cursor Position**

(AL) - Character to be written

(BH) - Page number (0 based); see Figure 2-5 on page 2-IN10-3 for maximum pages

(CX) - Count of characters to be written

Use the Write Attribute/Character function ((AH) = 09H) for graphics modes.

## **(AH) = 0BH—Set Color Palette**

(BH) - Color ID being set (from 0 to 1)

(BL) - Color value to be used with color ID

(BH) = 00H - For 320x200 graphics modes, set background color

- For alphanumeric modes, set border color

- For 640x200 graphics modes, set foreground color

(BL) = from 0 to 31

(BH) = 01H - Select palette for 320x200 graphics modes

(BL) = 0 - Green (1)/red (2)/brown (3)

= 1 - Cyan (1)/magenta (2)/white (3)

### **Notes:**

1. This interface has meaning for 320x200 graphics modes only.
2. In 40x25 or 80x25 alphanumeric modes, the value that is set for palette color 0 indicates which border color is to be used (0 to 31); values 16 to 31 indicate the high-intensity background set.
3. For EGA and Personal System/2 products, when the 640x200 graphics mode is active and the color ID is 0, the background color is set.

## **(AH) = 0CH—Write Dot**

(AL) - Color value

(CX) - Column number

(DX) - Row number

**Note:** If bit 7 of (AL) is set to 1, the color value is exclusively ORed with the current contents of the dot (except in mode hex 13).

In graphics modes that support more than one page:

(BH) - Page number (0 based); see Figure 2-5 on page 2-IN10-3 for maximum pages

## **(AH) = 0DH—Read Dot**

(CX) - Column number

(DX) - Row number

In graphics modes that support more than one page:

(BH) - Page number (0 based); see Figure 2-5 on page 2-IN10-3 for maximum pages

On Return:

(AL) - Dot that was read

## **(AH) = 0EH—Write Teletype to Active Page**

(AL) - Character to be written

(BL) - Foreground color in graphics mode

### **Notes:**

1. The mode that is currently set controls the screen width.
2. Carriage Return, Line Feed, Backspace, and Bell are treated as commands, not as printable characters.
3. For PC BIOS dated 4/24/81 and 10/19/81, (BH) must be set to the active page.

## **(AH) = 0FH—Read Current Video State**

On Return:

(AH) - Number of character columns on screen

(AL) - Mode that is currently set

(see (AH) = 00H on page 2-IN10-2 for explanation)

(BH) - Current active page number (0 based); see

Figure 2-5 on page 2-IN10-3 for maximum pages

## **(AH) = 10H—Set Palette Registers**

For PCjr, systems with EGA capability, and Personal System/2 products except Model 25 and Model 30:

(AL) = 00H - Set individual palette register

(BH) - Value to be set

(BL) - Palette register to be set

(AL) = 01H - Set overscan register

(BH) - Value to be set

(AL) = 02H - Set all palette registers and overscan  
(ES:DX) - Pointer to 17-byte table  
Byte 16 - Overscan value  
Bytes 15 to 0 - Palette values

(AL) = 03H - Toggle intensify/blinking bit  
(BL) = 00H - Enable intensify  
= 01H - Enable blinking

## **For Personal System/2 products except Model 25 and Model 30:**

(AL) = 04H to 06H - Reserved

(AL) = 07H - Read individual palette register  
(BL) - Palette register to be read (from 0 to 15)

On Return:  
(BH) - Value that was read

(AL) = 08H - Read overscan register

On Return:  
(BH) - Value that was read

(AL) = 09H - Read all palette registers and overscan  
(ES:DX) - Pointer to 17-byte buffer for return values

On Return:  
(ES:DX) - Pointer to 17-byte table destination  
Byte 16 - Overscan value  
Bytes 15 to 0 - Palette values

(AL) = 10H - Set individual color register  
(BX) - Color register to be set  
(CH) - Green value to be set  
(CL) - Blue value to be set  
(DH) - Red value to be set

(AL) = 11H - Reserved

(AL) = 12H - Set block of color registers  
(BX) - First color register to be set  
(CX) - Number of color registers to be set  
(ES:DX) - Pointer to table of color values  
(Table format: red, green, blue, red, green, blue)

(AL) = 13H - Select color page (not valid for mode hex 13)  
(BH) - Paging mode  
= 00H - Selects 4 register blocks of 64 registers  
= 01H - Selects 16 register blocks of 16 registers  
(BL) = 00H - Select paging mode

(BH) - Page number (0 based); see Figure 2-5 on page 2-IN10-3 for maximum pages

For 64-register block mode:

= 00H - Selects first block of 64 color registers

= 01H - Selects second block of 64 color registers

= 02H - Selects third block of 64 color registers

= 03H - Selects fourth block of 64 color registers

For 16-register block mode:

= 00H - Selects first block of 16 color registers

= 01H - Selects second block of 16 color registers

.

.

.

= 0FH - Selects 16th block of 16 color registers

(BL) = 01H - Select page

**Note:** The Set Mode function ((AH)=00H) defaults to the 64-register block mode, with the first block of 64 color registers active. Only these 64 color registers are initialized during mode setting. When using page selection, initialize the other blocks of the color registers.

(AL) = 14H - Reserved

(AL) = 15H - Read individual color register

(BX) - Color register to be read

On Return:

(CH) - Green value that was read

(CL) - Blue value that was read

(DH) - Red value that was read

(AL) = 16H - Reserved

(AL) = 17H - Read block of color registers

(BX) - First color register to be read

(CX) - Number of color registers to be read

(ES:DX) - Pointer to destination table for values

(Table format: red, green, blue, red, green, blue)

On Return:

(ES:DX) - Pointer to table of values

(AL) = 18H to 19H - Reserved

(AL) = 1AH - Read color-page state

On Return:

(BH) - Current page

(BL) - Current paging mode



**Note:** See (AL) = 13H on page 2-IN10-11 for paging modes and page information.

(AL) = 1BH - Sum color values to gray shades  
(BX) - First color register to be summed  
(CX) - Number of color registers to be summed

**Note:** This call reads the red, green, and blue values that are found in the color registers, performs a weighted sum (30% red + 59% green + 11% blue), then writes the result into each red, green, and blue component of the color register (the original data is not retained).

**For Personal System/2 Model 25 and Model 30:**

(AL) = 00H  
(BX) = 0712H - Color registers to be set, resulting  
in 8 consistent colors

(AL) = 01H to 02H - Reserved

(AL) = 03H - Toggle intensify/blinking bit  
(BL) = 00H - Enable intensify  
= 01H - Enable blinking

(AL) = 04H to 07H - Reserved

(AL) = 10H - Set individual color register  
(BX) - Color register to be set  
(CH) - Green value to be set  
(CL) - Blue value to be set  
(DH) - Red value to be set

(AL) = 11H - Reserved

(AL) = 12H - Set block of color registers  
(BX) - First color register to be set  
(CX) - Number of color registers to be set  
(ES:DX) - Pointer to table of color values  
(Table format: red, green, blue, red, green, blue)

(AL) = 13H to 14H - Reserved

(AL) = 15H - Read individual color register  
(BX) - Color register to be read

**On Return:**  
(CH) - Green value that was read  
(CL) - Blue value that was read  
(DH) - Red value that was read

(AL) = 16H - Reserved

(AL) = 17H - Read a block of color registers  
(BX) - First color register to be read  
(CX) - Number of color registers to be read  
(ES:DX) - Pointer to destination table for values  
(Table format: red, green, blue, red, green, blue)

On Return:  
(ES:DX) - Pointer to table of values

(AL) = 18H to 1AH - Reserved

(AL) = 1BH - Sum color values to gray shades  
(BX) - First color register to be summed  
(CX) - Number of color registers to be summed

**Note:** This call reads the red, green, and blue values that are found in the color registers, performs a weighted sum (30% red + 59% green + 11% blue), then writes the result into each red, green, and blue component of the color register (the original data is not retained).

For all others, no action is performed.

## **(AH) = 11H—Character Generator**

For systems with EGA capability, this call initiates a mode setting that completely resets the video environment but maintains the regenerative buffer.

(AL) = 00H - User alphanumeric load  
(BH) - Number of bytes per character  
(BL) - Block to be loaded  
(CX) - Count to be stored  
(DX) - Character offset into table  
(ES:BP) - Pointer to user table

(AL) = 01H - ROM monochrome set  
(BL) - Block to be loaded

(AL) = 02H - ROM 8x8 double dot  
(BL) - Block to be loaded

(AL) = 03H - Set block specifier (valid in alphanumeric modes)  
(BL) - Character-generator block selection  
If bit 3 of the character-attribute byte is set to 0,  
bits 1 and 0 of (BL) select a block from 0 to 3.  
If bit 3 of the character-attribute byte is set to 1,  
bits 3 and 2 of (BL) select a block from 0 to 3.

For example:

- To set a 256-character set using block 3, set (BL) to hex 0F; this selects a single block. Bit 3 of the character-attribute byte turns the foreground intensity on or off.
- To specify a 512-character set using blocks 0 and 3, set (BL) to hex 0C; this selects block 0 when bit 3 of the character-attribute byte is set to 0, and it selects block 3 when bit 3 of the character-attribute byte is set to 1.

If bits 1 and 0 and bits 3 and 2 are the same, only one block is selected, and bit 3 of the character-attribute byte turns the foreground intensity on or off.

When the 512-character set is active, a function call with (AX) set to hex 1000 and (BX) set to hex 0712 is recommended to set the color registers with eight consistent colors.

Register values (AL) = 10H, 11H, and 12H are similar to register values (AL) = 00H, 01H, and 02H, respectively, with the following exceptions:

- Page 0 must be active.
- Points (bytes per character) are recalculated.
- Rows are calculated as follows:

$$\text{INT} [(200 \text{ or } 350) / \text{points}] - 1$$

- The length of the regenerative buffer is calculated as follows:

$$(\text{Number of rows on screen}) \times (\text{Number of columns on screen}) \times 2$$

- The display controller registers are reprogrammed as follows:

Maximum scan line	Points - 1
Cursor start	Points - 2
Cursor end	Points - 1
Vertical display end	$[(\text{Number of rows on screen}) \times \text{Points}] - 1$
Underline location (mode hex 7 only)	Points - 1

- Register calculations must be close to the original table values, or the results might be unpredictable.

(AL) = 10H - User alphanumeric load  
(BH) - Number of bytes per character  
(BL) - Block to be loaded  
(CX) - Count to be stored  
(DX) - Character offset into table  
(ES:BP) - Pointer to user table

(AL) = 11H - ROM monochrome set  
(BL) - Block to be loaded

(AL) = 12H - ROM 8x8 double dot  
(BL) - Block to be loaded

(AL) = 20H - Set user-graphics-characters pointer at Interrupt 1FH  
(ES:BP) - Pointer to user table

(AL) = 21H - Set user-graphics-characters pointer at Interrupt 43H  
(BL) - Row specifier  
= 00H - User  
(CX) - Points (bytes per character)  
(DL) - Rows  
= 01H - 14 (0EH)  
= 02H - 25 (19H)  
= 03H - 43 (2BH)  
(ES:BP) - Pointer to user table

(AL) = 22H - ROM 8x14 set  
(BL) - Row specifier

(AL) = 23H - ROM 8x8 double dot  
(BL) - Row specifier

**Note:** (AL) = 10H, 11H, 12H, 20H, 21H, 22H, or 23H should be called only immediately after a mode setting is issued, or the results might be unpredictable.

(AL) = 30H - Information  
(BH) - Font pointer  
= 00H - Return current Interrupt 1FH pointer  
= 01H - Return current Interrupt 44H pointer  
= 02H - Return ROM 8x14 font pointer  
= 03H - Return ROM double-dot pointer  
= 04H - Return ROM double-dot pointer (top)  
= 05H - Return ROM alphanumeric alternative 9x14

On Return:

(CX) - Points  
(DL) - Rows  
(ES:BP) - Pointer to table

## For Personal System/2 products except Model 25 and Model 30:

(AL) = 00H - User alphanumeric load  
(BH) - Number of bytes per character  
(BL) - Block to be loaded  
(CX) - Count to be stored  
(DX) - Character offset into table  
(ES:BP) - Pointer to user table

(AL) = 01H - ROM 8x14 font  
(BL) - Block to be loaded

(AL) = 02H - ROM 8x8 double-dot font  
(BL) - Block to be loaded

(AL) = 03H - Set block specifier (valid in alphanumeric modes)  
(BL) - Character-generator block selection  
If bit 3 of the character-attribute byte is set to 0,  
bits 4, 1, and 0 of (BL) select a block from 0 to 7.  
If bit 3 of the character-attribute byte is set to 1,  
bits 5, 3, and 2 of (BL) select a block from 0 to 7.

### For example:

- To set a 256-character set using block 6, set (BL) to hex 03A; this selects a single block. Bit 3 of the character-attribute byte turns the foreground intensity on or off.
- To specify a 512-character set using blocks 0 and 6, set (BL) to hex 028; this selects block 0 when bit 3 of the character-attribute byte is set to 0, and it selects block 6 when bit 3 of the character-attribute byte is set to 1.

If bits 4, 1, and 0 and bits 5, 3, and 2 are the same, only one block is selected, and bit 3 of the character-attribute byte turns foreground intensity on or off.

When the 512-character set is active, a function call with (AX) set to hex 1000 and (BX) set to hex 0712 is recommended to set the color registers with eight consistent colors.

(AL) = 04H - ROM 8x16 font  
(BL) - Block to be loaded

Register values (AL) = 10H, 11H, 12H, and 14H are similar to register values (AL) = 00H, 01H, 02H, and 04H, respectively, with the following exceptions:

- Page 0 must be active.
- Points (bytes per character) are recalculated.

- Rows are calculated as follows:

$$\text{INT} [(200, 350, \text{ or } 450) / \text{points}] - 1$$

- The length of the regenerative buffer is calculated as follows:

$$(\text{Number of rows on screen}) \times (\text{Number of columns on screen}) \times 2$$

- The display controller registers are reprogrammed as follows:

Maximum scan line	Points - 1
Cursor start	Points - 2
Cursor end	Points - 1
Vertical displacement end	For 350- and 400-scan-line modes: $[(\text{Number of rows on screen}) \times \text{Points}] - 1$ For 200-scan-line modes: $\{[(\text{Number of rows on screen}) \times \text{Points}] \times 2\} - 1$
Underline location (mode hex 7 only)	Points - 1

- Register calculations must be close to the original table values, or the results might be unpredictable.

(AL) = 10H - User alphanumeric load  
(BH) - Number of bytes per character  
(BL) - Block to be loaded  
(CX) - Count to be stored  
(DX) - Character offset into table  
(ES:BP) - Pointer to user table

(AL) = 11H - ROM 8x14 font  
(BL) - Block to be loaded

(AL) = 12H - ROM 8x8 double-dot font  
(BL) - Block to be loaded

(AL) = 14H - ROM 8x16 font  
(BL) - Block to be loaded

(AL) = 20H - Set user-graphics-characters pointer at Interrupt 1FH  
(ES:BP) - Pointer to user table

(AL) = 21H - Set user-graphics-characters pointer at Interrupt 43H  
(BL) - Row specifier  
= 00H - User  
(CX) - Points (bytes per character)  
(DL) - Rows  
= 01H - 14 (0EH)  
= 02H - 25 (19H)  
= 03H - 43 (2BH)  
(ES:BP) - Pointer to user table

(AL) = 22H - ROM 8x14 font  
(BL) - Row specifier

(AL) = 23H - ROM 8x8 double-dot font  
(BL) - Row specifier

(AL) = 24H - ROM 8x16 font  
(BL) - Row specifier

**Note:** (AL) = 10H, 11H, 12H, 14H, 20H, 21H, 22H, 23H, or 24H should be called only immediately after a mode setting is issued, or the results might be unpredictable.

(AL) = 30H - Information  
(BH) - Font pointer  
= 00H - Return current Interrupt 1FH pointer  
= 01H - Return current Interrupt 43H pointer  
= 02H - Return ROM 8x14 font pointer  
= 03H - Return ROM 8x8 font pointer  
= 04H - Return ROM 8x8 font pointer (top)  
= 05H - Return ROM 9x14 font alternative  
= 06H - Return ROM 8x16 pointer  
= 07H - Return ROM 9x16 font alternative

On Return:

(CX) - Points  
(DL) - Rows (number of character rows on screen - 1)  
(ES:BP) - Pointer to table

For Personal System/2 Model 25 and Model 30:

(AL) = 00H - User alphanumeric load  
(BH) = 16 bytes per character for 400 scan lines  
(BL) - Block to be loaded  
(CX) - Count to be stored  
(DX) - Character offset into table  
(ES:BP) - Pointer to user table

**Note:** If (BH) is set to 14 bytes per character for 400 scan lines, characters are extended to 16 high by extending the last line of 14-high characters.

(AL) = 01H - Reserved  
(If this function is called, (AL) = 04H is executed.)

(AL) = 02H - ROM 8x8 double-dot font  
(BL) - Block to be loaded

(AL) = 03H - Set block specifier (valid in alphanumeric modes)  
(BL) - Character-generator block selection  
If bit 3 of the character-attribute byte is set to 0,  
bits 1 and 0 of (BL) select a block from 0 to 3.  
If bit 3 of the character-attribute byte is set to 1,  
bits 3 and 2 of (BL) select a block from 0 to 3.

For example:

- To set a 256-character set using block 2, set (BL) to hex 0A; this selects a single block. Bit 3 of the character-attribute byte turns the foreground intensity on or off.
- To specify a 512-character set using blocks 0 and 2, set (BL) to hex 08; this selects block 0 when bit 3 of the character-attribute byte is set to 0, and it selects block 2 when bit 3 of the character-attribute byte is set to 1.

If bits 1 and 0 and bits 3 and 2 are the same, only one block is selected, and bit 3 of the character-attribute byte turns the foreground intensity on or off.

When the 512-character set is active, a function call with (AX) set to hex 1000 and (BX) set to hex 0712 is recommended to set the color registers with eight consistent colors.

A block-specifier command must be issued after any character-load command to make the loaded block an active character set.

(AL) = 04H - ROM 8x16 font  
(BL) - Block to be loaded

The following register values are reserved. Calls to functions (AL)=10H, 11H, 12H, and 14H are executed as if they were calls to (AL)=00H, 01H, 02H, and 04H, respectively.

(AL) = 10H - Reserved  
(If this function is called, (AL)=00H is executed.)

(AL) = 11H - Reserved  
(If this function is called, (AL)=01H is executed.)

(AL) = 12H - Reserved  
(If this function is called, (AL)=02H is executed.)

(AL) = 14H - Reserved  
(If this function is called, (AL)=04H is executed.)

(AL) = 20H - Set user-graphics-characters pointer at Interrupt 1FH  
(ES:BP) - Pointer to user table



(AL) = 21H - Set user-graphics-characters pointer at Interrupt 43H  
(BL) - Row specifier  
= 00H - User  
(CX) - Points (bytes per character)  
(DL) - Rows  
= 01H - 14 (0EH)  
= 02H - 25 (19H)  
= 03H - 43 (2BH)  
(ES:BP) - Pointer to user table

(AL) = 22H - Reserved  
(If this function is called, (AL) = 24H is executed.)

(AL) = 23H - ROM 8x8 double-dot font  
(BL) - Row specifier

(AL) = 24H ROM 8x16 font  
(BL) - Row specifier

**Note:** (AL) = 20H, 21H, 22H, 23H, or 24H should be called only immediately after a mode setting is issued, or the results might be unpredictable.

(AL) = 30H - Information  
(BH) - Font pointer  
= 00H - Return current Interrupt 1FH pointer  
= 01H - Return current Interrupt 43H pointer  
= 02H - Reserved (if called, ROM 8x16 pointer is returned)  
= 03H - Return ROM 8x8 font pointer  
= 04H - Return ROM 8x8 font pointer (top)  
= 05H - Reserved  
= 06H - Return ROM 8x16 pointer  
= 07H - Reserved

On Return:  
(CX) - Points  
(DL) - Rows (number of character rows on screen - 1)  
(ES:BP) - Pointer to table

For all others, no action is performed.

## **(AH) = 12H—Alternative Selection**

For systems with EGA capability and Personal System/2 products except Model 25 and Model 30:

- (BL) = 10H - Return EGA information
  - (BH) = 00H - Color mode in effect (3Dx address range)
    - = 01H - Monochrome mode in effect (3Bx address range)
  - (BL) - Memory value
    - = 00H - 64KB
    - = 01H - 128KB
    - = 02H - 192KB
    - = 03H - 256KB
    - = 04H to FFH - Reserved
  - (CH) = Adapter bits
  - (CL) = Switch setting
- (BL) = 20H - Select alternative print-screen routine

For Personal System/2 products except Model 25 and Model 30:

- (BL) = 30H - Select scan lines for alphanumeric modes (takes effect on the next mode setting)
  - (AL) = 0 - 200 scan lines
    - = 1 - 350 scan lines
    - = 2 - 400 scan lines
- On Return:
- (AL) = 12H - Function is supported
- (BL) = 31H - Default palette loading during mode setting
- (AH) = 00H
  - (AL) = 0 - Enable default palette loading
    - = 1 - Disable default palette loading
- On Return:
- (AL) = 12H - Function is supported

**Note:** The EGA palette registers, the overscan register, and the color registers are not altered during any mode setting when they are disabled.

- (BL) = 32H - Video
- (AL) = 0 - Enable video
  - = 1 - Disable video

On Return:

- (AL) = 12H - Function is supported

**Note:** Decoding of the video I/O port addresses and the regenerative buffer addresses is enabled or disabled for the display that is currently active.

(BL) = 33H - Summing to gray shades  
(AL) = 0 - Enable summing  
      = 1 - Disable summing

On Return:  
(AL) = 12H - Function is supported

**Note:** When enabled, summing occurs when color registers are loaded through the Set Mode function ((AH)=00H) and the Set Palette Registers function ((AH)=10H).

(BL) = 34H - Cursor emulation  
(AL) = 0 - Enable cursor emulation  
      = 1 - Disable cursor emulation

On Return:  
(AL) = 12H - Function is supported

**Note:** When enabled, the requested start or end value that is passed to the Set Cursor Type function ((AH)=01H) is scaled to the current character height. The power-on default condition is for cursor emulation to be enabled.

**For Personal System/2 Model 25 and Model 30:**

(BL) = 20H - Select alternative print-screen routine

(BL) = 30H - Reserved

(BL) = 31H - Default palette loading during mode setting ((AH)=00H)  
(AL) = 0 - Enable default palette loading  
      = 1 - Disable default palette loading (the 256 color registers are not altered during any mode setting when default palette loading is disabled)

On Return:  
(AL) = 12H - Function is supported

(BL) = 32H - Video I/O address and buffers  
(AL) = 0 - Enable video I/O address and buffers  
      = 1 - Disable video I/O address and buffers

On Return:  
(AL) = 12H - Function is supported

(BL) = 33H - Summing to gray shades  
(AL) = 0 - Enable summing  
      = 1 - Disable summing

On Return:

(AL) = 12H - Function is supported

(BL) = 34H - Reserved

**Note:** When enabled, summing occurs when color registers are loaded through the Set Mode function ((AH)=00H) and the Set Palette Registers function ((AH)=10H).

**For Personal System/2 products:**

(BL) = 35H - Display switching  
(AL) = 00H - Initial adapter video off  
      (ES:DX) - Pointer to 128-byte switch-state save area  
(AL) = 01H - Initial system-board video on  
(AL) = 02H - Switch active video off  
      (ES:DX) - Pointer to switch-state buffer save area  
(AL) = 03H - Switch inactive video on  
      (ES:DX) - Pointer to previously-saved switch-state buffer

On Return:

(AL) = 12H - Function is supported

This interface enables display switching between a system-board-video driven display and an adapter-video driven display when there is overlap in the use of the BIOS data area and in hardware capabilities.

Display switching requires that a disable function ((AH)=12H, (BL)=32H) be available for the system-board-video and adapter-video functions.

If there is no conflict between the adapter video and the system-board video, both video functions are active in the system, and display switching is not required.

If there is a conflict between the adapter video and the system-board video, the system-board video function is the primary video source. The adapter-video function remains disabled until it disables the system-board video and enables itself.

When display switching is performed for the first time, the following steps are used:

1. Call Initial Adapter Video Off ((AL)=00H).
2. Call Initial System-Board Video On ((AL)=01H).

These steps are valid only the first time switching is performed. After the initiation steps, switching between the system-board video and adapter video is performed through the Switch Active Video Off request ((AL)=02H) and the Switch Inactive Video On request ((AL)=03H).

When Switch Active Video Off ((AL)=02H) is called, the currently-active video function and display are disabled. The switch-state buffer saves the video-state information, which is required when the display is reactivated through a Switch Inactive Video On ((AL)=03H) request.

When Switch Inactive Video On ((AL)=03H) is called, the currently-inactive video function and display are enabled. The switch-state buffer restores the video-state information, which was saved on a previous Switch Active Video Off ((AL)=02H) request for the display.

### **Video Adapter**

If a video adapter supports display switching, it must perform the following steps.

For Initial Adapter Video Off ((AL)=00H), the adapter:

1. Tests bit 6 at address hex 40:89 to ensure that the bit is set to 0, which indicates that this is the first call.
2. Issues Interrupt 42H with bit 7 in (AL) set to 1. Interrupt 42H activates display switching and sets bit 6 at address hex 40:89 to 1.
3. Tests bit 6 at address hex 40:89 to ensure that display switching is now active.
4. Switches the active video off ((AL)=02H).

For Switch Active Video Off ((AL)=02H), the adapter:

1. Tests bit 6 at address hex 40:89 to ensure that display switching is now active
2. Disables the system interrupts
3. Saves the BIOS data areas
4. Swaps the active and inactive video vectors to pass control to the inactive video BIOS
5. Disables the adapter
6. Enables the system interrupts.

**For Switch Inactive Video On ((AL)=03H), the adapter:**

1. Tests bit 6 at address hex 40:89 to ensure that display switching is now active
2. Disables the system interrupts
3. Restores the BIOS data areas
4. Enables the adapter
5. Enables the system interrupts.

**In addition, the following areas are affected by display switching:**

- Bits 5 and 4 of data area address hex 40:10 (installed hardware, video bits)
- Data area addresses hex 40:49 to hex 40:66 (video-control data area 1)
- Data area addresses hex 40:84 to hex 40:8A (video-control data area 2)
- Pointer to video parameter table
- Interrupt 05H vector (print screen)
- Interrupt 1DH vector (CGA video parameters)
- Interrupt 1FH vector (upper 128 8x8 characters)
- Interrupt 43H vector (graphics character table).

**For Personal System/2 products except Model 25 and Model 30:**

(BL) = 36H - Video screen off or on  
(AL) = 0 - Screen on  
      = 1 - Screen off

**On Return:**

(AL) = 12H - Function is supported

**For Personal System/2 Model 35 SX, Model 35 LS, Model 40 SX, and Model 57 SX:**

(BL) = 37H - MFI alternative attribute set  
(AL) = 0 - Disable MFI attributes  
      = 1 - Enable MFI attributes

**On Return:**

(AL) = 12H - Function is supported

**For all others, no action is performed.**

## **(AH) = 13H—Write String**

For PC/XT BIOS dated 1/10/86 and later, AT, EGA, PC Convertible, and Personal System/2 products:

(AL) = 00H

(BL) - Attribute

String: "char, char, char, . . .";  
the cursor is not moved

(AL) = 01H

(BL) - Attribute

String: "char, char, char, . . .";  
the cursor is moved

(AL) = 02H

String: "char, attr, char, attr, . . .";  
the cursor is not moved; valid for alphanumeric modes only

(AL) = 03H

String: "char, attr, char, attr, . . .";  
the cursor is moved; valid for alphanumeric modes only

(BH) - Page number (0 based); see Figure 2-5 on page 2-IN10-3 for maximum pages

(CX) - Character-only count

(DX) - Position in which to begin string, in cursor terms

(ES:BP) - Pointer to string to be written

**Note:** Carriage Return, Line Feed, Backspace, and Bell are treated as commands, not as printable characters.

For all others, no action is performed.

## **(AH) = 14H—Load LCD Character Font/Set LCD High-Intensity Substitute**

For PC Convertible:

(AL) = 00H - Load user-specified font

(BH) - Number of bytes per character (from 1 to 255), value checked

(BL) = 00H - Load main font (block 0)

= 01H - Load alternative font (block 1)

= 02H to FFH - No operation

(CX) - Number of characters to be stored (from 1 to 256), value checked

(DX) - Character offset into RAM font area

(ES:DI) - Pointer to character font within user table where loading starts

(AL) = 01H - Load system-ROM default font

(BL) = 00H - Load main font (block 0)

= 01H - Load alternative font (block 1)

= 02H to FFH - No operation

(AL) = 02H - Set mapping of LCD high-intensity attribute  
 (BL) = 00H - Ignore high-intensity attribute  
       = 01H - Map high intensity to reverse image  
       = 02H - Map high intensity to underscore  
       = 03H - Map high intensity to select alternative font  
       = 04H to FFH - No operation

(AL) = 03H to FFH - No operation

For all others, no action is performed.

## **(AH) = 15H—Return Physical-Display Parameters for Active Display**

For PC Convertible:

On Return:

(AX) - Alternative display-adapter type  
       = 0 - No alternative display adapter  
       = 5140 - LCD  
       = 5153 - CGA-type display  
       = 5151 - Monochrome-type display

(ES:DI) - Points to table defined as follows:

Word 1 - Display model number  
 Word 2 - Number of vertical pels per meter  
 Word 3 - Number of horizontal pels per meter  
 Word 4 - Total number of vertical pels  
 Word 5 - Total number of horizontal pels  
 Word 6 - Horizontal pel separation, in micrometers  
          (center to center)  
 Word 7 - Vertical pel separation, in micrometers  
          (center to center)

The following figure lists the display types that are defined for the PC Convertible:

Word	Monochrome	CGA	LCD as CGA	LCD (Monochrome)
1	5151H	5153H	5140H	5140H
2	0	0498H	08E1H	0
3	0	0A15H	0987H	0
4	0	00C8H	00C8H	0
5	0	0280H	0280H	0
6	0	0352H	01B8H	0
7	0	0184H	019AH	0

*Figure 2-6. Display Types for PC Convertible*

For all others, no action is performed.

## **(AH) = 16H to 19H—Reserved**



## **(AH) = 1AH—Read/Write Display-Combination Code**

For Personal System/2 products:

(AL) = 00H - Read display-combination code

On Return:

(AL) = 1AH - Function is supported (see display codes  
on page 2-IN10-29)

(BH) - Alternative display code

(BL) - Active display code

(AL) = 01H - Write display-combination code (see display codes  
on page 2-IN10-29)

(BH) - Alternative display code

(BL) - Active display code

On Return:

(AL) = 1AH - Function supported

Display Codes:

00H - No display

01H - Monochrome with 5151 (monochrome)

02H - CGA with 5153/5154 (color)

03H - Reserved

04H - EGA with 5153/5154 (color)

05H - EGA with 5151 (monochrome)

06H - Professional Graphics System with 5175 (color)

07H - Personal System/2 products, except Model 25 and Model 30,  
with monochrome display (for Model 25 and Model 30, see  
display code 0BH)

08H - Personal System/2 products, except Model 25 and Model 30,  
with color display (for Model 25 and Model 30, see  
display code 0BH)

09H to 0AH - Reserved

0BH - Personal System/2 products, except Model 25 and Model 30,  
video with analog monochrome

0CH - Personal System/2 Model 25 and Model 30 video with analog  
color

0DH to FEH - Reserved

For all others, no action is performed.

## **(AH) = 1BH—Return Functionality/State Information**

For Personal System/2 products:

(BX) - Implementation type

(ES:DI) - User-buffer pointer for return of information

On Return:

User buffer contains functionality/state information

(AL) = 1BH - Function is supported

For implementation type hex 00:

(BX) = 00H

(ES:DI) = Buffer size is hex 40 bytes

(DI + 00H) word - Offset to static functionality information

(DI + 02H) word - Segment to static functionality information

Video states (the following information is dynamically generated and reflects the current video state):

(DI + 04H) byte - Video mode (see (AH) = 00H on  
page 2-IN10-2 for supported modes)

(DI + 05H) word - Columns on screen (character columns on screen)

(DI + 07H) word - Length of regenerative buffer (in bytes)

(DI + 09H) word - Starting address in regenerative buffer

(DI + 0BH) word - Cursor position for eight display pages  
(row, column)

(DI + 1BH) word - Cursor-type setting (cursor start/end value)

(DI + 1DH) byte - Active display page

(DI + 1EH) word - Display-controller address (3Bx – monochrome,  
3Dx – color)

(DI + 20H) byte - Current setting of 3x8 register

(DI + 21H) byte - Current setting of 3x9 register

(DI + 22H) byte - Rows on screen (character lines on screen)

(DI + 23H) word - Character height (scan lines per character)

(DI + 25H) byte - Display-combination code (active)

(DI + 26H) byte - Display-combination code (alternative)

(DI + 27H) word - Colors supported for current video mode

(DI + 29H) byte - Display pages supported for current video mode

(DI + 2AH) byte - Scan lines in current video mode

= 0 - 200 scan lines

= 1 - 350 scan lines

= 2 - 400 scan lines

= 3 - 480 scan lines

= 4 to 255 - Reserved

(DI + 2BH) byte - Primary character block (reserved on Personal  
System/2 Model 25 and Model 30)

= 0 - Block 0

= 1 - Block 1

= 2 - Block 2

. .

. .

. .

= 255 - Block 255

This information is based on the block specifier. See the  
Character Generator function ((AH) = 11H), (AL) = 03H.

(DI + 2CH) byte - Secondary character block (reserved on Personal  
System/2 Model 25 and Model 30)

= 0 - Block 0

= 1 - Block 1

= 2 - Block 2

. .

. .

. .

= 255 - Block 255

This information is based on the block specifier. See the  
Character Generator function ((AH) = 11H), (AL) = 03H.

**(DI+2DH) byte - Miscellaneous state information**

**Bits 7, 6 - Reserved**

**Bit 5 = 0 - Background intensity**  
**= 1 - Blinking**

**Bit 4 = 1 - Cursor emulation active (always set to 0 for**  
**Personal System/2 Model 25 and Model 30)**

**Bit 3 = 1 - Mode setting default palette loading disabled**

**Bit 2 = 1 - Monochrome display attached**

**Bit 1 = 1 - Summing active**

**Bit 0 = 1 - All modes on all displays active (always set to**  
**0 for Personal System/2 Model 25 and Model 30)**

**(DI+2EH) byte - Video that requires the video adapter interface**  
**driver to support modes outside of the current**  
**VGA range**

**Bits 7 to 5 - Reserved**

**Bit 4 = 0 - 132-column mode not supported**  
**= 1 - 132-column mode supported**

**Bit 3 = 0 - VGA attributes set**  
**= 1 - MFI attributes set**

**Bit 2 = 0 - 16-bit VGA graphics not present**  
**= 1 - 16-bit VGA graphics present**

**Bit 1 = 0 - AI driver not required**  
**= 1 - AI driver required**

**Bit 0 = 0 - BIOS does not support AI information return**  
**= 1 - BIOS supports AI information return**

**(DI+2FH) byte - Reserved**

**(DI+30H) byte - Reserved**

**(DI+31H) byte - Video memory available**

**= 0 - 64KB**

**= 1 - 128KB**

**= 2 - 192KB**

**= 3 - 256KB**

**= 4 to 255 - Reserved**

**(DI+32H) byte - Save pointer-state information**

**Bits 7, 6 - Reserved**

**Bit 5 = 1 - DCC extension active**

**Bit 4 = 1 - Palette override active**

**Bit 3 = 1 - Graphics font override active**

**Bit 2 = 1 - Alphanumeric font override active**

**Bit 1 = 1 - Dynamic save area active**

**Bit 0 = 1 - 512-character set active**

**(DI+33H) to (DI+3FH) 13 bytes - Reserved**

**Format of static functionality table:**

**= 0 - Not supported**

**= 1 - Supported**

**(00H) byte - Video modes**

**Bit 7 = 1 - Mode 07H**

**Bit 6 = 1 - Mode 06H**

**Bit 5 = 1 - Mode 05H**

**Bit 4 = 1 - Mode 04H**

**Bit 3 = 1 - Mode 03H**

**Bit 2 = 1 - Mode 02H**

**Bit 1 = 1 - Mode 01H**

**Bit 0 = 1 - Mode 00H**

**(01H) byte - Video modes**

- Bit 7 = 1 - Mode 0FH
- Bit 6 = 1 - Mode 0EH
- Bit 5 = 1 - Mode 0DH
- Bit 4 = 1 - Mode 0CH
- Bit 3 = 1 - Mode 0BH
- Bit 2 = 1 - Mode 0AH
- Bit 1 = 1 - Mode 09H
- Bit 0 = 1 - Mode 08H

**(02H) byte - Video modes**

- Bits 7 to 5 - Reserved
- Bit 4 = 1 - Mode 14H
- Bit 3 = 1 - Mode 13H
- Bit 2 = 1 - Mode 12H
- Bit 1 = 1 - Mode 11H
- Bit 0 = 1 - Mode 10H

See (AH) = 00H on page 2-IN10-2 for video mode information.

**(03H) to (07H) 4 bytes - Reserved**

**(07H) byte - Scan lines available in text modes**

- Bits 7 to 3 - Reserved
- Bit 2 = 400 scan lines
- Bit 1 = 350 scan lines
- Bit 0 = 200 scan lines

See (AH) = 12H, (BL) = 30H on page 2-IN10-22 for text-mode scan-line selection.

**(08H) byte - Character blocks available in text modes**

**(09H) byte - Maximum number of active character blocks in text modes (see (AH) = 11H on page 2-IN10-14 for character-block loading interfaces)**

**(0AH) byte - Miscellaneous functions**

- Bit 7 = Color paging (always set to 0 for Personal System/2 Model 25 and Model 30; see (AH) = 10H on page 2-IN10-10)
- Bit 6 = Color palette (see (AH) = 10H)
- Bit 5 = EGA palette (see (AH) = 10H)
- Bit 4 = Cursor emulation (see (AH) = 01H on page 2-IN10-6)
- Bit 3 = Mode-setting default-palette loading (see (AH) = 12H on page 2-IN10-22)
- Bit 2 = Character-font loading (see (AH) = 11H on page 2-IN10-14)
- Bit 1 = Summing (see (AH) = 10H and (AH) = 12H)
- Bit 0 = All modes on all displays (always set to 0 for Personal System/2 Model 25 and Model 30)

**(0BH) byte - Miscellaneous functions**

**Bits 7 to 4 - Reserved**

**Bit 3 = DCC (see (AH) = 1AH on page 2-IN10-6)**

**Bit 2 = Background intensity/blinking control (see (AH) = 10H on page 2-IN10-10)**

**Bit 1 = Save/restore (always set to 0 for Personal System/2 Model 25 and Model 30; see (AH) = 1CH on page 2-IN10-33)**

**Bit 0 = Light pen (see (AH) = 04H on page 2-IN10-6)**

**(0CH) to (0DH) 2 bytes - Reserved**

**(0EH) byte - Save pointer functions**

**Bits 7, 6 = Reserved**

**Bit 5 = DCC extension (always set to 0 for Personal System/2 Model 25 and Model 30)**

**Bit 4 = Palette override**

**Bit 3 = Graphics font override**

**Bit 2 = Alphanumeric font override**

**Bit 1 = Dynamic save area**

**Bit 0 = 512-character set**

**(0FH) byte - Reserved**

**For all others, no action is performed.**

### **(AH) = 1CH—Save/Restore Video State**

**For Personal System/2 products except Model 25 and Model 30:**

**(AL) = 00H - Return save/restore state buffer size**

**(CX) - Requested states (see supported save/restore states on page 2-IN10-34)**

**On Return:**

**(AL) = 1CH - Function is supported**

**(BX) - Save/restore buffer-size block count (number of 64-byte blocks for saving requested states in (CX))**

**(AL) = 01H - Save state**

**(CX) = Requested states (see supported save/restore states on page 2-IN10-34)**

**(ES:BX) - Buffer pointer to save state**

**On Return:**

**(AL) = 1CH - Function is supported**

**The requested states are saved.**

**(AL) = 02H - Restore state**

**(CX) - Requested states (see supported save/restore states below)**

**(ES:BX) - Buffer pointer to restore state**

**On Return:**

**(AL) = 1CH - Function is supported**

**The requested states are restored.**

**Supported save/restore states:**

**Bits 15 to 3 - Reserved and set to 0**

**Bit 2 = 1 - Save/restore video DAC state and color registers**

**Bit 1 = 1 - Save/restore video BIOS data area**

**Bit 0 = 1 - Save/restore video hardware state**

**Note:** The current video state is altered during the save-state operation. To maintain the current video state, perform a restore-state operation.

For all others, no action is performed.

**(AH) = 1DH to FFH—Reserved**

---

## Interrupt 11H—Equipment Determination

This routine returns information about the optional devices that are attached to the system.

BIOS data area hex 40:10 (installed hardware) is set during POST as follows:

On Return:

(AX) - Equipment flags

Bits 15, 14 - Number of parallel ports

Bit 13 = 1 - Internal modem installed (PC Convertible only)

Bit 12 - Not used

Bits 11 to 9 - Number of asynchronous communication ports

Bit 8 - Not used

Bits 7, 6 - Number of diskette drives installed, other than physical drive 0 (values are binary). These 2 bits do not indicate the locations of the installed drives.

= 00 - No other drives

= 01 - 1 drives

= 10 - 2 drives

= 11 - 3 drives

Bits 5, 4 - Video mode type (values are binary)

= 00 - Reserved

= 01 - 40x25 (color)

= 10 - 80x25 (color)

= 11 - 80x25 (monochrome)

Bit 3 - Not used

Bit 2 = 1 - Pointing device installed

Bit 1 = 1 - Math coprocessor installed

Bit 0 = 0 - Diskette drive 0 is not present

= 1 - Diskette drive 0 is present

## Notes:



---

## Interrupt 12H—Memory Size Determination

This routine returns the amount of RAM up to 640KB in the system as determined by POST, minus the memory that is allocated to the extended BIOS data area. See Interrupt 15H, Return Extended BIOS Data Area Segment Address function ((AH)=C1H) and Interrupt 15H, Extended-Memory Size Determination function ((AH)=88H) for additional information.

On return, (AX) contains the number of contiguous 1KB blocks of memory.

**Notes:**

## Interrupt 13H—Diskette

This interface provides access to diskette drives. The following is a summary of the diskette functions of Interrupt 13H.

(AH) = 00H	— Reset Diskette System
(AH) = 01H	— Read Status of Last Operation
(AH) = 02H	— Read Desired Sectors into Memory
(AH) = 03H	— Write Desired Sectors from Memory
(AH) = 04H	— Verify Desired Sectors
(AH) = 05H	— Format Desired Track
(AH) = 06H to 07H	— Reserved
(AH) = 08H	— Read Drive Parameters
(AH) = 09H to 14H	— Reserved
(AH) = 15H	— Read Diskette Drive Type
(AH) = 16H	— Diskette Change Line Status
(AH) = 17H	— Set Diskette Type for Format
(AH) = 18H	— Set Media Type for Format
(AH) = 19H	— Reserved
(AH) = 20H	— Get Media Type
(AH) = 21H to FFH	— Reserved

*Figure 2-7. INT 13H Diskette Functions*

### Notes:

1. All reserved input fields must be set to 0.
2. For the diskette drive parameters, see the Diskette Drive Parameter Table in the "ROM Tables" section.

For AT, PC/XT BIOS dated 1/10/86 and later, PC/XT Model 286, PC Convertible, and Personal System/2 products, operations that require the diskette-drive motor to be turned on call Interrupt 15H, Diskette Drive Motor Start function ((AX)=90FDH). This allows the operating system to perform a different task while waiting for the diskette-drive motor to accelerate.

Before waiting for the diskette interrupt, BIOS calls Interrupt 15H, Device Busy function ((AH)=90H) with (AL)=01H (Type=Diskette). This informs the operating system of the wait. The complementary Interrupt 15H, Interrupt Complete function ((AH)=91H) with (AL)=01H (Type=Diskette) is called to indicate that the operation is complete. See "Multitasking Provisions" in the "Additional Information" section.

If the caller changes the values of the head settle time (byte 9) and the motor startup time (byte 10) to values that are inconsistent with the diskette drive specifications, BIOS enforces the minimum values for these parameters as specified for the diskette drive. (For the

| diskette drive parameters, see the Diskette Drive Parameter Table in  
| the "ROM Tables" section.) The values of these parameters can be  
| increased to allow for correcting possible future problems if some  
| diskette drives require more than the nominal values for these  
| parameters.

### **(AH) = 00H—Reset Diskette System**

(DL) - Drive number (0-based)

Bit 7 = 0 - Diskette (value checked)

On Return:

(AH) - Status of operation

= 00H - No error

= 01H - Invalid diskette parameter

= 02H - Address mark not found

= 03H - Write-protect error

= 04H - Requested sector not found

= 06H - 'Diskette change' line active

= 08H - DMA overrun on operation

= 09H - DMA attempt across a 64KB boundary

= 0CH - Media type not found

= 10H - Cyclic redundancy check (CRC) error on diskette  
read

= 20H - General controller failure

= 30H - Drive does not support media sense

= 31H - No media in drive

= 32H - Media type not supported by drive

= 40H - Seek operation failed

= 80H - Diskette drive not ready

CF = 0 - Status is 0

= 1 - Status is non 0

Address hex 40:41 is set to the status of operation.

### **Notes:**

1. If the diskette BIOS reports an error, reset the diskette system and retry the operation.
2. If the value in (DL) is greater than or equal to hex 80, the diskette system is reset, then the fixed-disk system is reset. The status returned in (AH) is the status of the fixed-disk reset. Read the status of the diskette system after completing the operation.

## **(AH) = 01H—Read Status of Last Operation**

(DL) - Drive number (0-based)  
Bit 7 = 0 - Diskette (value checked)

### **On Return:**

(AH) - Status of operation (see values for the status of operation in (AH) = 00H on page 2-IN13D-2)

CF = 0 - Status is 0  
= 1 - Status is non 0

## **(AH) = 02H—Read Desired Sectors Into Memory**

(AL) - Number of sectors (not value checked)  
(CH) - Track number (not value checked; low 8 bits of 10-bit track number, 0-based)  
(CL) - Bits 7, 6 - High 2 bits of 10-bit track number, 0-based  
Bits 5 to 0 - Sector number (not value checked)  
(DH) - Head number (not value checked, 0-based)  
(DL) - Drive number (0-based)  
Bit 7 = 0 - Diskette (value checked)  
(ES:BX) - Address of buffer

### **On Return:**

(AH) - Status of operation (see values for the status of operation in (AH) = 00H on page 2-IN13D-2)

(AL) - Number of sectors actually transferred

CF = 1 - Status is non 0  
= 0 - Status is 0

Address hex 40:41 is set to the status of operation.

**Note:** If the diskette BIOS reports an error, reset the diskette system and retry the operation.

## **(AH) = 03H—Write Desired Sectors from Memory**

(AL) - Number of sectors (not value checked)  
(CH) - Track number (not value checked; low 8 bits of 10-bit track number, 0-based)  
(CL) - Bits 7, 6 - High 2 bits of 10-bit track number, 0-based  
Bits 5 to 0 - Sector number (not value checked)  
(DH) - Head number (not value checked, 0-based)  
(DL) - Drive number (0-based)  
Bit 7 = 0 - Diskette (value checked)  
(ES:BX) - Address of buffer

**On Return:**

(AH) - Status of operation (see values for the status of operation in (AH)=00H on page 2-IN13D-2)

(AL) - Number of sectors actually transferred

CF = 0 - Status is 0

= 1 - Status is non 0

Address hex 40:41 is set to the status of operation.

**Notes:**

1. If the diskette BIOS reports an error, reset the diskette system and retry the operation.
2. For PC/XT Model 286, (AL) is not required.

**(AH) = 04H—Verify Desired Sectors**

(AL) - Number of sectors (not value checked)

(CH) - Track number (not value checked; low 8 bits of 10-bit track number, 0-based)

(CL) - Bits 7, 6 - High 2 bits of 10-bit track number, 0-based

Bits 5 to 0 - Sector number (not value checked)

(DH) - Head number (not value checked, 0-based)

(DL) - Drive number (0-based)

Bit 7 = 0 - Diskette (value checked)

(ES:BX) - Address of buffer

**On Return:**

(AH) - Status of operation (see values for the status of operation in (AH)=00H on page 2-IN13D-2)

(AL) - Number of sectors verified

CF = 0 - Status is 0

= 1 - Status is non 0

Address hex 40:41 is set to the status of operation.

**Notes:**

1. If the diskette BIOS reports an error, reset the diskette system and retry the operation.
2. (ES:BX) is not required for AT BIOS dated 11/15/85 and later, PC/XT Model 286, PC Convertible, or Personal System/2 products.

## **(AH) = 05H—Format Desired Track**

The buffer pointer (ES:BX) must point to the collection of desired address fields for the track. Each field has the following 4 bytes:

Byte 0 - Track number  
Byte 1 - Head number  
Byte 2 - Sector number  
Byte 3 - Number of bytes per sector  
= 00H - Reserved  
= 01H - Reserved  
= 02H - 512 bytes per sector  
= 03H - Reserved

There must be one entry for every sector on the track. This information is used to find the requested sector during read or write access. Before a diskette is formatted, if more than one format is supported for the drive that is being used, it is necessary to call Interrupt 13H, Set Diskette Type for Format function ((AH)=17H) or Interrupt 13H, Set Media Type for Format function ((AH)=18H) to specify the diskette type that is to be formatted.

(AL) - Number of sectors (not value checked)  
(CH) - Track number (not value checked; low 8 bits of 10-bit track number, 0-based)  
(CL) - Bits 7, 6 - High 2 bits of 10-bit track number, 0-based  
Bits 5 to 0 - Not used  
(DH) - Head number (not value checked, 0-based)  
(DL) - Drive number (0-based)  
Bit 7 = 0 - Diskette (value checked)  
(ES:BX) - Address of buffer

### **On Return:**

(AH) - Status of operation (see values for the status of operation in (AH)=00H on page 2-IN13D-2)  
CF = 0 - Status is 0  
= 1 - Status is non 0

Address hex 40:41 is set to the status of operation.

### **Notes:**

1. If the diskette BIOS reports an error, reset the diskette system and retry the operation.
2. The diskette drive parameter table is used to format the diskette. See the Diskette Drive Parameter Table in the "ROM Tables" section.
3. For PC/XT Model 286, (AL) is not required.

**(AH) = 06H to 07H—Reserved**

**(AH) = 08H—Read Drive Parameters**

Each supported media type has a parameter table.

For PCjr, PC, PC/XT, and for AT BIOS dated 1/10/84, this function is not supported:

On Return:

(AH) - Status of operation  
= 01H - Invalid command  
CF = 1 - Error

Address hex 40:41 is set to the status of operation.

For all others:

(DL) - Drive number (0-based)  
Bit 7 = 0 - Diskette (value checked)

On Return:

(AH) = Status of operation  
(AL) = 0  
(BH) = 0  
(BL) - Bits 7 to 4 = 0

Bits 3 to 0 - Valid drive type value in CMOS  
= 01H - 360KB, 5.25 inch, 40 track  
= 02H - 1.2MB, 5.25 inch, 80 track  
= 03H - 720KB, 3.5 inch, 80 track  
= 04H - 1.44MB, 3.5 inch, 80 track  
= 05H - 720KB, 5.25 inch, 80 track  
= 06H - 2.88MB, 3.5 inch, 80 track

(KB = 1024 bytes; MB = 1 048 576 bytes.)

(CH) - Maximum number of tracks (low 8 bits of 10-bit track number, 0-based)

(CL) - Bits 7, 6 - Maximum number of tracks (high 2 bits of 10-bit track number, 0-based)

- Bits 5 to 0 - Maximum sectors per track

(DH) - Maximum head number

(DL) - Number of diskette drives installed

(ES:DI) - Pointer to 11-byte parameter table that is associated with the maximum supported media type within the drive (see the Diskette Drive Parameter Table in the "ROM Tables" section)

All registers are returned as described above, except that (BL) is set to 0, when the drive type is known and any of the following conditions exists:



- The CMOS type is invalid.
- The CMOS is not present.
- The CMOS battery is discharged.
- The CMOS checksum is invalid.

If the requested drive is not installed, (AX), (BX), (CX), (DX), (DI), and (ES) are set to 0.

Address hex 40:41 is set to 0, and CF is set to 0.

For drive number hex 80 or above (indicating fixed disks), this function is not supported:

(AH) - Status of operation  
       = 01H - Invalid command  
 CF = 1 - Error

(ES), (AX), (BX), (CX), (DH), and (DI) are equal to 0, and (DL) contains the number of drives when any of the following conditions exists:

- The drive number is invalid.
- The drive type is unknown, and the CMOS is not present.
- The CMOS battery is discharged, or the CMOS checksum is invalid.
- The drive type is unknown, and the CMOS drive type is invalid.

Address hex 40:41 is set to 0, and CF is set to 0.

### **(AH) = 09H to 14H—Reserved**

### **(AH) = 15H—Read Diskette Drive Type**

For AT, PC/XT BIOS dated 1/10/86 and later, PC/XT Model 286, PC Convertible, and Personal System/2 products:

(DL) - Drive number (0-based)  
       Bit 7 = 0 - Diskette (value checked)

On Return:

(AH) = 00H - Drive not present  
       = 01H - Diskette, no 'diskette change' line available  
       = 02H - Diskette, 'diskette change' line available  
       = 03H - Reserved  
 CF = 0 - Operation successfully completed

Address hex 40:41 is set to the status of operation.

For all others, this function is not supported:

On Return:

(AH) - Status of operation  
= 01H - Invalid command  
CF = 1 - Error

Address hex 40:41 is set to the status of operation.

### **(AH) = 16H—'Diskette Change' Line Status**

For AT, PC/XT BIOS dated 1/10/86 and later, PC/XT Model 286, PC Convertible, and Personal System/2 products:

(DL) - Drive number (0-based)  
Bit 7 = 0 - Diskette (value checked)

On Return:

(AH) = 00H - 'Diskette change' signal not active  
= 01H - Invalid diskette parameter  
= 06H - 'Diskette change' signal active  
= 80H - Diskette drive not ready  
CF = 0 if (AH) is 0  
= 1 if (AH) is non 0

Address hex 40:41 is set to the value in (AH).

For all others, this function is not supported:

On Return:

CF = 1 - Error  
(AH) - Status of operation  
= 01H - Invalid command

Address hex 40:41 is set to the status of operation.

### **(AH) = 17H—Set Diskette Type for Format**

For AT, PC/XT BIOS dated 1/10/86 and later, PC Convertible, and Personal System/2 products:

(DL) - Drive number (0-based)  
Bit 7 = 0 - Diskette (value checked)

- (AL) = 00H - Invalid request
  - = 01H - Diskette 320/360KB in 360KB drive
  - = 02H - Diskette 360KB in 1.2MB drive
  - = 03H - Diskette 1.2MB in 1.2MB drive
  - = 04H - AT BIOS before 6/10/85: Invalid request
    - All others: Diskette 720KB in 720KB drive
  - = 05H to 0FFH - Invalid request

**On Return:**

- (AH) - Status of operation (see values for the status of operation in (AH)=00H on page 2-IN13D-2)
- CF = 0 - Status is 0
  - = 1 - Status is non 0

Address hex 40:41 is set to the status of operation.

**For all others, this function is not supported:**

**On Return:**

- CF = 1 - Error
- (AH) - Status of operation
  - = 01H - Invalid command

Address hex 40:41 is set to the status of operation.

## **(AH) = 18H—Set Media Type for Format**

For AT BIOS dated 11/15/85 and later, PC/XT BIOS dated 1/10/86 and later, PC/XT Model 286, and Personal System/2 products, this function is called before Interrupt 13H, Format the Desired Track function ((AH)=05H) is issued. If the diskette is changed, the function is called again. A diskette must be present in the drive.

Each supported media type has a parameter table.

- (DL) - Drive number (0-based)
  - Bit 7 = 0 - Diskette (value checked)
- (CH) - Number of tracks (low 8 bits of 10-bit track number, 0-based)
- (CL) - Bits 7, 6 - Number of tracks (high 2 bits of 10-bit track number, 0-based)
  - Bits 5 to 0 - Sectors per track

**On Return:**

(ES:DI) - Pointer to 11-byte parameter table for this media type, unchanged if (AH) is non 0 (see the Diskette Drive Parameter Table in the "Data Areas and ROM Tables" section)

(AH) - Status of operation (see values for the status of operation in (AH)=00H on page 2-IN13D-2)

CF = 0 - Status is 0

= 1 - Status is non 0

**Note:** For PC/XT Model 286 and Personal System/2 products, this function monitors the 'diskette change' signal. If the signal is active, BIOS attempts to reset the 'diskette change' line to the inactive state. If the attempt is successful (for example, when media is present), BIOS sets the correct data rate for formatting. If the attempt fails (for example, when no media is present), BIOS returns hex 80 in (AH) (Diskette Drive Not Ready), and the carry flag is set to 1.

When the 'diskette change' signal is inactive, BIOS performs the function as requested.

For all others, this function is not supported:

**On Return:**

(AH) - Status of operation

= 01H - Invalid command

CF = 1 - Error

Address hex 40:41 is set to the status of operation.

**(AH) = 19H—Reserved**

**| (AH) = 20H—Get Media Type**

(DL) - Drive number (0-based)

Bit 7 = 0 - Diskette (value checked)

**On Return:**

(AH) - Status of operation

= 01H - Invalid request

= 30H - Drive does not support media sense

= 31H - No media in drive

= 32H - Drive does not support media type

(AL) - Type of media installed

= 06H - 4MB diskette

= 04H - 2MB diskette

= 03H - 1MB diskette

- All others are reserved

CF = 0 - Status is 0

= 1 - Status is non 0

**(AH) = 21H to FFH—Reserved**

**| Supported Drives and Media**

**| Interrupt 13H supports the following drives.**

5.25"	40 track	180KB	One head
5.25"	40 track	360KB	Two heads
5.25"	80 track	1.2MB	Two heads
3.5"	80 track	720KB	Two heads
3.5"	80 track	1.44MB	Two heads
3.5"	80 track	2.88MB	Two heads

**| Figure 2-8. INT 13H Supported Diskette Drives**

**| Interrupt 13H supports the following media.**

5.25"	40 track	8 sectors/track	320KB	Double sided
5.25"	40 track	9 sectors/track	360KB	Double sided
5.25"	80 track	15 sectors/track	1.2MB	Double sided
3.5"	80 track	9 sectors/track	720KB	Double sided
3.5"	80 track	18 sectors/track	1.44MB	Double sided
3.5"	80 track	36 sectors/track	2.88MB	Double sided

**| Figure 2-9. INT 13H Supported Media**

## **| Diskette Drive Parameters**

**| The following tables contain the recommended parameters for diskette drives that are supported on Personal System/2 products.**

<b>Byte Definition</b>	<b>320K Media</b>	<b>360K Media</b>
First specification byte	D0H	D0H
Second specification byte	02H	02H
Motor-off time	25H	25H
Bytes per sector	02H	02H
Sectors per track	08H	09H
Gap length	2AH	2AH
Data length	FFH	FFH
Gap length (format)	50H	50H
Fill byte (format)	F6H	F6H
Head settle time (in microseconds)	0FH	0FH
Motor start (in 1/6-seconds)	08H	08H
Maximum track numbers	27H	27H
Data-transfer rate	80H	80H
Multi-rate capability	00H	00H

**| Figure 2-10. Media Parameter Table — 360KB Slimline Drive**

<b>Byte Definition</b>	<b>720K Media</b>
First specification byte	D0H
Second specification byte	02H
Motor-off time	25H
Bytes per sector	02H
Sectors per track	09H
Gap length	2AH
Data length	FFH
Gap length (format)	50H
Fill byte (format)	F6H
Head settle time (in microseconds)	0FH
Motor start (in 1/6-seconds)	04H
Maximum track numbers	4FH
Data-transfer rate	80H
Multi-rate capability	00H

**| Figure 2-11. Media Parameter Table — 720KB Slimline Drive**

<b>Byte Definition</b>	<b>320K Media</b>	<b>360K Media</b>	<b>1.2M Media</b>
First specification byte	E0H	E0H	D0H
Second specification byte	02H	02H	02H
Motor-off time	25H	25H	25H
Bytes per sector	02H	02H	02H
Sectors per track	08H	09H	0FH
Gap length	2AH	2AH	1BH
Data length	FFH	FFH	FFH
Gap length (format)	50H	50H	54H
Fill byte (format)	F6H	F6H	F6H
Head settle time (in microseconds)	0FH	0FH	0FH
Motor start (in 1/6-seconds)	04H	04H	04H
Maximum track numbers	27H	27H	4FH
Data-transfer rate	40H	40H	00H
Multi-rate capability	02H	02H	02H

**Figure 2-12. Media Parameter Table – 1.2MB Slimline Drive**

<b>Byte Definition</b>	<b>720K Media</b>	<b>1.44M Media</b>
First specification byte	E0H	D0H
Second specification byte	02H	02H
Motor-off time	25H	25H
Bytes per sector	02H	02H
Sectors per track	08H	12H
Gap length	2AH	1BH
Data length	FFH	FFH
Gap length (format)	50H	65H
Fill byte (format)	F6H	F6H
Head settle time (in microseconds)	0FH	0FH
Motor start (in 1/6-seconds)	04H	04H
Maximum track numbers	4FH	4FH
Data-transfer rate	80H	00H
Multi-rate capability	02H	02H

**Figure 2-13. Media Parameter Table – 1.44MB Slimline Drive**

<b>Byte Definition</b>	<b>720K Media</b>	<b>1.44M Media</b>	<b>2.88M Media</b>
First specification byte	E0H	D0H	A0H
Second specification byte	02H	02H	02H
Motor-off time	25H	25H	25H
Bytes per sector	02H	02H	02H
Sectors per track	09H	12H	24H
Gap length	2AH	1BH	38H
Data length	FFH	FFH	FFH
Gap length (format)	50H	65H	53H
Fill byte (format)	F6H	F6H	F6H
Head settle time (in microseconds)	0FH	0FH	0FH
Motor start (in 1/6-seconds)	04H	04H	04H
Maximum track numbers	4FH	4FH	4FH
Data-transfer rate	80H	00H	C0H
Multi-rate capability	02H	02H	02H

**Figure 2-14. Media Parameter Table – 2.88MB Slimline Drive**

<b>Byte Definition</b>	<b>720K Media</b>	<b>1.44M Media</b>
First specification byte	D0H	A0H
Second specification byte	02H	02H
Motor-off time	25H	25H
Bytes per sector	02H	02H
Sectors per track	09H	12H
Gap length	2AH	1BH
Data length	FFH	FFH
Gap length (format)	50H	65H
Fill byte (format)	F6H	F6H
Head settle time (in microseconds)	0FH	0FH
Motor start (in 1/6-seconds)	04H	04H
Maximum track numbers	4FH	4FH
Data-transfer rate	80H	00H
Multi-rate capability	02H	02H

**Figure 2-15. Media Parameter Table – 1.44MB Half-High Drive**



## Interrupt 13H—Fixed Disk

This interface provides access to fixed disk drives. The following is a summary of the fixed disk functions of Interrupt 13H.

(AH) = 00H	— Reset Disk System
(AH) = 01H	— Read Status of Last Operation
(AH) = 02H	— Read Desired Sectors into Memory
(AH) = 03H	— Write Desired Sectors from Memory
(AH) = 04H	— Verify Desired Sectors
(AH) = 05H	— Format Desired Track
(AH) = 06H	— Format Desired Track and Set Bad-Sector Flags
(AH) = 07H	— Format Drive Starting at Desired Cylinder
(AH) = 08H	— Read Drive Parameters
(AH) = 09H	— Initialize Drive Pair Characteristics
(AH) = 0AH to 0BH	— Reserved
(AH) = 0CH	— Seek
(AH) = 0DH	— Alternative Disk Reset
(AH) = 0EH to 0FH	— Reserved
(AH) = 10H	— Test Drive Ready
(AH) = 11H	— Recalibrate
(AH) = 12H to 14H	— Reserved
(AH) = 15H	— Read DASD Type
(AH) = 16H to 18H	— Reserved
(AH) = 19H	— Park Heads
(AH) = 1AH to 20H	— Reserved
(AH) = 21H	— Read Multiple Sectors into Memory
(AH) = 22H	— Write Multiple Sectors from Memory
(AH) = 23H	— Reserved
(AH) = 24H	— Set Multiple Mode
(AH) = 25H	— Identify Drive
(AH) = 26H to FFH	— Reserved

**Figure 2-16.** INT 13H Fixed Disk Functions

### Notes:

1. All reserved input fields must be set to 0.
2. If a fixed disk drive adapter is not installed, the code is not hooked into Interrupt 13H; the values that are returned are described in "Interrupt 13H—Diskette."
3. For the fixed disk interface, the drive number in (DL) is value checked for all functions that use the device number.
4. For AT, PC/XT Model 286, and Personal System/2 products, before waiting for this interrupt, BIOS calls Interrupt 15H, Device Busy function ((AH)=90H) with (AL)=00H (Type=Disk) to inform the operating system of the wait. The complementary Interrupt 15H, Interrupt Complete function ((AH)=91H) with (AL)=00H (Type=Disk) is called to indicate that the operation is complete.

5. For Personal System/2 products, before waiting for the fixed disk reset, BIOS calls Interrupt 15H, Device Busy function ((AH)=90H) with (AL)=FCH (Type=Fixed Disk Reset). This is a time-out-only function. There is no complementary power-on self-test (POST) operation. (See "Multitasking Provisions" in the "Additional Information" section.)
6. Bit 7 of the drive number in (DL) must be set upon entry to the fixed disk BIOS.
7. For the drive parameters see the Fixed Disk Drive Parameter Table in the "ROM Tables" section.

### **(AH) = 00H—Reset Disk System**

(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

On Return:

(AH) - Status of operation

- = 00H - No error
- = 01H - Unsupported function or parameter
- = 02H - Address mark not found
- = 03H - Write-protect error
- = 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 detected
- = 0BH - Bad cylinder detected
- = 0DH - Invalid number of sectors on format
- = 0EH - Control data address mark detected
- = 0FH - DMA arbitration level out of range
- = 10H - Uncorrectable error checking and correction (ECC)  
or cyclic redundancy check (CRC) error
- = 11H - ECC corrected data error
- = 12H - Command in progress
- = 13H - Device not powered-on
- = 20H - General controller failure
- = 40H - Seek operation failed
- = 80H - Time-out
- = AAH - Drive not ready
- = BBH - Undefined error occurred
- = CCH - Write fault on selected drive
- = E0H - Status error/error register = 0
- = FFH - Sense operation failed

CF = 0 - Status is 0

= 1 - Status is non 0

Address hex 40:74 is set to the status of operation.

## Notes:

1. This function is issued only if the 7-bit drive number is less than the maximum number of fixed disk drives that are installed. The diskette system is also reset for all values of (DL).
2. For Personal System/2 products, before waiting for the fixed disk reset, BIOS calls Interrupt 15H, Device Busy function ((AH)=90H) with (AL)=FCH (Type=Fixed Disk Reset) to inform the operating system of the wait.
3. For Personal System/2 Model 35 SX, Model 35 LS, Model 40 SX, and Model L40 SX, both the master drive and the slave drive respond to the Reset function that is issued to either drive. Both drives are reset.

## (AH) = 01H—Read Status of Last Operation

(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

On Return:

(AH) - Status of this operation (see values for the status of operation on page 2-IN13F-2)

(AL) - Status of the last operation

CF = 0 - Status is 0

    = 1 - Status is non 0

Address hex 40:74 is set to 0.

**Note:** This function returns the status of the last operation that was performed on the specified drive. The result is not valid if another drive has been accessed since the last operation was performed on the specified drive.

## (AH) = 02H—Read Desired Sectors Into Memory

(AL) - Number of sectors

(CH) - Cylinder number (low 8 bits of 10-bit cylinder number, 0-based, not value checked)

(CL) - Bits 7, 6 - Cylinder number (high 2 bits of 10-bit cylinder number, 0-based, not value checked)

    - Bits 5 to 0 - Sector number (not value checked)

(DH) - Head number (0-based, not value checked)

(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

(ES:BX) - Address of buffer

**On Return:**

(AH) - Status of operation (see values for the status of operation on page 2-IN13F-2)

CF = 0 - Status is 0

= 1 - Status is non 0

Address hex 40:74 is set to the status of operation.

**Notes:**

1. (AH) = 11H indicates that the data that was read had a recoverable error that was corrected by the error checking and correction (ECC) algorithm. The data is good; however, the BIOS routine indicates an error to notify the controlling program of the correction. The error might not recur if the data is rewritten.
2. If the fixed disk BIOS reports an error, reset the disk system and retry the operation.

**(AH) = 03H—Write Desired Sectors from Memory**

(AL) - Number of sectors

(CH) - Cylinder number (low 8 bits of 10-bit cylinder number, 0-based, not value checked)

(CL) - Bits 7, 6 - Cylinder number (high 2 bits of 10-bit cylinder number, 0-based, not value checked)

- Bits 5 to 0 - Sector number (not value checked)

(DH) - Head number (0-based, not value checked)

(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

(ES:BX) - Address of buffer

**On Return:**

(AH) - Status of operation (see values for the status of operation on page 2-IN13F-2)

CF = 0 - Status is 0

= 1 - Status is non 0

Address hex 40:74 is set to the status of operation.

**Note:** If the fixed disk BIOS reports an error, reset the disk system and retry the operation.

## **(AH) = 04H—Verify Desired Sectors**

(AL) - Number of sectors

(CH) - Cylinder number (low 8 bits of 10-bit cylinder number, 0-based, not value checked)

(CL) - Bits 7, 6 - Cylinder number (high 2 bits of 10-bit cylinder number, 0-based, not value checked)

- Bits 5 to 0 - Sector number (not value checked)

(DH) - Head number (0-based, not value checked)

(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

### **On Return:**

(AH) - Status of operation (see values for the status of operation on page 2-IN13F-2)

CF = 0 - Status is 0

= 1 - Status is non 0

Address hex 40:74 is set to the status of operation.

**Note:** If the fixed disk BIOS reports an error, reset the disk system and retry the operation.

## **(AH) = 05H—Format Desired Track**

(CH) - Cylinder number (low 8 bits of 10-bit cylinder number, 0-based, not value checked)

(CL) - Bits 7, 6 - Cylinder number (high 2 bits of 10-bit cylinder number, 0-based, not value checked)

- Bits 5 to 0 - Sector number (not value checked)

(DH) - Head number (0-based, not value checked)

(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

### **For PC/XT:**

(AL) - Interleaving value

### **On Return:**

(AH) - Status of operation (see values for the status of operation on page 2-IN13F-2)

CF = 0 - Status is 0

= 1 - Status is non 0

**For AT, PC/XT Model 286, and Personal System/2 products:**

**(ES:BX) - Address of buffer**

**(ES:BX) points to a 512-byte buffer. The first  
[2 × (number of sectors per track)] bytes contain *F* and *N*  
for each sector, where**

***F* = 00H - Good sector**

**= 80H - Bad sector**

***N* - Sector number**

**On Return:**

**(AH) - Status of operation (see values for the status of  
operation on page 2-IN13F-2)**

**CF = 0 - Status is 0**

**= 1 - Status is non 0**

**Address hex 40:74 is set to the status of operation.**

**For any device that uses enhanced small device interface (ESDI)  
type or small computer system interface (SCSI) type commands,  
this function is not supported:**

**On Return:**

**(AH) - Status of operation**

**= 01H - Invalid function request**

**CF = 1 - Error**

**Address hex 40:74 is set to the status of operation.**

**Note: If the fixed disk BIOS reports an error, reset the disk  
system and retry the operation.**

## **(AH) = 06H—Format Desired Track and Set Bad-Sector Flags**

**Warning:** Formatting destroys all information on the fixed disk.

For PC/XT:

- (AL) - Interleaving value
- (CH) - Cylinder number (low 8 bits of 10-bit cylinder number, 0-based, not value checked)
- (CL) - Bits 7, 6 - Cylinder number (high 2 bits of 10-bit cylinder number, 0-based, not value checked)
  - Bits 5 to 0 - Sector number (not value checked)
- (DH) - Head number (0-based, not value checked)
- (DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

On Return:

(AH) - Status of operation (see values for the status of operation on page 2-IN13F-2)

CF = 0 - Status is 0

= 1 - Status is non 0

Address hex 40:74 is set to the status of operation.

For AT, PC/XT Model 286, Personal System/2 products, and any device that uses ESDI- or SCSI-type commands, this function is not supported:

On Return:

(AH) - Status of operation

= 01H - Invalid function request

CF = 1 - Error

**Note:** If the fixed disk BIOS reports an error, reset the disk system and retry the operation.

## **(AH) = 07H—Format Drive Starting at Desired Cylinder**

For PC/XT:

- (AL) - Interleaving value
- (CH) - Cylinder number (low 8 bits of 10-bit cylinder number, 0-based, not value checked)
- (CL) - Bits 7, 6 - Cylinder number (high 2 bits of 10-bit cylinder number, 0-based, not value checked)
  - Bits 5 to 0 - Sector number (not value checked)
- (DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

On Return:

(AH) - Status of operation (see values for the status of operation on page 2-IN13F-2)

CF = 0 - Status is 0

= 1 - Status is non 0

Address hex 40:74 is set to the status of operation.

For AT, PC/XT Model 286, Personal System/2 products, and any device that uses ESDI- or SCSI-type commands, this function is not supported:

On Return:

(AH) - Status of operation

= 01H - Invalid function request

CF = 1 - Error

**Note:** If the fixed disk BIOS reports an error, reset the disk system and retry the operation.

## **(AH) = 08H—Read Drive Parameters**

If the drive number is not valid, (AH) and address hex 40:74 are set to hex 07 (last fixed disk drive operation status), (CX) and (DX) are set to 0, and CF is set to 1. If no fixed disk drive is attached or no fixed disk drive adapter is installed, (AH) and address hex 40:41 are set to hex 01 (last diskette drive operation status), and CF is set to 1.

(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

On Return:

(CH) - Maximum value for cylinder number (range is from 0 to hex 3FFF)

(CL) - Maximum value for sector and high-order 2 bits of cylinder numbers

(DL) - Number of consecutive drives attached

(DH) - Maximum value for head number (range is from 0 to hex 3F)



## **(AH) = 09H—Initialize Drive Pair Characteristics**

(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

### **On Return:**

(AH) - Status of operation (see values for the status of operation on page 2-IN13F-2)

CF = 0 - Status is 0

= 1 - Status is non 0

### **For PC/XT:**

Interrupt 41H points to the parameter tables. Four entries in the PC/XT table correspond to the switch setting on the fixed disk drive adapter. The switches act as an index into the parameter table. For example, if both switches are set to the "on" position, the drive is initialized with the first entry of the parameter table. If the drive number is an allowable value (that is,  $80H \leq (DL) \leq 87H$ ), both drive 0 and drive 1 are initialized. For all other values, an unsupported-command status is returned. If drive 0 initialization fails, drive 1 initialization is not attempted. If either attempt fails, address hex 40:74 (last fixed disk drive operation status) and (AH) are updated with the appropriate error code.

For AT, PC/XT Model 286, and Personal System/2 products with Micro Channel\* architecture that do not have ESDI or SCSI:

Interrupt 41H points to the single parameter table for drive 0, and Interrupt 46H points to the single parameter table for drive 1. If (DL) = 80H, drive 0 is initialized through Interrupt 41H. If (DL) = 81H, drive 1 is initialized through Interrupt 46H. For all other values, an unsupported-command status is returned.

---

\* Micro Channel is a trademark of the International Business Machines Corporation.

For any device that uses ESDI- or SCSI-type commands:

This function performs no action. Drive configuration information is obtained from the drive, not from a table in the system ROM. The controller automatically performs drive-type initialization.

**Note:** If the fixed disk BIOS reports an error, reset the disk system and retry the operation.

#### **(AH) = 0AH to 0BH—Reserved**

#### **(AH) = 0CH—Seek**

(CH) - Cylinder number (low 8 bits of 10-bit cylinder number, 0-based, not value checked)

(CL) - Bits 7, 6 - Cylinder number (high 2 bits of 10-bit cylinder number, 0-based, not value checked)  
- Bits 5 to 0 - Sector number (not value checked)

(DH) - Head number (0-based, not value checked)

(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

On Return:

(AH) - Status of operation (see values for the status of operation on page 2-IN13F-2)

CF = 0 - Status is 0

= 1 - Status is non 0

Address hex 40:74 is set to the status of operation.

**Note:** If the fixed disk BIOS reports an error, reset the disk system and retry the operation.

#### **(AH) = 0DH—Alternative Disk Reset**

(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

On Return:

(AH) - Status of operation (see values for the status of operation on page 2-IN13F-2)

CF = 0 - Status is 0

= 1 - Status is non 0

Address hex 40:74 is set to the status of operation.

**Notes:**

1. The Alternative Disk Reset function is issued only if the 7-bit drive number is less than the maximum number of fixed disk drives that are installed. The diskette system is not reset.
2. For Personal System/2 Model 35 SX, Model 35 LS, Model 40 SX, and Model L40 SX, both the master drive and the slave drive respond to the Reset function that is issued to either drive. Both drives are reset.

**(AH) = 0EH to 0FH—Reserved**

**(AH) = 10H—Test Drive Ready**

(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

On Return:

(AH) - Status of operation (see values for the status of operation on page 2-IN13F-2)

CF = 0 - Status is 0  
= 1 - Status is non 0

Address hex 40:74 is set to the status of operation.

**(AH) = 11H—Recalibrate**

(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

On Return:

(AH) - Status of operation (see values for the status of operation on page 2-IN13F-2)

CF = 0 - Status is 0  
= 1 - Status is non 0

Address hex 40:74 is set to the status of operation.

**Note:** If the fixed disk BIOS reports an error, reset the disk system and retry the operation.

**(AH) = 12H to 14H—Reserved**

## **(AH) = 15H—Read DASD Type**

For PC/XT, this function is not supported:

On Return:

- (AH) - Status of operation
  - = 01H - Invalid function request
- CF = 1 - Error

Address hex 40:74 is set to the status of operation.

For AT, PC/XT Model 286, and Personal System/2 products:

(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

On Return:

- (CX,DX) - Number of 512-byte blocks
  - DX - Least significant byte
  - CX - Most significant byte
  - If (AH) = 0, (CX) = 0 and (DX) = 0
- (AH) = 00H - Drive not present or (DL) is not supported
  - = 01H - Reserved
  - = 02H - Reserved
  - = 03H - Fixed disk
- CF = 0 - Operation successfully completed

Address hex 40:74 is set to the status of operation.

## **(AH) = 16H to 18H—Reserved**

## **(AH) = 19H—Park Heads**

For PC/XT, AT, and PC/XT Model 286, this function is not supported:

On Return:

- (AH) - Status of operation
  - = 01H - Invalid function request
- CF = 1 - Error

Address hex 40:74 is set to the status of operation.

**For Personal System/2 products except Model 35 SX,  
Model 35 LS, Model 40 SX, and Model L40 SX:**

**(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)**

**On Return:**

**(AH) - Status of operation (see values for the status of  
operation on page 2-IN13F-2)**

**CF = 0 - Status is 0**

**= 1 - Status is non 0**

**Address hex 40:74 is set to the status of operation.**

### **(AH) = 1AH to 20H—Reserved**

### **(AH) = 21H—Read Multiple Sectors Into Memory**

**For Personal System/2 Model 35 SX, Model 35 LS, Model 40 SX,  
and Model L40 SX:**

**(AL) - Number of sectors**

**(CH) - Cylinder number (low 8 bits of 10-bit cylinder number,  
0-based, not value checked)**

**(CL) - Bits 7, 6 - Cylinder number (high 2 bits of 10-bit  
cylinder number, 0-based, not value checked)**

**- Bits 5 to 0 - Sector number (not value checked)**

**(DH) - Head number (0-based, not value checked)**

**(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)**

**(ES:BX) - Address of buffer**

**On Return:**

**(AH) - Status of operation (see values for the status of  
operation on page 2-IN13F-2)**

**CF = 0 - Status is 0**

**= 1 - Status is non 0**

**Address hex 40:74 is set to the status of operation.**

**This function is exactly the same as the Read Desired Sectors  
into Memory function ((AH)=02H) except for the way in which  
data transfers are performed. The unit of data transfer (block  
size) must be specified through the Set Multiple Mode function  
((AH)=24H) before this function is called.**

## **| (AH) = 22H—Write Multiple Sectors from Memory**

**| For Personal System/2 Model 35 SX, Model 35 LS, Model 40 SX,  
| and Model L40 SX:**

- (AL) - Number of sectors
- (CH) - Cylinder number (low 8 bits of 10-bit cylinder number,  
0-based, not value checked)
- (CL) - Bits 7, 6 - Cylinder number (high 2 bits of 10-bit  
cylinder number, 0-based, not value checked)  
- Bits 5 to 0 - Sector number (not value checked)
- (DH) - Head number (0-based, not value checked)
- (DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)
- (ES:BX) - Address of buffer

**| On Return:**

(AH) - Status of operation (see values for the status of  
operation on page 2-IN13F-2)

CF = 0 - Status is 0

= 1 - Status is non 0

**| Address hex 40:74 is set to the status of operation.**

**| This function is exactly the same as the Write Desired Sectors  
| from Memory function ((AH) = 03H) except for the way in which  
| data transfers are performed. The unit of data transfer (block  
| size) must be specified through the Set Multiple Mode function  
| ((AH) = 24H) before this function is called.**

## **(AH) = 23H—Reserved**

## **| (AH) = 24H—Set Multiple Mode**

**| For Personal System/2 Model 35 SX, Model 35 LS, Model 40 SX,  
| and Model L40 SX, this function specifies the number of sectors  
| that are transferred by the Read Multiple Sectors into Memory  
| function ((AH) = 21H) and the Write Multiple Sectors from Memory  
| function ((AH) = 22H).**

**| If the specified number of sectors is 0, the multiple-transfer mode  
| is disabled. If an error is detected during execution of the Set  
| Multiple Mode function, the number of sectors is set to 0, which  
| causes the multiple-transfer mode to be disabled until the Set  
| Multiple Mode function is called again.**

(AL) - Block size (the number of sectors per interrupt that are transferred by multiple read or write operations)  
(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

**On Return:**

(AH) - Status of operation (see values for the status of operation on page 2-IN13F-2)

CF = 0 - Status is 0

= 1 - Status is non 0

Address hex 40:74 is set to the status of operation.

The maximum value for the block size depends on the fixed disk drive type. The value is stored in byte hex 15 of the fixed disk drive parameter table that is created by POST.

**(AH) = 25H—Identify Drive**

For Personal System/2 Model 35 SX, Model 35 LS, Model 40 SX, and Model L40 SX:

(DL) - Drive number; bit 7 = 1 for fixed disk drive (0-based)

(ES:BX) - Address of buffer

**On Return:**

(AH) - Status of operation (see values for the status of operation on page 2-IN13F-2)

CF = 0 - Status is 0

= 1 - Status is non 0

Address hex 40:74 is set to the status of operation.

The buffer stores 512 bytes of characteristic information for the fixed disk drive:

Word	Definition
0	General configuration
1	Number of logical cylinder
2	Reserved
3	Number of logical cylinder
4	Number of unformatted bytes per logical track
5	Number of unformatted bytes per sector
6	Number of logical sectors per track
7	Number of bytes in the inter-sector gaps
8	Number of bytes in the Sync field
9	Number of bytes of vendor-unique status
10 to 19	Serial number (20 ASCII characters)
20	Controller type
21	Controller-buffer size (in 512-byte increments)
22	Number of ECC bytes passed on read/write long
23 to 26	Controller-firmware revision (8 ASCII characters)
27 to 46	Model number (40 ASCII characters)
47	Number of sectors for multiple read/write
48	Doubleword I/O capacity
49	Programmable reallocation capacity
50 to 255	Reserved

*Figure 2-17. Fixed Disk Drive Identification Buffer*

**(AH) = 26H to FFH—Reserved**



## Interrupt 14H—Asynchronous Communication

These routines provide RS-232C support. The following is a summary of the RS-232C support functions of Interrupt 14H.

(AH) = 00H	— Initialize the Communication Port
(AH) = 01H	— Send Character
(AH) = 02H	— Receive Character
(AH) = 03H	— Read Status
(AH) = 04H	— Extended Initialization
(AH) = 05H	— Extended Communication Port Control
(AH) = 06H to FFH	— Reserved

**Figure 2-18. INT 14H Asynchronous Communication Functions**

**Note:** All reserved input fields must be set to 0.

### (AH) = 00H—Initialize the Communication Port

(AL) - Parameters for initialization

Bits 7 to 5 - Baud rate (values are binary)

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

For Personal System/2 products, for baud rates above 9600, see Interrupt 14H, (AH) = 04H and (AH) = 05H.

Bits 4, 3 - Parity (values are binary)

- = 00 - None
- = 01 - Odd
- = 10 - None
- = 11 - Even

Bit 2 - Stop bit

- = 0 - 1
- = 1 - 2

**Bits 1, 0 - Data length (values are binary)**

**= 00 - Reserved**

**= 01 - Reserved**

**= 10 - 7 bits**

**= 11 - 8 bits**

**(DX) - RS-232C communication line (0, 1, 2, or 3) to be used, corresponding to the actual port base addresses at address hex 40:00**

**On Return:**

**(AH) - Line status**

**Bit 7 = 1 - Time-out**

**Bit 6 = 1 - Transmitter shift register empty**

**Bit 5 = 1 - Transmitter holding register empty**

**Bit 4 = 1 - Break detection**

**Bit 3 = 1 - Framing error**

**Bit 2 = 1 - Parity error**

**Bit 1 = 1 - Overrun error**

**Bit 0 = 1 - Data ready**

**(AL) - Modem status**

**Bit 7 = 1 - Receive line signal detection**

**Bit 6 = 1 - Ring indicator**

**Bit 5 = 1 - Data set ready**

**Bit 4 = 1 - Clear to send**

**Bit 3 = 1 - Delta receive line signal detection**

**Bit 2 = 1 - Trailing-edge ring detector**

**Bit 1 = 1 - Delta data set ready**

**Bit 0 = 1 - Delta clear to send**

**Note:** If bit 7 of the line-status byte is set to 1, settings of other bits are unpredictable.

### **(AH) = 01H—Send Character**

**(AL) - Character to be sent**

**(DX) - RS-232C communication line (0, 1, 2, or 3) to be used, corresponding to actual port base addresses at address hex 40:00**

**On Return:**

**(AH) - Line status (see values for the line status on page 2-IN14-2)**

**(AL) is preserved**

## **(AH) = 02H—Receive Character**

(DX) - RS-232C communication line (0, 1, 2, or 3) to be used, corresponding to actual port base addresses at address hex 40:00

On Return:

(AH) - Line status (see values for the line status on page 2-IN14-2)

(AL) - Character that was received

### **Notes:**

1. The routine waits for the character.
2. If bit 7 of the line status byte is set to 1, settings of other bits are unpredictable.

## **(AH) = 03H—Read Status**

(DX) - RS-232C communication line (0, 1, 2, or 3) to be used, corresponding to actual port base addresses at address hex 40:00

On Return:

(AH) - Line status (see values for the line status on page 2-IN14-2)

(AL) - Modem status (see values for the modem status on page 2-IN14-2)

## **(AH) = 04H—Extended Initialization**

For Personal System/2 products:

(AL) - Break

= 00H - No break

= 01H - Break

(BH) - Parity

= 00H - None

= 01H - Odd

= 02H - Even

= 03H - Stick parity odd

= 04H - Stick parity even

(BL) - Stop bit

= 00H - 1

= 01H - If the data-bit length is 6, 7, or 8 bits, the stop-bit length is 2 bits.

If the data-bit length is 5 bits, the stop-bit length is 1½ bits.

(CH) - Data-bit length

- = 00H - 5 bits
- = 01H - 6 bits
- = 02H - 7 bits
- = 03H - 8 bits

(CL) - Baud rate

- = 00H - 110 baud
- = 01H - 150 baud
- = 02H - 300 baud
- = 03H - 600 baud
- = 04H - 1200 baud
- = 05H - 2400 baud
- = 06H - 4800 baud
- = 07H - 9600 baud
- = 08H - 19200 baud
- = 09H - 31250 baud

(DX) - RS-232C communication line (0, 1, 2, or 3) to be used, corresponding to actual port base addresses at address hex 40:00

On Return:

(AH) - Line status (see values for the line status on page 2-IN14-2)

(AL) - Modem status (see values for the modem status on page 2-IN14-2)

For all others, no action is performed.

## **(AH) = 05H—Extended Communication Port Control**

For Personal System/2 products:

(AL) = 00H - Read modem-control register

(DX) - RS-232C communication line (0, 1, 2, or 3) to be used, corresponding to actual port base addresses at address hex 40:00

On Return:

(BL) - Modem-control register

Bits 7 to 5 - Reserved

Bit 4 = 1 - Loop

Bit 3 = 1 - Out2

Bit 2 = 1 - Out1

Bit 1 = 1 - Request to send

Bit 0 = 1 - Data terminal ready

(AL) = 01H - Write modem-control register  
(BL) - Modem-control register  
Bits 7 to 5 - Reserved  
Bit 4 = 1 - Loop  
Bit 3 = 1 - Out2  
Bit 2 = 1 - Out1  
Bit 1 = 1 - Request to send  
Bit 0 = 1 - Data terminal ready  
(DX) - RS-232C communication line (0, 1, 2, or 3) to be used, corresponding to actual port base addresses at address hex 40:00

On Return:

(AH) - Line status (see values for the line status on page 2-IN14-2)  
(AL) - Modem status (see values for the modem status on page 2-IN14-2)

For all others, no action is performed.

**(AH) = 06H to FFH—Reserved**

### **Programming Consideration**

| If the communication line that is specified in (DX) does not have a  
| corresponding port base address at hex 40:00, the function is not  
| executed, and it does not return an error to the caller.

## Notes:

## Interrupt 15H—System Services

The following is a summary of the system services of Interrupt 15H.

(AH) = 00H – Turn Cassette Motor On  
(AH) = 01H – Turn Cassette Motor Off  
(AH) = 02H – Read Blocks from Cassette  
(AH) = 03H – Write Blocks to Cassette  
(AH) = 04H – Build System Parameters Table  
(AH) = 05H – Build Initialization Table  
(AH) = 06H to 0EH – Reserved  
(AH) = 0FH – Format-Unit Periodic Interrupt  
(AH) = 10H to 20H – Reserved  
(AH) = 21H – Power-On Self-Test Error Log  
(AH) = 22H – ROM BASIC Support  
(AH) = 23H – Reserved  
(AH) = 24H – A20 Gate Support  
(AH) = 25H to 3FH – Reserved  
(AH) = 40H – Read/Modify Profiles  
(AH) = 41H – Wait for External Event  
(AH) = 42H – Request System Power-Off  
(AH) = 43H – Read System Status  
(AH) = 44H – Activate/Deactivate Internal Modem Power  
(AH) = 45H to 4EH – Reserved  
(AH) = 4FH – Keyboard Intercept  
(AH) = 50H – Reserved  
(AH) = 51H – Expansion-Unit Information  
(AH) = 52H to 7FH – Reserved  
(AH) = 80H – Device Open  
(AH) = 81H – Device Close  
(AH) = 82H – Program Termination  
(AH) = 83H – Event Wait  
(AH) = 84H – Joystick Support  
(AH) = 85H – System Request Key Pressed  
(AH) = 86H – Wait  
(AH) = 87H – Move Block  
(AH) = 88H – Extended-Memory Size Determination  
(AH) = 89H – Switch Processor to Protected Mode  
(AH) = 8AH to 8FH – Reserved  
(AH) = 90H – Device Busy  
(AH) = 91H – Interrupt Complete  
(AH) = 92H to BFH – Reserved  
(AH) = C0H – Return System Configuration Parameters  
(AH) = C1H – Return Extended BIOS Data Area Segment Address  
(AH) = C2H – Pointing Device BIOS Interface  
(AH) = C3H – Enable/Disable Watchdog Time-Out  
(AH) = C4H – Programmable Option Select  
(AH) = C5H to C6H – Reserved

Figure 2-19 (Part 1 of 2). INT 15H System Services Functions

(AH) = C7H	Return Memory-Map Information
(AH) = C8H	Enable/Disable Processor Functions
(AH) = C9H	Processor Type and Stepping Level
(AH) = CAH to CDH	Reserved
(AH) = CEH	Allocate Arbitration Level
(AH) = CFH	Deallocate Arbitration Level
(AH) = D0H	Reserved
(AH) = D1H	Return Device Descriptor Table (DDT) Information
(AH) = D2H to D3H	Reserved
(AH) = D4H	Return Physical Fixed Disk Drive Number (Selectable Boot)
(AH) = D5H	Reserved
(AH) = D6H	Return Boot Device ID and Key
(AH) = D7H to FFH	Reserved

*Figure 2-19 (Part 2 of 2). INT 15H System Services Functions*

**Note:** All reserved input fields must be set to 0.

**(AH) = 00H—Turn Cassette Motor On**

For PCjr and PC:

On Return:  
 (AH) = 00H  
 CF = 0

For all others, this function is not supported:

On Return:  
 (AH) = 86H  
 CF = 1

**(AH) = 01H—Turn Cassette Motor Off**

For PCjr and PC:

On Return:  
 (AH) = 00H  
 CF = 0

For all others, this function is not supported:

On Return:  
 (AH) = 86H  
 CF = 1



## **(AH) = 02H—Read Blocks from Cassette**

For PCjr and PC:

(CX) - Count of bytes to be read  
(ES:BX) - Pointer to data buffer

On Return:

(DX) - Count of bytes read  
(ES:BX) - Pointer to last byte that was read + 1  
CF = 0 - Operation successfully completed  
= 1 - Operation failed  
For PCjr, when CF = 1:  
(AH) = 01H - CRC error  
= 02H - Lost data transitions  
= 04H - No data found

For all others, this function is not supported:

On Return:  
(AH) = 86H  
CF = 1

## **(AH) = 03H—Write Blocks to Cassette**

For PCjr and PC:

(CX) - Count of bytes to be written  
(ES:BX) - Pointer to data buffer

On Return:

(CX) = 00H  
(ES:BX) - Pointer to last byte that was written + 1  
CF = 0 - Operation successfully completed  
= 1 - Operation failed  
For PCjr, when CF = 1:  
(AH) = 01H - CRC error  
= 02H - Lost data transitions  
= 04H - No data found

For all others, this function is not supported:

On Return:  
(AH) = 86H  
CF = 1

**| (AH) = 04H—Build System Parameters Table**

**| See “Build System Parameters Table—Operating System” and “Build System Parameters Table—BIOS” in the Advanced BIOS “Initialization” section.**

**| (AH) = 05H—Build Initialization Table**

**| See “Build Initialization Table—Operating System” and “Build Initialization Table—BIOS” in the Advanced BIOS “Initialization” section.**

**(AH) = 0FH—Format-Unit Periodic Interrupt**

For any device that uses ESDI-type commands:

(AL) - Phase code  
= 00H - Reserved  
= 01H - Surface analysis  
= 02H - Formatting

On Return:

CF = 0 - Continue formatting or scanning  
= 1 - End formatting or scanning

**Note:** This function provides a hook to the caller when the formatting or scanning of each cylinder is complete. If no handler is hooked, CF is set to 1 on return.

For PCjr and PC, this function is not supported:

On Return:

(AH) = 80H  
CF = 1

For all others, this function is not supported:

On Return:

(AH) = 86H  
CF = 1

**(AH) = 10H to 20H—Reserved**

## **(AH) = 21H—Power-On Self-Test Error Log**

| This function reads from and writes to the power-on self-test (POST) error log area, which is used to communicate POST errors to the operating system.

For PCjr and PC, this function is not supported:

On Return:  
(AH) = 80H  
CF = 1

For Personal System/2 products except Model 25 and Model 30:

(AL) = 00H - Read POST error log

On Return:  
(BX) - Number of POST error codes that were stored  
(ES:DI) - Pointer to POST error log  
(AH) = 00H  
CF = 0

(AL) = 01H - Write error code to POST error log  
(BX) - POST error code (word)  
(BH) - Device code  
(BL) - Device error

On Return:  
(AH) = 00H - Successfully stored  
      = 01H - Error code location full  
CF = 0 - Successfully stored  
      = 1 - Error code location full

For all others, this function is not supported:

On Return:  
(AH) = 86H  
CF = 1

## **| (AH) = 22H—ROM BASIC Support**

On Return:  
(ES:BX) - Pointer to ROM BASIC  
CF = 0 - Operation successfully completed  
      = 1 - Request failed  
(AH) - Return code  
      = 00H - Operation successfully completed  
      = 86H - Function not supported (ROM BASIC is at  
              address hex F600:0000)

**(AH) = 23H—Reserved**

**(AH) = 24H—A20 Gate Support**

**(AL) = 00H - Disable A20 gate**

**On Return:**

**(AH) - Return code**

- = 00H - Operation successfully completed
- = 01H - Keyboard controller is in secure mode
- = 86H - Function not supported

**CF = 0 - Operation successfully completed**

= 1 - Operation failed

**(AL) = 01H - Enable A20 gate**

**On Return:**

**(AH) - Return code**

- = 00H - Operation successfully completed
- = 01H - Keyboard controller is in secure mode
- = 86H - Function not supported

**CF = 0 - Operation successfully completed**

= 1 - Operation failed

**(AL) = 02H - Query status of A20 gate**

**On Return:**

**(AL) - Status of A20 gate**

- = 00H - Disabled
- = 01H - Enabled

**(AH) - Return code**

- = 00H - Operation successfully completed
- = 01H - Keyboard controller is in secure mode
- = 86H - Function not supported

**CF = 0 - Operation successfully completed**

= 1 - Operation failed

**(AL) = 03H - Query A20 gate support**

**On Return:**

**(BX) - Status of A20 gate support**

**Bit 15 = 0 - No additional data available**

- = 1 - All bits in register are used;  
additional data is available (location  
of additional data is to be determined)

**Bit 1 = 0 - Not supported with bit 1 of port hex 92**

- = 1 - Supported with bit 1 of port hex 92

**Bit 0 = 0 - Not supported on keyboard controller**

- = 1 - Supported on keyboard controller

**(AH) - Return code**

- = 00H - Operation successfully completed
- = 01H - Keyboard controller is in secure mode
- = 86H - Function not supported

**CF = 0 - Operation successfully completed**

= 1 - Operation failed

**(AH) = 25H to 3FH—Reserved**

**(AH) = 40H—Read/Modify Profiles**

For PCjr and PC, this function is not supported:

On Return:  
(AH) = 80H  
CF = 1

For PC Convertible:

(AL) = 00H - Read system profile

On Return:  
(CX,BX) - Profile information

(AL) = 01H - Modify system profile  
(CX,BX) - Profile information

(AL) = 02H - Read internal modem profile

On Return:  
(BX) - Profile information

(AL) = 03H - Modify internal modem profile  
(BX) - Profile information

On Return for all:  
(AL) = 00H - Operation successfully completed  
      = 80H - Profile execution failed  
CF = 0 - Operation successfully completed  
      = 1 - Operation failed

For all others, this function is not supported:

On Return:  
(AH) = 86H  
CF = 1

**(AH) = 41H—Wait for External Event**

For PCjr and PC, this function is not supported:

On Return:  
(AH) = 80H  
CF = 1

For products that support this function:

- (AL) - Event-type code
  - = 00H - Return after any event has occurred
  - = 01H - Compare values, return if equal
  - = 02H - Compare values, return if not equal
  - = 03H - Test bit, return if not 0
  - = 04H - Test bit, return if 0
- (BH) - Condition compare or mask value
- (BL) - Time-out value (in 55-millisecond units)
  - = 0 - No time-out
- (ES:DI) - Pointer to byte in user area for event determination (event-type codes 01H to 04H)
  - or —
  - (DX) contains the I/O port address to be read for event determination (event-type codes 11H to 14H)

On Return:  
CF = 1 - Time-out

**Notes:**

1. Event type codes (AL) = 11H, 12H, 13H, and 14H are the same as codes (AL) = 01H, 02H, 03H, and 04H, respectively, except that (DX) is used to contain the event determination address.
2. To determine which products support this function, see Interrupt 15H, Return System Configuration Parameters function ((AH) = C0H), feature information byte 1, bit 3.

For all others, this function is not supported:

On Return:  
(AH) = 86H  
CF = 1

**(AH) = 42H—Request System Power-Off**

For PCjr and PC, this function is not supported:

On Return:  
(AH) = 80H  
CF = 1

**For PC Convertible:**

(AL) = 00H - Use system profile for suspend/IPL determination  
(AL) = 01H - Force system suspend mode, regardless of profile

On Return:  
(AX) is modified

**For all others, this function is not supported:**

On Return:  
(AH) = 86H  
CF = 1

**(AH) = 43H—Read System Status**

**For PCjr and PC:**

On Return:  
CF = 1  
(AH) = 80H

**For PC Convertible:**

On Return:  
(AH) is modified  
(AL) - Status  
    Bit 7 - Low battery indication  
    Bit 6 - Operating on external power source  
    Bit 5 - Standby power lost (real-time clock time bad)  
    Bit 4 - Power activated by real-time clock alarm  
    Bit 3 - Internal modem power-on  
    Bit 2 - RS-232C/parallel power-on  
    Bit 1 - Reserved  
    Bit 0 - LCD detached

**For all others, this function is not supported:**

On Return:  
(AH) = 86H  
CF = 1

## **(AH) = 44H—Activate/Deactivate Internal Modem Power**

For PCjr and PC, this function is not supported:

On Return:  
(AH) = 80H  
CF = 1

For PC Convertible:

(AL) = 00H - Power-off internal modem  
(AL) = 01H - Power-on internal modem and configure according  
to system profile

On Return:  
(AL) = 00H - Operation successfully completed  
= 80H - Operation failed  
CF = 0 - Operation successfully completed  
= 1 - Operation failed

For all others, this function is not supported:

On Return:  
(AH) = 86H  
CF = 1

## **(AH) = 45H to 4EH—Reserved**

## **(AH) = 4FH—Keyboard Intercept**

For PCjr and PC, this function is not supported:

On Return:  
(AH) = 80H  
CF = 1

For PC/XT BIOS dated 11/8/82 and AT BIOS dated 1/10/84, this  
function is not supported:

On Return:  
(AH) = 86H  
CF = 1



For all others, Interrupt 09H (Keyboard) calls the keyboard intercept (keyboard escape) to allow the keystroke to be changed or absorbed. Normally, the system returns the scan code unchanged, but the operating system can point Interrupt 15H to itself and do one of the following:

1. Replace (AL) with a different scan code and return the carry flag set to 1, effectively changing the keystroke.
2. Process the keystroke and return with the carry flag reset, causing Interrupt 09H to ignore the keystroke.

(AL) - Scan code  
CF = 1

On Return:  
CF = 1  
(AL) - New scan code  
— or —  
CF = 0  
(AL) - Unchanged scan code

**Note:** To determine which products support this function, see Interrupt 15H, Return System Configuration Parameters function ((AH) = C0H), feature information byte 1, bit 4.

**(AH) = 50H to 7FH—Reserved**

## **| (AH) = 51H—Expansion-Unit Information**

| (AL) = 01H - Return configuration number

On Return:

(AL) - Current configuration number

= 00H - System unit only

= FFH - Configuration not recognized

(BX) - Status flag

Bit 15 = 0 - No additional configuration information is available

= 1 - Additional configuration information is available

(the access method is to be determined)

Bits 14 to 0 - Reserved

(AH) - Return code

= 00H - Operation successfully completed

= 01H - Expansion unit is not present

= 86H - Function not supported

CF = 0 - Operation failed

= 1 - Operation successfully completed

## **(AH) = 80H—Device Open**

For PCjr and PC, this function is not supported:

On Return:

(AH) = 80H

CF = 1

For PC/XT BIOS dated 11/8/82, this function is not supported:

On Return:

(AH) = 86H

CF = 1

For all others:

(BX) - Device ID

(CX) - Process ID

## **(AH) = 81H—Device Close**

For PCjr and PC, this function is not supported:

On Return:

(AH) = 80H

CF = 1

For PC/XT BIOS dated 11/8/82, this function is not supported:

On Return:  
(AH) = 86H  
CF = 1

For all others:

(BX) - Device ID  
(CX) - Process ID

### **(AH) = 82H—Program Termination**

For PCjr and PC, this function is not supported:

On Return:  
(AH) = 80H  
CF = 1

For PC/XT BIOS dated 11/8/82, this function is not supported:

On Return:  
(AH) = 86H  
CF = 1

For all others:

(BX) - Device ID

### **(AH) = 83H—Event Wait**

For PCjr and PC, this function is not supported:

On Return:  
(AH) = 80H  
CF = 1

For PC/XT, this function is not supported:

On Return:  
(AH) = 86H  
CF = 1

**For AT BIOS dated 1/10/84:**

(CX,DX) - Microseconds until posting  
(granularity is 976 microseconds)  
(ES:BX) - Pointer to byte in caller's memory that has the  
high-order bit set by BIOS as soon as possible  
after the interval expires

**On Return:**

CF = 0 - Operation successfully completed  
= 1 - Operation failed; function busy

**For all others:**

(AL) = 00H - Set interval  
(ES:BX) - Pointer to byte in caller's memory that has the  
high order bit set by BIOS as soon as possible  
after the interval expires  
(CX,DX) - Microseconds until posting  
(granularity is 976 microseconds)

**On Return:**

CF = 0 - Operation successfully completed  
= 1 - Operation failed, function busy

(AL) = 01H - Cancel set interval

**On Return:**

CF = 0 - Operation successfully completed

(Personal System/2 Model 25 and Model 30 always return with CF = 1.)

### **(AH) = 84H—Joystick Support**

**For PCjr, PC, and PC Convertible, this function is not supported:**

**On Return:**

(AH) = 80H  
CF = 1

**For PC/XT BIOS dated 11/8/82, this function is not supported:**

**On Return:**

(AH) = 88H  
CF = 1

**For all others:**

**(DX) = 00H - Read current switch settings**

**On Return:**

**(AL) - Switch settings (bits 7 to 4)**

**CF = 1 - Invalid call**

**(DX) = 01H - Read resistive inputs**

**On Return:**

**(AX) - Joystick A, x value**

**(BX) - Joystick A, y value**

**(CX) - Joystick B, x value**

**(DX) - Joystick B, y value**

**CF = 1 - Invalid call**

### **(AH) = 85H—System Request Key Pressed**

**For PCjr and PC, this function is not supported:**

**On Return:**

**(AH) = 80H**

**CF = 1**

**For PC/XT BIOS dated 11/8/82, this function is not supported:**

**On Return:**

**(AH) = 86H**

**CF = 1**

**For all others:**

**(AL) = 00H - Key make**

**(AL) = 01H - Key break**

### **(AH) = 86H—Wait**

**For PCjr and PC, this function is not supported:**

**On Return:**

**(AH) = 80H**

**CF = 1**

For PC/XT, this function is not supported:

On Return:  
(AH) = 86H  
CF = 1

For all others:

(CX,DX) - Time before return to caller, in microseconds  
(granularity is 976 microseconds)

On Return:  
CF = 0 - Successful wait  
= 1 - Wait function already in progress

### **(AH) = 87H—Move Block**

For PCjr and PC, this function is not supported:

On Return:  
(AH) = 80H  
CF = 1

For PC/XT, PC Convertible, and Personal System/2 Model 25 and Model 30, this function is not supported:

On Return:  
(AH) = 86H  
CF = 1

For AT, PC/XT Model 286, and Personal System/2 products except Model 25 and Model 30, this function allows a real-mode program or system to transfer a block of data to and from storage above the 1MB protected-mode address range by switching to the protected mode.

(CX) - Word count of storage block to be moved  
(maximum count = 8000H for 32KB words [65KB])  
(ES:SI) - Location of a global descriptor table (GDT) that was built by a routine that is using this function

(ES:SI) points to a global descriptor table (GDT) that was built before this function was called. The descriptors are used to perform the block move in the protected mode. The source and target descriptors that are built by the user must have a segment length equal to or greater than  $2 \times ((CX) - 1)$ . The data-access rights byte must be set to current privilege level 0 with read/write

access (hex 93). The 24-bit address (byte high, word low) must be set to the target or source.

**Note:** No interrupts are allowed during transfers. Large block moves might cause lost interrupts.

On Return:

(AH) = 00H - Operation successfully completed

(AH) = 01H - RAM parity (parity error registers cleared)

(AH) = 02H - Other exception interrupt error occurred

(AH) = 03H - Gate address line 20H failed; all registers except (AH) are restored

If (AH) = 00H:

CF = 0

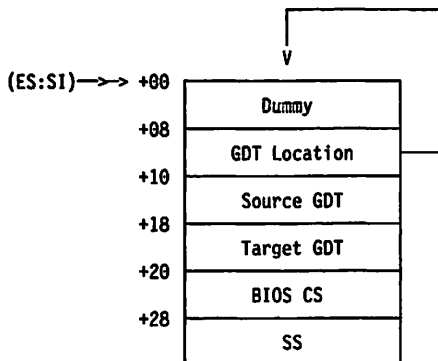
ZF = 1

If (AH) = 01H to 03H:

CF = 1

ZF = 0

The following figure shows the organization of a block-move global descriptor table (GDT).



**Figure 2-20. Block Move Global Descriptor Table**

The following is the format of the global descriptor table (the actual location that is pointed to by (ES:SI)):

BLOCKMOVE_GDT_DEF	STRUC		
	DW	0,0,0,0	; First descriptor not accessible
CGDT_LOC	DW	?,?,?,0	; Location of calling routine GDT
SOURCE	DW	?,?,?,0	; Source descriptor
TARGET	DW	?,?,?,0	; Target descriptor
BIOS_CS	DW	?,?,?,0	; BIOS code descriptor
TEMP_SS	DW	?,?,?,0	; Stack descriptor
BLOCKMOVE_GDT_DEF	ENDS		

**Figure 2-21. Global Descriptor Table Format**

The descriptors are defined as follows:

- The first descriptor is the required dummy and is user-initialized to 0.
- The second descriptor points to the GDT as a data segment. It is user-initialized to 0 and can be modified by BIOS.
- The third descriptor points to the source to be moved and is user-initialized (see Figure 2-22 on page 2-IN15-19).
- The fourth descriptor points to the destination segment and is user-initialized (see Figure 2-22 on page 2-IN15-19).
- The fifth descriptor is used by BIOS to create the protected-mode code segment. It is user-initialized to 0 and can be modified by BIOS.
- The sixth descriptor is used by BIOS to create a protected-mode stack segment. It is user-initialized to 0, can be modified by BIOS, and points to the user stack.



The following is an example of a source or target descriptor:

```

SOURCE_TARGET_DEF  STRUC
SEG_LIMIT           DW      ?           ; Segment limit (1 to 65536 bytes)
LO_WORD             DW      ?           ; 24-bit segment physical
HI_BYTE             DB      ?           ; address (0 to [16MB-1])
DATA_ACC_RIGHTS     DB      93H        ; Access rights byte (CPL 0 - R/W)
Reserved            DW      0           ; Reserved word (must be 0)
SOURCE_TARGET_DEF  ENDS

```

*Figure 2-22. Source or Target Descriptor Example*

### **(AH) = 88H—Extended-Memory Size Determination**

For PCjr and PC, this function is not supported:

On Return:  
 (AH) = 80H  
 CF = 1

For PC/XT, PC Convertible, and Personal System/2 Model 25 and Model 30, this function is not supported:

On Return:  
 (AH) = 86H  
 CF = 1

For 80286-, 80386-, and 80486-processor systems, this function returns the amount of system memory that is between addresses hex 100000 and hex FFFFFFFF, as determined by POST.

On Return:  
 (AX) - Contiguous 1KB blocks of available memory between addresses hex 100000 and hex FFFFFFFF (1MB to 16MB)

**Note:** The Return Memory-Map Information function ((AH)=C7H), if it is supported, returns the complete memory information.

### **(AH) = 89H—Switch Processor to Protected Mode**

For PCjr and PC, this function is not supported:

On Return:  
 (AH) = 80H  
 CF = 1

**For PC/XT, PC Convertible, and Personal System/2 Model 25 and Model 30, this function is not supported:**

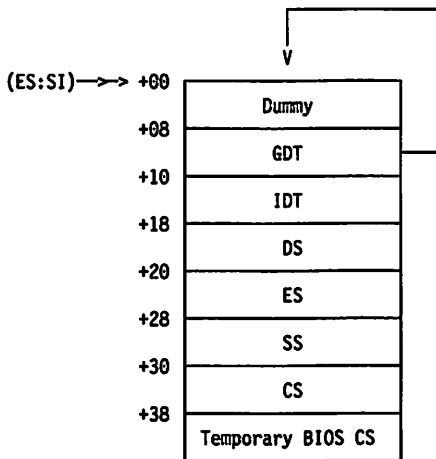
**On Return:**  
**(AH) = 86H**  
**CF = 1**

**For AT, PC/XT Model 286, and Personal System/2 products except Model 25 and Model 30, this function allows the user to switch the system microprocessor into the protected (virtual address) mode. When this function is completed, the system microprocessor is in the protected mode, and control is transferred to the code segment that is specified by the user.**

**The entry requirements are as follows:**

- **(ES:SI) points to a global descriptor table (GDT) that was built before this function was called. These descriptors initialize the interrupt descriptor table (IDT) register, the GDT register, and the stack segment (SS) selector. The data segment (DS) selector, the extra segment (ES) selector, and the code segment (CS) selector are initialized from descriptors that were built by the routine that is using this function.**
- **(BH) contains an index into the interrupt descriptor table that indicates where the first eight hardware interrupts begin (interrupt level 1). (BL) contains an index into the interrupt descriptor table that indicates where the second eight hardware interrupts begin (interrupt level 2).**

The following figure shows the organization of the selectors in this GDT; the actual location is pointed to by (ES:SI).



**Figure 2-23. Global Descriptor Table**

Each descriptor must contain the limit, the base address, and the access-rights byte. The descriptors are defined as follows:

- The first descriptor is the required dummy and is user-initialized to 0.
- The second descriptor points to the GDT as a data segment and is user-initialized.
- The third descriptor points to the user-defined interrupt descriptor table (IDT) and is user-initialized.
- The fourth descriptor points to the user data segment (DS) and is user-initialized.
- The fifth descriptor points to the user extra segment (ES) and is user-initialized.
- The sixth descriptor points to the user stack segment (SS) and is user-initialized.
- The seventh descriptor points to the user code segment (CS) that this function returns to. It is user-initialized.
- The eighth descriptor is used to establish a code segment for itself. This is necessary for this function to complete its operation while the system microprocessor is in the

protected mode. When control is passed to the user code, this descriptor can be reused.

(AH) = 89H

(ES:SI) - Location of GDT built by a routine that is using this function

On Return:

(AH) = 00H - Operation successfully completed

All segment registers are changed; (AX) and (BP) are modified.

#### Considerations:

- BIOS functions are not available to the user. The user must handle all I/O commands.
- Interrupt vector locations must be moved, because of the 80286 reserved areas.
- The hardware interrupt controllers must be reinitialized to define locations that do not reside in the 80286 reserved areas.
- An exception interrupt table and handler must be initialized by the user.
- The interrupt descriptor table cannot overlap the real-mode BIOS interrupt descriptor table.

The following is an example of a way to switch the system microprocessor to the protected (virtual address) mode:

- User code -

```
MOV     AX, gdt segment
MOV     ES, AX
MOV     SI, gdt offset
MOV     BH, hardware interrupt level 1 offset into IDT
MOV     BL, hardware interrupt level 2 offset into IDT
MOV     AH, 89H
INT     15H
```

- User code -

(Protected mode established)

The following is the format of the global descriptor table (the actual location that is pointed to by (ES:SI)).

```

VIRTUAL_ENABLE_GDT_DEF STRUC
    DW 0,0,0,0      ; First descriptor not accessible
GDTPTR_DW DW ?,?,?,0 ; GDT descriptor
IDTPTR DW ?,?,?,0 ; IDT descriptor
USER_DS DW ?,?,?,0 ; User data segment descriptor
USER_ES DW ?,?,?,0 ; User extra segment descriptor
USER_SS DW ?,?,?,0 ; User stack segment descriptor
USER_CS DW ?,?,?,0 ; User code segment descriptor
BIO_CS DW ?,?,?,0 ; Temporary BIOS descriptor
VIRTUAL_ENABLE_GDT_DEF ENDS

```

*Figure 2-24. Global Descriptor Table Format*

**(AH) = 8AH to 8FH—Reserved**

**(AH) = 90H—Device Busy**

For PCjr and PC, this function is not supported:

On Return:  
 (AH) = 80H  
 CF = 1

For PC/XT BIOS dated 11/8/82, this function is not supported:

On Return:  
 (AH) = 86H  
 CF = 1

For all others, this function is called to tell the operating system that the system is about to wait for a device.

The type-code assignments for (AH) = 90H and (AH) = 91H use the following general guidelines:

- Type codes hex 00 to hex 7F are for serially-reusable devices (the operating system must serialize access).
- Type codes hex 80 to hex BF are for reentrant devices. (ES:BX) is used to distinguish between different calls (multiple I/O calls are allowed simultaneously).
- Type codes hex C0 to hex FF are for wait-only calls. There is no complementary posting for these waits. These are time-out only. Times are function-number dependent.

**(AL) - Type code**

- = 00H - Disk (time-out)
- = 01H - Diskette (time-out)
- = 02H - Keyboard (no time-out)
- = 03H - Pointing device (time-out)
- = 80H - Network (no time-out)  
(ES:BX) - Network control block (NCB)
- = FCH - Fixed disk reset for Personal System/2 products only  
(time-out)
- = FDH - Diskette drive motor start (time-out)
- = FEH - Printer (time-out)

**On Return:**

- CF = 0 - Wait not satisfied
- = 1 - Minimum wait time satisfied for  
this type code

**(AH) = 91H—Interrupt Complete**

For PCjr and PC, this function is not supported:

**On Return:**

- (AH) = 80H
- CF = 1

For PC/XT BIOS dated 11/8/82, this function is not supported:

**On Return:**

- (AH) = 86H
- CF = 1

For all others, the interrupt-complete flag is set to tell the operating system that the interrupt has occurred.

- (AL) - Type code (see the type codes for the Device Busy function  
((AH) = 90H) on page 2-IN15-24)

**(AH) = 92H to BFH—Reserved**

**(AH) = C0H—Return System Configuration Parameters**

To obtain a complete Micro Channel configuration table, the Return Memory-Map Information function ((AH) = C7H), the Processor Type and Stepping Level function ((AH) = C9H), and the Return Device Descriptor Table (DDT) Information function ((AH) = D1H) must be called in addition to the Return System Configuration Parameters function ((AH) = C0H).

**For PCjr and PC, this function is not supported:**

**On Return:**  
**(AH) = 80H**  
**CF = 1**

**For PC/XT BIOS dated 11/8/82 and AT BIOS dated 1/10/84, this function is not supported:**

**On Return:**  
**(AH) = 86H**  
**CF = 1**

**For AT BIOS dated 6/10/85 and later, PC/XT BIOS dated 1/10/86 and later, PC/XT Model 286, PC Convertible, and Personal System/2 products:**

**On Return:**  
**(ES:BX) - Pointer to system configuration parameter table**  
**(AH) = 0**  
**CF = 0**

### **System configuration parameters:**

DW	XXXX	Length of table (in bytes)
DB	XX	Model byte
DB	XX	Submodel byte
DB	XX	BIOS revision level
DB	XX	Feature information byte 1
		Bit 7 = 0 - Fixed-disk BIOS does not use DMA channel 3 or channel-3 usage cannot be determined
		= 1 - Fixed-disk BIOS uses DMA channel 3 (not used in Micro Channel systems)
		Bit 6 = 0 - Second interrupt chip is not present
		= 1 - Second interrupt chip is present
		Bit 5 = 0 - Real-time clock is not present
		= 1 - Real-time clock is present
		Bit 4 = 0 - Keyboard escape sequence (Interrupt 15H, (AH)=4FH) is not called in Keyboard interrupt (09H)
		= 1 - Keyboard escape sequence is called in Keyboard interrupt

DB

XX

- Bit 3 = 0 - Wait for External Event function (Interrupt 15H, (AH) = 41H) is not supported
  - = 1 - Wait for External Event function is supported
- Bit 2 = 0 - Extended BIOS data area is not allocated
  - = 1 - Extended BIOS data area is allocated
- Bit 1 = 0 - PC-type I/O channel is implemented
  - = 1 - Micro Channel bus is implemented
- Bit 0 = 0 - System does not have dual-bus capability
  - = 1 - System has dual-bus capability

Feature information byte 2

Bit 7 - Reserved

- Bit 6 = 0 - Keyboard Functionality Determination function (Interrupt 16H, (AH) = 09H) is not supported
  - = 1 - Keyboard Functionality Determination function is supported

- Bit 5 = 0 - Return POS Data function (Interrupt 15H, (AH) = C8H) is not supported
  - = 1 - Return POS Data function is supported

- Bit 4 = 0 - Return Memory-Map Information function (Interrupt 15H, (AH) = C7H) is not supported
  - = 1 - Return Memory-Map Information function is supported

- Bit 3 = 0 - Enable/Disable Processor Functions function (Interrupt 15H, (AH) = C8H) is not supported
  - = 1 - Enable/Disable Processor Functions function is supported

- Bit 2 = 0 - 8042 keyboard controller is in the system
  - = 1 - Non-8042 keyboard controller is in the system

- Bit 1 = 0 - Data streaming is not supported
  - = 1 - Data streaming is supported

Bit 0 - Reserved

DB

XX

Feature information byte 3

Bits 7 to 4 - Reserved

- Bit 3 = 0 - SCSI subsystem is not supported on the system board
  - = 1 - SCSI subsystem is supported on the system board

- Bit 2 = 0 - Information panel is not installed
  - = 1 - Information panel is installed

- Bit 1 = 0 - Non-IML system
  - = 1 - IML system

- Bit 0 = 0 - No SCSI support in IML
  - = 1 - SCSI support in IML



DB	XX	Feature information byte 4 Bits 7 to 0 - Reserved
DB	XX	Feature information byte 5 Bits 7 to 0 - Reserved

**Note:** For Personal System/2 products except Model 25 and Model 30, if the system model cannot be determined, this function returns (AH)=86H and CF=1, and (ES:BX) is not changed.

### **(AH) = C1H—Return Extended BIOS Data Area Segment Address**

On Return:

(ES) - Extended BIOS data area segment address

CF = 0 - No error

= 1 - Error

For PCjr and PC, this function is not supported:

On Return:

(AH) = 80H

CF = 1

For PC/XT, AT, PC/XT Model 286, and PC Convertible, this function is not supported:

On Return:

(AH) = 86H

CF = 1

### **(AH) = C2H—Pointing-Device BIOS Interface**

(AL) = 00H - Enable/disable pointing device

(BH) = 00H - Disable

= 01H - Enable

On Return:

(AH) - Status

= 00H - No error

= 01H - Invalid function call

= 02H - Invalid input

= 03H - Interface error

= 04H - Resend

= 05H - No far call installed

CF = 0 - Operation successfully completed

= 1 - Operation failed

**(AL) = 01H - Reset pointing device**

**On Return:**

**(BL) - Value returned by the attached device after reset**

**((BL) is set to hex AA if the device is a mouse)**

**(AH) - Status (see return for (AL) = 00H)**

**CF = 0 - Operation successfully completed**

**= 1 - Operation failed**

**If the operation is successfully completed:**

**(BH) - Device ID**

**= 00H**

**The pointing-device state is as follows:**

**- Disabled**

**- Sample rate of 100 reports per second**

**- Resolution of 4 counts per millimeter**

**- Scaling of 1:1**

**- Data package size remains the same as before this function was called**

**(AL) = 02H - Set sample rate**

**(BH) - Sample rate value**

**= 00H - 10 reports per second**

**= 01H - 20 reports per second**

**= 02H - 40 reports per second**

**= 03H - 60 reports per second**

**= 04H - 80 reports per second**

**= 05H - 100 reports per second**

**= 06H - 200 reports per second**

**On Return:**

**(AH) - Status (see return for (AL) = 00H)**

**CF = 0 - Operation successfully completed**

**= 1 - Operation failed**

**(AL) = 03H - Set resolution**

**(BH) - Resolution value**

**= 00H - 1 count per millimeter**

**= 01H - 2 counts per millimeter**

**= 02H - 4 counts per millimeter**

**= 03H - 8 counts per millimeter**

**On Return:**

**(AH) - Status (see return for (AL) = 00H)**

**CF = 0 - Operation successfully completed**

**= 1 - Operation failed**

**(AL) = 04H - Read device type**

**On Return:**

**(AH) - Status (see return for (AL) = 00H)**

**CF = 0 - Operation successfully completed**

**= 1 - Operation failed**

**If the operation is successfully completed:**

**(BH) - Device ID**

**= 00H**

(AL) = 05H - Pointing-device interface initialization

(BH) - Data package size

- = 00H - Reserved
- = 01H - 1 byte
- = 02H - 2 bytes
- = 03H - 3 bytes
- = 04H - 4 bytes
- = 05H - 5 bytes
- = 06H - 6 bytes
- = 07H - 7 bytes
- = 08H - 8 bytes

On Return:

(AH) - Status (see return for (AL)=00H)

CF = 0 - Operation successfully completed

= 1 - Operation failed

The pointing-device state is as follows:

- Disabled
- Sample rate of 100 reports per second
- Resolution of 4 counts per millimeter
- Scaling at 1:1

(AL) = 06H - Extended commands

(BH) = 00H - Return status

On Return:

(AH) - Status (see return for (AL)=00H)

CF = 0 - Operation successfully completed

= 1 - Operation failed

If the operation is successfully completed:

(BL) - Status byte 1

- Bit 7 = 0 - Reserved
- Bit 6 = 0 - Stream mode
- = 1 - Remote mode
- Bit 5 = 0 - Disable
- = 1 - Enable
- Bit 4 = 0 - 1:1 scaling
- = 1 - 2:1 scaling
- Bit 3 = 0 - Reserved
- Bit 2 = 1 - Left button pressed
- Bit 1 = 0 - Reserved
- Bit 0 = 1 - Right button pressed

(CL) - Status byte 2

- = 00H - 1 count per millimeter
- = 01H - 2 counts per millimeter
- = 02H - 4 counts per millimeter
- = 03H - 8 counts per millimeter

**(DL) - Status byte 3**

- = 0AH - 10 reports per second
- = 14H - 20 reports per second
- = 28H - 40 reports per second
- = 3CH - 60 reports per second
- = 50H - 80 reports per second
- = 64H - 100 reports per second
- = C8H - 200 reports per second

**(BH) = 01H - Set scaling to 1:1**

**On Return:**

**(AH) - Status (see return for (AL) = 00H)**

**CF = 0 - Operation successfully completed**

**= 1 - Operation failed**

**(BH) = 02H - Set scaling to 2:1**

**On Return:**

**(AH) - Status (see return for (AL) = 00H)**

**CF = 0 - Operation successfully completed**

**= 1 - Operation failed**

**(AL) = 07H - Device-driver far-call initialization**

**(ES) - Segment**

**(BX) - Offset**

**On Return:**

**(AH) - Status (see return for (AL) = 00H)**

**CF = 0 - Operation successfully completed**

**= 1 - Operation failed**

**For PCjr and PC, this function is not supported:**

**On Return:**

**(AH) = 80H**

**CF = 1**

**For PC/XT, AT, PC/XT Model 286, and PC Convertible, this function is not supported:**

**On Return:**

**(AH) = 86H**

**CF = 1**

The user codes a routine to receive control when the pointing-device data is available. The device-driver far-call initialization communicates the address of this routine to the BIOS. Each time the pointing-device data is available, the pointing-device interrupt handler calls the user routine, with the following parameters on the stack:

Status - First word pushed on the stack  
X data - Second word pushed on the stack  
Y data - Third word pushed on the stack  
Z data - Fourth word pushed on the stack

Word 1 on the stack:

Low byte - Status

Bit 7 - Y data overflow

= 1 - Overflow

Bit 6 - X data overflow

= 1 - Overflow

Bit 5 - Y data sign

= 1 - Negative

Bit 4 - X data sign

= 1 - Negative

Bit 3 - Reserved (must be set to 1)

Bit 2 - Reserved (must be set to 0)

Bit 1 - Right-button status

= 1 - Pressed

Bit 0 - Left-button status

= 1 - Pressed

High byte = 0

Word 2 on the stack:

Low byte - X data

Bit 7 = Most-significant bit

Bit 0 - Least-significant bit

High byte = 0

Word 3 on the stack:

Low byte - Y data

Bit 7 = Most-significant bit

Bit 0 - Least-significant bit

High byte = 0

Word 4 on the stack:

Low byte = 0

High byte = 0

The pointing-device interrupt handler uses a far call to transfer control to the user routine. This routine should be coded as a far procedure and should not pop the parameters off the stack before returning.

## **(AH) = C3H—Enable/Disable Watchdog Time-Out**

For PCjr and PC, this function is not supported:

On Return:  
(AH) = 80H  
CF = 1

For PC/XT, AT, PC/XT Model 286, PC Convertible, and Personal System/2 Model 25 and Model 30, this function is not supported:

On Return:  
(AH) = 86H  
CF = 1

For Personal System/2 products except Model 25 and Model 30:

(AL) = 00H - Disable the PS/2\* watchdog timer  
= 01H - Enable the PS/2 watchdog timer  
= 02H - Disable the Gearbox\* system  
= 03H - Enable the Gearbox system

(BX) - Watchdog timer count (values from 1 to 255 are  
valid for Personal System/2 products)

On Return:  
CF = 0 - Operation successfully completed  
= 1 - Operation failed or function not supported

## **(AH) = C4H—Programmable Option Select (POS)**

For PCjr and PC, this function is not supported:

On Return:  
(AH) = 80H  
CF = 1

---

\* PS/2 and Gearbox are trademarks of the International Business Machines Corporation.

For PC/XT, AT, PC/XT Model 286, PC Convertible, and Personal System/2 Model 25 and Model 30, this function is not supported:

On Return:

(AH) = 86H

CF = 1

For Personal System/2 products except Model 25 and Model 30:

(AL) = 00H - Return base POS-adapter-register address

On Return:

(DX) - Base POS-adapter-register address

CF = 0 - Operation successfully completed

= 1 - Operation failed or function not supported

(AL) = 01H - Enable selected slot for setup cycles

(BL) - Slot number

On Return:

CF = 0 - Operation successfully completed

= 1 - Operation failed or function not supported

(AL) = 02H - Disable setup cycles for all slots

On Return:

CF = 0 - Operation successfully completed

= 1 - Operation failed or function not supported

The following is the procedure for reading the POS registers for all slots in the system:

1. Call the Return Base POS Adapter Register Address function ((AL)=00H). (Examine the return code and carry flag after each BIOS call.)
2. Call the Enable Selected Slot for Setup Cycles function ((AL)=01H) with (BL) set to 1 to put slot 1 into setup mode.
3. Read the POS registers for slot 1, beginning with the base POS adapter register address.
4. Repeat steps 2 and 3 (with the slot number in (BL)) for each slot.
5. Call the Disable Setup Cycles for All Slots function ((AL)=02H) to take all slots out of setup mode.

For additional information about the POS adapter registers, see the "Programmable Option Select" section in the *Personal System/2 Hardware Interface Technical Reference*.

**(AH) = C5H to C6H—Reserved**

## **((AH) = C7H—Return Memory-Map Information**

To obtain a complete Micro Channel configuration table, the Return System Configuration Parameters function ((AH) = C0H), the Processor Type and Stepping Level function ((AH) = C9H), and the Return Device Descriptor Table (DDT) Information function ((AH) = D1H) must be called in addition to the Return Memory-Map Information function ((AH) = C7H).

This function is not supported on all systems. If the function is not supported, the carry flag is set to 1, and the value that is returned in (AH) is either hex 80 (for PCjr and PC) or hex 86.

(DS:SI) - Pointer to the user-supplied memory-map table  
with a minimum length of 66 bytes

On Return:

(AH) = 80H or 86H - Function not supported

CF = 0 - Operation successfully completed

= 1 - Operation failed or function not supported

The word at offset hex 00 is the number of bytes that make up the rest of the table; the minimum value is hex 42 (decimal 66).

Memory values are expressed as the number of 1KB blocks.

Memory-map table structure:

Size	Offset	Description
Word	00H	Number of significant bytes of returned data (excluding this word)
DWord	02H	Amount of local memory between 1MB and 16MB, in 1KB blocks
DWord	06H	Amount of local memory between 16MB and 4GB, in 1KB blocks
DWord	0AH	Amount of system memory between 1MB and 16MB, in 1KB blocks
DWord	0EH	Amount of system memory between 16MB and 4GB, in 1KB blocks
DWord	12H	Amount of cacheable memory between 1MB and 16MB, in 1KB blocks
DWord	16H	Amount of cacheable memory between 16MB and 4GB, in 1KB blocks
DWord	1AH	Number of 1KB blocks before the start of non-system memory between 1MB and 16MB
DWord	1EH	Number of 1KB blocks before the start of non-system memory between 16MB and 4GB
DWord	22H to 2AH	Reserved



The various memory types are defined as follows:

<b>Local memory</b>	Memory on the system board or memory that is not accessible from the channel. It can be system or non-system memory.
<b>Channel memory</b>	Memory on adapters. It can be system or non-system memory.
<b>System memory</b>	Memory that is managed and allocated by the primary operating system. This memory is cached if the cache is enabled.
<b>Non-system memory</b>	Memory that is not managed or allocated by the primary operating system. This memory includes memory-mapped I/O devices; memory that is on an adapter and can be directly modified by the adapter; and memory that can be relocated within its address space, such as bank-switched and expanded-memory-specifications (EMS) memory. This memory is not cached.

The following are constraints on memory configurations for Personal System/2 systems:

- Memory below 1MB is local memory.
- All local memory is system memory and can be cached except for:
  - The extended BIOS data area
  - The video-buffer areas (hex A000 and hex B000)
  - The adapter-ROM areas (hex C000 and hex D000)
  - The BIOS code areas (hex E000 and hex F000).
- Local memory from 1MB to 16MB is configured contiguously from 1MB.
- System memory on adapters from 1MB to 16MB is configured contiguously from 1MB and immediately after any local memory in this range.
- Local memory from 16MB to 4GB is configured contiguously from 16MB.
- System memory on adapters from 16MB to 4GB is configured contiguously from 16MB and immediately after any local memory in this range.

- Local memory that is reassigned from the first 1MB might be configured above channel system memory in either range to make full use of the first 1MB. This memory is treated as an extension of the channel memory in that range.

## | (AH) = C8H—Enable/Disable Processor Functions

| **Note:** In an 80486 processor, any external caches must be disabled when the on-chip cache (L1) is disabled.

### | For Personal System/2 Model 70-A21:

(AL) = 00H - Disable cache  
 = 01H - Enable cache  
 = 02H to FFH - Reserved

On Return:

CF = 0 - Operation successfully completed  
 = 1 - Operation failed

### | For Personal System/2 Model 70/486 and Model 80-A21:

(AL) = 00H - Disable cache  
 = 01H - Enable cache  
 = 02H to FFH - Reserved

On Return:

(AH) - Return code

= 00H - Operation successfully completed  
 = 01H - Function choice in (AL) is invalid  
 = 02H - NVRAM data is invalid  
 = 03H - Cache test error

CF = 0 - Operation successfully completed  
 = 1 - Operation failed

### | For Personal System/2 Model 90 and Model 95:

(AL) = 00H - Disable cache L1 (cache is flushed when disabled)  
 = 01H - Enable cache L1 (cache is flushed when enabled)  
 = 02H - Disable cache L2 (cache is flushed when disabled)  
 = 03H - Enable cache L2 (cache is flushed when enabled)  
 = 04H - Disable both caches  
 = 05H - Enable both caches

On Return:

(AH) - Return code

- = 00H - Operation successfully completed
- = 01H - Function choice in (AL) is invalid
- = 02H - NVRAM data is invalid
- = 03H - Cache test error
- = 04H - Cannot perform the operation because of the state of the other cache
- = 05H - No L2 cache is present

CF = 0 - Operation successfully completed

= 1 - Operation failed

(AL) = 06H - Return status of both caches

On Return:

(BH) - Status of cache L2

= 0 - Enabled

= 1 - Disabled

(BL) - Status of cache L1

= 0 - Enabled

= 1 - Disabled

If the L2 cache is not installed, the L2 cache status is always set to 1 (disabled).

(AH) - Return code

- = 00H - Operation successfully completed
- = 01H - Function choice in (AL) is invalid
- = 02H - NVRAM data is corrupt
- = 03H - Cache test error
- = 04H - Cannot perform the operation because of the state of the other cache
- = 05H - No L2 cache is present

CF = 0 - Operation successfully completed

= 1 - Operation failed

(AL) = 07H to FFH - Reserved

**For all other Personal System/2 systems, this function is not supported:**

On Return:

(AH) = 86H - Function not supported

CF = 0 - Operation successfully completed

= 1 - Operation failed

### **((AH) = C9H—Processor Type and Stepping Level**

To obtain a complete Micro Channel configuration table, the Return System Configuration Parameters function ((AH)=C0H), the Return Memory-Map Information function ((AH)=C7H), and the Return Device Descriptor Table (DDT) Information function ((AH)=D1H) must be called in addition to the Processor Type and Stepping Level function ((AH)=C9H).

This function is not supported on all systems. If the function is not supported, the carry flag is set to 1, and the value that is returned in (AH) is either hex 80 (for PCjr and PC) or hex 86.

On Return:

(CH) - Microprocessor type

= 03 - 80386

= 23 - 80386 SX

= 04 - 80486

(CL) - Microprocessor stepping level

(AH) - Return code

= 00H - Operation successfully completed

= 80H or 86H - Function not supported

CF = 0 - Operation successfully completed

= 1 - Operation failed or function not supported

**(AH) = CAH to CDH—Reserved**

**(AH) = CEH—Allocate DMA Arbitration Level**

This function is not supported on all systems. If the function is not supported, the carry flag is set to 1, and the value that is returned in (AH) is either hex 80 (for PCjr and PC) or hex 86.

DMA channels are either physical or virtual. A physical channel can have only one arbitration level assigned to it. A virtual channel can be programmed to use any arbitration level that is not currently assigned to a different channel.

There is no difference in function between physical and virtual channels. Priority of the channels is determined by the arbitration level; arbitration level 0 has the highest priority, and arbitration level hex 0E has the lowest priority.

To perform a DMA transfer operation, the caller performs the following steps:

1. Request that CBIOS DMA allocate an arbitration level.
2. Set up the hardware and perform the transfer to a device.
3. Request that CBIOS DMA deallocate the arbitration level. See the Deallocate DMA Arbitration Level function ((AH) = CFH) on page 2-IN15-39.

Failure to use this function for the allocation of arbitration levels can cause unpredictable results.

(BL) = 00H to 0EH - Arbitration level to be allocated  
= 0FH to FFH - Reserved

(AL) - Option byte

Bits 7 to 1 = 0 - Reserved

Bit 0 = 0 - Need DMA channel for arbitration level requested  
= 1 - No channel required for arbitration level

On Return:

(AL) - Channel number

= 00H to 07H - Channel number allocated for the arbitration level

= 08H to FEH - Reserved

= FFH - No channel requested for arbitration level

(AH) - Return code

= 00H - Operation successfully completed

= 01H - Arbitration level not available

= 02H - Channel not available

= 03H - Invalid arbitration level passed

= 80H or 86H - Function not supported

CF = 0 - Operation successfully completed

= 1 - Operation failed or function not supported

### **(AH) = CFH—Deallocate DMA Arbitration Level**

This function is not supported on all systems. If the function is not supported, the carry flag is set to 1, and the value that is returned in (AH) is either hex 80 (for PCjr and PC) or hex 86.

(BL) = 00H to 0EH - Arbitration level to be deallocated  
= 0FH to FFH - Reserved

On Return:

(AH) - Return code

= 00H - Operation successfully completed

= 04H - Arbitration level not allocated

= 80H or 86H - Function not supported

CF = 0 - Operation successfully completed

= 1 - Operation failed or function not supported

Failure to use this function for the allocation of arbitration levels can cause unpredictable results.

### **(AH) = D0H—Reserved**

## **(AH) = D1H—Return Device Descriptor Table (DDT) Information**

This function is not supported on all systems. If the function is not supported, the carry flag is set to 1, and the value that is returned in (AH) is either hex 80 (for PCjr and PC) or hex 86.

To obtain a complete Micro Channel configuration table, the Return System Configuration Parameters function ((AH)=C0H), the Return Memory-Map Information function ((AH)=C7H), and the Processor Type and Stepping Level function ((AH)=C9H) must be called in addition to the Return Device Descriptor Table Information function ((AH)=D1H).

(AL) = C0H - Return number of device descriptor table (DDT) entries  
(DX) - Reserved (set to 0)

### **On Return:**

(BL) - Size of one DDT entry, in bytes  
(CX) - Number of DDT entries  
(AH) - Return code  
    = 00H - Operation successfully completed  
    = 01H - Requested DDT entry not found  
    = 02H - DDT data not valid  
    = 86H - Function not supported  
CF = 0 - Operation successfully completed  
    = 1 - Operation failed or function not supported

(AL) = 01H - Return DDT entry on the basis of the entry number  
(BX) - Number of requested entry (one-based)  
(DX) - Reserved (set to 0)  
(ES:DI) - Pointer to buffer that contains DDT entry

### **On Return:**

(ES:DI) - Pointer to buffer that contains DDT entry  
(AH) - Return code  
    = 00H - Operation successfully completed  
    = 01H - Requested DDT entry not found  
    = 02H - DDT data not valid  
    = 86H - Function not supported  
CF = 0 - Operation successfully completed  
    = 1 - Operation failed or function not supported

(AL) = 02H - Return DDT entry on the basis of the I/O port address  
(The DDT is searched from the specified entry  
for the I/O port in (CX), and the first entry that is  
found is returned.)

(BX) - Entry number from which to start search

(CX) - Requested I/O port address

(DX) - Reserved (set to 0)

(ES:DI) - Pointer to buffer that contains DDT entry

**On Return:**

(BX) - DDT entry number where requested I/O port address  
was found

(If the requested I/O port address is not found,  
this value is the total number of DDT entries  
plus 1.)

(ES:DI) - Pointer to buffer that contains DDT entry

(AH) - Return code

= 00H - Operation successfully completed

= 01H - Requested DDT entry not found

= 02H - DDT data not valid

= 86H - Function not supported

CF = 0 - Operation successfully completed

= 1 - Operation failed or function not supported

(AL) = 03H - Return entire DDT

(DX) - Reserved (set to 0)

(ES:DI) - Pointer to buffer that contains DDT entry

**On Return:**

(ES:DI) - Pointer to buffer that contains DDT entry

(AH) - Return code

= 00H - Operation successfully completed

= 01H - Requested DDT entry not found

= 02H - DDT data not valid

= 86H - Function not supported

CF = 0 - Operation successfully completed

= 1 - Operation failed or function not supported

(AL) = 04H - Return DDT entry on the basis of the device ID

(The DDT is searched from the specified entry  
for the device ID in (CX), and the first entry  
that is found is returned.)

(BX) - Entry number from which to start search

(CX) - Requested device ID

(DX) - Reserved (set to 0)

(ES:DI) - Pointer to buffer that contains DDT entry

**On Return:**

(BX) - Entry number where requested device ID was found  
(If the requested device ID is not found, this value is the total number of DDT entries plus 1.)

(ES:DI) - Pointer to buffer that contains DDT entry

(AH) - Return code

= 00H - Operation successfully completed

= 01H - Requested DDT entry not found

= 02H - DDT data not valid

= 86H - Function not supported

CF = 0 - Operation successfully completed

= 1 - Operation failed or function not supported

The device descriptor table (DDT) contains information about resources that are used by each device. An adapter is considered to be a single device unless it is divided into multiple devices by the BEGIN and END keywords in the adapter descriptor files (refer to the *Personal System/2 Hardware Technical Reference* for a description of adapter descriptor files). The first byte of the data block that contains the DDT indicates the size of each DDT entry. The format of the DDT entry is shown in the following table.

Offset	Size	Bit	Definition
0	Byte	7 to 4 3 to 0	Reserved (set to 0) Slot of device (0 = system board)
1	Byte	7 to 4 3 to 0	Second interrupt for this device (0 = no second interrupt) First interrupt for this device (0 = no first interrupt)
2	Byte	7 to 4 3 to 0	Second arbitration level for this device First arbitration level for this device

**Figure 2-25 (Part 1 of 2). Device Descriptor Table (DDT) Entry Format**



Offset	Size	Bit	Definition
3	Word	15	DDT indicators
		15	Reserved (set to 0)
		14	Second arbitration-level indicator = 0 - No second arbitration level = 1 - Second arbitration level exists
		13	First arbitration-level indicator = 0 - No first arbitration level = 1 - First arbitration level exists
		12	Serial-interface type = 0 - Not RS-422 = 1 - RS-422
		11	Not address limited = 0 - Address limited = 1 - Not address limited
		10	DMA channel required = 0 - DMA channel not used = 1 - DMA channel used
		9	Second arbitration level can be shared = 0 - Not shared = 1 - Shared
		8	First arbitration level can be shared = 0 - Not shared = 1 - Shared
		7 to 0	Reserved (set to 0)
5	Byte		Reserved (set to 0)
6	Word		Device ID = 0 - No device ID
8	Word		Starting address of first I/O block = 0 - No first I/O block
10	Word		Starting address of second I/O block = 0 - No second I/O block
12	Word		Starting address of third I/O block = 0 - No third I/O block
14	DWord		Start of first non-system memory block = 0 - No first non-system memory block
18	Word		Size of first non-system memory block (number of 1KB blocks)
20	DWord		Start of second non-system memory block = 0 - No first non-system memory block
24	Word		Size of second non-system memory block (number of 1KB blocks)
<b>Note:</b> I/O block addresses and non-system memory addresses are listed in ascending order.			
26	Byte		Implementation identifier of the device
27	Byte		Implementation revision level of the device

**Figure 2-25 (Part 2 of 2). Device Descriptor Table (DDT) Entry Format**

**(AH) = D2H to D3H—Reserved**

**(AH) = D4H—Return Physical Fixed Disk Drive Number (Selectable Boot)**

This function is not supported on all systems. If the function is not supported, the carry flag is set to 1, and the value that is returned in (AH) is either hex 80 (for PCjr and PC) or hex 86.

(DL) - Logical fixed disk drive number

On Return:

(AL) - Physical fixed disk drive number

(AH) - Return code

= 00H - Operation successfully completed

= 01H - Specified logical drive number is invalid

= 86H - Function not supported

CF = 0 - Operation successfully completed

= 1 - Operation failed

**(AH) = D5H—Reserved**

**(AH) = D6H—Return Boot Device ID and Key**

(AL) = 00H - Read/write boot device ID

(BL) = 0 - Read

= 1 - Write

(DX) - Device ID to be written/read

On Return:

(AH) = 00H - Operation successfully completed

= 86H - Function not supported

CF = 0

(AL) = 01H - Read/write boot device key

(BL) = 0 - Read

= 1 - Write

(DX) - Device ID to be written/read

On Return:

(AH) = 00H - Operation successfully completed

= 86H - Function not supported

CF = 0

(AL) = 02H - Query boot reference partition

On Return:

(AL) - Status of reference-partition boot request

= 00H - Reference-partition boot not requested

= 01H - Reference-partition boot requested

(AH) = 00H - Operation successfully completed

= 86H - Function not supported

CF = 0

**(AH) = D7H to FFH—Reserved**

## Interrupt 16H—Keyboard

These routines provide keyboard support. The following is a summary of the keyboard functions of Interrupt 16H.

(AH) = 00H	— Keyboard Read
(AH) = 01H	— Keystroke Status
(AH) = 02H	— Shift Status
(AH) = 03H	— Set Typematic Rate
(AH) = 04H	— Keyboard Click Adjustment
(AH) = 05H	— Keyboard Write
(AH) = 06H to 08H	— Reserved
(AH) = 09H	— Keyboard Functionality Determination
(AH) = 0AH	— Return Keyboard ID
(AH) = 0BH to 0FH	— Reserved
(AH) = 10H	— Extended-Keyboard Read
(AH) = 11H	— Extended Keystroke Status
(AH) = 12H	— Extended Shift Status
(AH) = 13H to 1FH	— Reserved
(AH) = 20H	— Keyboard Read for the 122-Key Keyboard
(AH) = 21H	— Keystroke Status for the 122-Key Keyboard
(AH) = 22H	— Shift Status for the 122-Key Keyboard
(AH) = 23H to FFH	— Reserved

**Figure 2-26.** INT 16H Keyboard Functions

**Note:** All reserved input fields must be set to 0.

Functions (AH)=10H, 11H, and 12H support the 101-/102-key keyboard. Functions (AH)=20H, 21H, and 22H support the 122-key keyboard. The keyboard scan codes for these functions fall into three categories:

- When only one key produces an ASCII character, the scan code that is read from the keyboard port is the same as that with standard keyboards.
- When more than one key produces the same character, one of the keys generates the standard keyboard scan code. The other key generates a unique sequence of scan codes, enabling the system to differentiate between the keys.
- New scan codes are assigned to keys that do not exist on the standard keyboards.

The 101-/102-key keyboard functions and the 122-key keyboard functions enable new programs to take advantage of all categories and avoid compatibility problems with existing programs.

| If the system BIOS does not support the 101-/102-key keyboard functions and the 122-key keyboard functions, the scan code/character code combination that was put into the keyboard buffer by the keyboard interrupt handler is returned unchanged when the Keyboard Read function ((AH)=00H) and the Keystroke Status function ((AH)=01H) are called.

| If the system BIOS supports the 101-/102-key keyboard functions and the 122-key keyboard functions:

- Keys with identical names are differentiated by the character code that is put into the keyboard buffer by the keyboard interrupt handler.
- The keyboard interrupt handler puts the scan code/character code combination for new keys into the keyboard buffer.
- The Extended-Keyboard Read function ((AH)=10H), the Extended Keystroke Status function ((AH)=11H), the Keyboard Read function for the 122-Key Keyboard function ((AH)=20H), and the Keystroke Status for the 122-Key Keyboard function ((AH)=21H), extract the scan code/character code combination from the buffer as is and return it to the caller. The scan code/character code combination is returned for new keys. It is also returned for keys with identical names, and the character code is used to differentiate between them. If the character code is hex F0 and the scan code is not hex 00, the character code is set to hex 00.
- The Keyboard Read function ((AH)=00H) and the Keystroke Status function ((AH)=01H) extract the scan code/character code combination and translate it, if necessary, to the scan code/character code combination that is compatible with previous keyboards. The translation:
  1. Converts like codes to compatible codes
  2. Extracts scan code/character code combinations until a compatible combination is found.
- The Extended Shift Status function ((AH)=12H) and the Shift Status for the 122-Key Keyboard function ((AH)=22H) return the existing keyboard shift state and the shift state of the separate Ctrl and Alt keys.

| If the Keyboard Functionality Determination function ((AH)=09H) is supported, use it to determine whether the system BIOS supports the functions for the 101-/102-key keyboard ((AH)=10H, 11H, and 12H) and the 122-key keyboard ((AH)=20H, 21H, and 22H).

If the Keyboard Functionality Determination function ((AH)=09H) is not supported, the program must use the following procedure:

1. Use Interrupt 16H, Keyboard Write function ((AH)=05H) to write a scan code/character code combination of hex FFFF to the buffer.
2. If (AL) is set to 00H on return, the function has successfully written hex FFFF to the buffer.
3. Use Interrupt 16H, Extended-Keyboard Read function ((AH)=10H) or Keyboard Read for the 122-Key Keyboard function ((AH)=20H) to read the scan code/character code combination from the keyboard buffer.
4. If (AX) is set to hex FFFF on return, the functions for the 101-/102-key keyboard and the functions for the 122-key keyboard are supported.
5. If (AX) is not set to hex FFFF on return, call Interrupt 16H, (AH)=10H or (AH)=20H repeatedly until (AX) is set to hex FFFF on return.
6. If (AX) is still not set to hex FFFF after 16 calls to (AH)=10H or (AH)=20H, the functions for the 101-/102-key keyboard and the functions for the 122-key keyboard are not supported.

See the "Scan Code/Character Code Combinations" section.

### **(AH)=00H—Keyboard Read**

The scan code/character code combination is extracted from the buffer. The keyboard-buffer head pointer (address hex 40:1A) is increased by 2 or, if the pointer is already at the end, it is reinitialized to the start of the buffer.

On Return:

(AH) - Scan code

(AL) - ASCII character code

For AT, PC/XT Model 286, PC Convertible, and Personal System/2 products, if no keystroke is available, Interrupt 15H, Device Busy function ((AH)=90H) is called with (AL)=02H (Type=Keyboard) to inform the operating system that a keyboard loop is about to take place, enabling the operating system to perform another task. When the keyboard operation is complete, Interrupt 09H calls Interrupt 15H, Interrupt Complete function ((AH)=91H), with (AL)=02H (Type=Keyboard). See "Multitasking Provisions" in the "Additional Information" section.

**Note:** Control is returned only when a keystroke is available. The keystroke is removed from the buffer.

### **(AH) = 01H—Keystroke Status**

On Return:

CF = 0 - Code is available  
= 1 - No code is available

If a code is available:

(AH) - Scan code  
(AL) - ASCII character code

**Note:** The keystroke is not removed from the buffer.

### **(AH) = 02H—Shift Status**

On Return:

(AH) - Reserved

(AL) - Current shift status

Bit 7 = 1 - Insert locked  
Bit 6 = 1 - Caps Lock locked  
Bit 5 = 1 - Num Lock locked  
Bit 4 = 1 - Scroll Lock locked  
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

### **(AH) = 03H—Set Typematic Rate**

For PCjr and Personal System/2 BIOS that supports Interrupt 16H, Keyboard Functionality Determination function ((AH) = 09H) with bit 0 set to 1:

(AL) = 00H - Returns to default; restores original state  
(typematic on, normal initial delay and normal typematic rate)

For PCjr only:

(AL) = 01H - Increases the initial delay (this is the delay between the first character and the burst of typematic characters)  
(AL) = 02H - Slows typematic characters by one-half  
(AL) = 03H - Increases the initial delay and slows typematic characters by one-half

For PCjr and Personal System/2 BIOS that supports Interrupt 16H, Keyboard Functionality Determination function ((AH)=09H) with bit 1 set to 1:

(AL) = 04H - Turns off typematic characters

For AT BIOS dated 11/15/85 and later, PC/XT Model 286, and Personal System/2 products:

(AL) = 05H - Set typematic rate and delay

(BH) - Delay value (in milliseconds)

00H = 250

01H = 500

02H = 750

03H = 1000

04H to FFH - Reserved

(BL) - Typematic rate (in characters per second)

00H = 30.0      0BH = 10.9      16H = 4.3

01H = 26.7      0CH = 10.0      17H = 4.0

02H = 24.0      0DH = 9.2      18H = 3.7

03H = 21.8      0EH = 8.6      19H = 3.3

04H = 20.0      0FH = 8.0      1AH = 3.0

05H = 18.5      10H = 7.5      1BH = 2.7

06H = 17.1      11H = 6.7      1CH = 2.5

07H = 16.0      12H = 6.0      1DH = 2.3

08H = 15.0      13H = 5.5      1EH = 2.1

09H = 13.3      14H = 5.0      1FH = 2.0

0AH = 12.0      15H = 4.6      20H to FFH - Reserved

For Personal System/2 BIOS that supports Interrupt 16H, Keyboard Functionality Determination function ((AH)=09H) with bit 3 set to 1:

(AL) = 06H - Return current typematic rate and delay

On Return:

(BH) - Delay value

(BL) - Typematic rate

For all others, no action is performed.

### **((AH)=04H—Keyboard Click Adjustment**

For PCjr and PC Convertible:

(AL) = 00H - Set keyboard click off

(AL) = 01H - Set keyboard click on

For all others, no action is performed.

## **(AH) = 05H—Keyboard Write**

For AT BIOS dated 11/15/85 and later, PC/XT dated 1/10/86 and later, PC/XT Model 286, and Personal System/2 products, this function puts scan code/character code combinations into the keyboard buffer as if they came from the keyboard.

(CH) - Scan code

(CL) - ASCII character code

On Return:

(AL) = 00H - Operation successfully completed

= 01H - Buffer full

For all others, no action is performed.

## **(AH) = 06H to 08H—Reserved**

## **(AH) = 09H—Keyboard Functionality Determination**

To determine whether Interrupt 16H, Keyboard Functionality Determination function ((AH)=09H) is supported, the program must use Interrupt 15H, Return System Configuration Parameters function ((AH)=C0H), testing bit 6 of feature information byte 2. If this bit is set to 0, the function is not supported. If this bit is set to 1, the function is as follows. This test can be performed for all systems that support Interrupt 15H, Return System Configuration Parameters function ((AH)=C0H).

On Return:

(AL) - Function code

Bit 7 - Reserved

Bit 6 = 1 - 122-key keyboard functions are supported

= 0 - 122-key keyboard functions are not supported

Bit 5 = 1 - 101-/102-key keyboard functions are supported

= 0 - 101-/102-key keyboard functions are not supported

Bit 4 = 1 - Keyboard ID is supported

= 0 - Keyboard ID is not supported

Bit 3 = 1 - Get current typematic rate/delay is supported

= 0 - Get current typematic rate/delay is not supported

Bit 2 = 1 - Set typematic rate/delay is supported

= 0 - Set typematic rate/delay is not supported

Bit 1 = 1 - Turn on/off typematic is supported

= 0 - Turn on/off typematic is not supported

Bit 0 = 1 - Return to default typematic rate/delay is supported

= 0 - Return to default typematic rate/delay is not supported



## **| (AH) = 0AH—Return Keyboard ID**

**| Use this function to determine the type of keyboard that is attached.**

**| To determine whether this function is supported, use Interrupt 16H, Keyboard Functionality Determination function ((AH)=09H), testing bit 4 of the AL register. If this bit is set to 0, the function is not supported. If this bit is set to 1, the function is as follows:**

**On Return:**

**(BX) = Keyboard ID**

**= 0000 - No keyboard attached**

**= 0001 to FFFFH - Keyboard attached, reserved for keyboard IDs**

**(BH) - Second byte of keyboard ID**

**(BL) - First byte of keyboard ID**

**| The valid keyboard IDs are:**

**(BX) = 41AB - G-layout, 101- and 102-key (translated)**

**= 83AB - G-layout, 101- and 102-key**

**= 54AB - P-layout (translated)**

**= 84AB - P-layout**

**= 86AB - 122-key**

**= 90AB - DBCS G-layout (5576-002)**

**= 91AB - DBCS P-layout (5576-003)**

**= 92AB - DBCS A-layout (5576-001)**

## **(AH) = 0BH to 0FH—Reserved**

## **(AH) = 10H—Extended-Keyboard Read**

For AT BIOS dated 11/15/85 and later, PC/XT dated 1/10/86 and later, PC/XT Model 286, and Personal System/2 products, the scan code/character code combination is extracted from the buffer. The keyboard-buffer head pointer (address hex 40:1A) is increased by 2 or, if the pointer is already at the end, it is reinitialized to the start of the buffer.

**On Return:**

**(AH) - Scan code**

**(AL) - ASCII character code**

**Note:** Control is returned only when a keystroke is available. The keystroke is removed from the buffer.

For all others, no action is performed.

## **(AH) = 11H—Extended Keystroke Status**

For AT BIOS dated 11/15/85 and later, PC/XT dated 1/10/86 and later, PC/XT Model 286, and Personal System/2 products:

On Return:

CF = 0 - Code is available  
= 1 - No code is available

If a code is available:

(AL) - ASCII character code  
(AH) - Scan code

**Note:** The keystroke is not removed from the buffer.

For all others, no action is performed.

## **(AH) = 12H—Extended Shift Status**

For AT BIOS dated 11/15/85 and later, PC/XT dated 1/10/86 and later, PC/XT Model 286, and Personal System/2 products:

On Return:

(AH) - Extended shift status

Bit 7 = 1 - SysRq key pressed  
Bit 6 = 1 - Caps Lock key pressed  
Bit 5 = 1 - Num Lock key pressed  
Bit 4 = 1 - Scroll Lock key pressed  
Bit 3 = 1 - Right Alt key pressed  
Bit 2 = 1 - Right Ctrl key pressed  
Bit 1 = 1 - Left Alt key pressed  
Bit 0 = 1 - Left Ctrl key pressed

(AL) - Shift status

Bit 7 = 1 - Insert locked  
Bit 6 = 1 - Caps Lock locked  
Bit 5 = 1 - Num Lock locked  
Bit 4 = 1 - Scroll Lock locked  
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

For all others, no action is performed.

## **(AH) = 13H to 1FH—Reserved**

## | **(AH) = 20H—Keyboard Read for the 122-Key Keyboard**

| For Personal System/2 Model 35 SX, Model 35 LS, Model 40 SX, and  
| Model 57 SX, the scan code/character code combination is extracted  
| from the buffer. The keyboard-buffer head pointer (address hex  
| 40:1A) is increased by 2 or, if the pointer is already at the end, it is  
| reinitialized to the start of the buffer.

| On Return:

| (AH) - Scan code

| (AL) - ASCII character code

| **Note:** Control is returned only when a keystroke is available. The  
| keystroke is removed from the buffer.

## | **(AH) = 21H—Keystroke Status for the 122-Key Keyboard**

| For Personal System/2 Model 35 SX, Model 35 LS, Model 40 SX,  
| and Model 57 SX:

| On Return:

| CF = 0 - Code is available

| = 1 - No code is available

| If a code is available:

| (AL) - ASCII character code

| (AH) - Scan code

| **Note:** The keystroke is not removed from the buffer.

**(AH) = 22H—Shift Status for the 122-Key Keyboard**

For Personal System/2 Model 35 SX, Model 35 LS, Model 40 SX,  
and Model 57 SX:

On Return:

(AH) - Extended shift status

- Bit 7 = 1 - SysRq key pressed
- Bit 6 = 1 - Caps Lock key pressed
- Bit 5 = 1 - Num Lock key pressed
- Bit 4 = 1 - Scroll Lock key pressed
- Bit 3 = 1 - Right Alt key pressed
- Bit 2 = 1 - Right Ctrl key pressed
- Bit 1 = 1 - Left Alt key pressed
- Bit 0 = 1 - Left Ctrl key pressed

(AL) - Shift status

- Bit 7 = 1 - Insert locked
- Bit 6 = 1 - Caps Lock locked
- Bit 5 = 1 - Num Lock locked
- Bit 4 = 1 - Scroll Lock locked
- 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

**(AH) = 23H to FFH—Reserved**

## Interrupt 17H—Printer

These routines provide printer support. The following is a summary of the printer-support functions of Interrupt 17H.

(AH) = 00H — Print Character  
(AH) = 01H — Initialize the Printer Port  
(AH) = 02H — Read Status  
(AH) = 03H to FFH — Reserved

*Figure 2-27. INT 17H Printer Functions*

**Note:** All reserved input fields must be set to 0.

### **(AH) = 00H—Print Character**

(AL) - Character to be printed  
(DX) - Printer to be used (0, 1, or 2); index into the port base address table at address hex 40:08

On Return:

(AH) - Status  
Bit 7 = 1 - Not busy  
Bit 6 = 1 - Acknowledgment  
Bit 5 = 1 - Out of paper  
Bit 4 = 1 - Selected  
Bit 3 = 1 - I/O error  
Bits 2, 1 - Reserved  
Bit 0 = 1 - Time-out

### **(AH) = 01H—Initialize the Printer Port**

(DX) - Printer to be used (0, 1, or 2); index into the port base address table at address hex 40:08

On Return:

(AH) - Status (see status values on page 2-IN17-1)

### **(AH) = 02H—Read Status**

(DX) - Printer to be used (0, 1, or 2); index into the port base address table at address hex 40:08

On Return:

(AH) - Status (see status values on page 2-IN17-1)

### **(AH) = 03H to FFH—Reserved**

## **Notes:**

1. For AT, PC/XT Model 286, PC Convertible, and Personal System/2 products, when the printer is busy, BIOS calls Interrupt 15H, Device Busy function ((AH)=90H) with (AL)=FEH (Type=Printer) to inform the operating system that a time-out loop is about to begin. See "Multitasking Provisions" in the "Additional Information" section.
2. For AT BIOS dated before 11/15/85, PCjr, PC, and PC/XT BIOS dated 11/8/82, the printer-port number that is associated with (DX) is tested for 0. If (DX) is set to 0, no action occurs. If (DX) is not set to 0, the print operation is performed. The (DX) register is not tested for a valid printer-port number at the offset into the printer base address data area at address hex 40:08.
3. For PC/XT BIOS dated 1/10/86 and later, if the value of (DX) is greater than 3 or the printer port that is associated with (DX) is printer port 0, no action is performed, and on return, (AH) is set to 29H.
4. For PC Convertible, if the printer port that is associated with (DX) is printer port 0, (AH) is set to 01H on return. If (DX) is not set to 0, the print operation is performed. The (DX) register is not tested to determine whether a valid printer-port number exists at the offset into the printer base address data area at address hex 40:08.
5. For AT BIOS dated 11/15/85 and PC/XT Model 286, if the value of (DX) is greater than 3 or the printer port that is associated with (DX) is printer port 0, no action is performed, and (AH) is returned unchanged.
6. For Personal System/2 products, if the value of (DX) is greater than 2 or the printer port that is associated with (DX) is printer port 0, no action is performed, and (AH) is returned unchanged.

## Interrupt 19H—Bootstrap Loader

Devices that are supported by Interrupt 13H read the boot record from cylinder 0, head 0, sector 1. Devices that are supported by Interrupt 4BH read the boot record from logical block address 1. The boot record is then read into system memory at address hex 0000:7C00.

The power-on self-test (POST) checks for the following characteristics to validate a boot record:

- For a device with removable media, such as a diskette, the boot record must contain valid code. For example, the value of the first byte of the boot record must be greater than 6.
- For a device with nonremovable media, such as a fixed disk, there must be a signature of hex 55AA at the end of the boot record.

Parameters that identify the device from which the boot record was obtained are passed to the boot record. Passing of these parameters enables the boot record to continue the system boot operation.

In systems that do not support Interrupt 15H, Return Boot Device and Key function ((AH)=D6H), the following information is passed to the boot record:

(CS) = 0000H  
 (IP) = 7C00H  
 (DL) - Drive from which the boot record was read

In systems that support Interrupt 15H, Return Boot Device and Key function ((AH)=D6H), information is passed also to the DX register. If the access-mode bit (bit 5) is set to 0, the DX register is defined as the "boot device identifier"; if the access-mode bit is set to 1, the DX register is defined as the "device key." The device key is a 16-bit value that is returned by Interrupt 4BH, Allocate Device function ((AH)=80H, (AL)=02H), (AX)=8002H. (The device key is available also from Interrupt 15H, Return Boot Device ID and Key function ((AH)=D6H).)

| The following information is passed to the boot record:

| (CS) = 0000H

| (IP) = 7C00H

| (DX) - Boot device identifier or device key

| When (DX) is a boot device identifier, (DH) and (DL) are defined as follows:

| (DH) - Device identifier

|   Bit 7 - Removable-media indicator

|     = 0 - Nonremovable media

|     = 1 - Removable media

|   Bit 6 - Reserved

|   Bit 5 - Access mode

|     = 0 - Device is supported by interrupt 13H

|     = 1 - Device is supported by interrupt 4BH

|   Bits 4 to 0 - Peripheral-device type

| (DL) - Device instance

|   = 00H - First diskette drive (if the access-mode bit is set to 0)

|     - First CD-ROM drive (if the access-mode bit is set to 1)

|   = 80H - First fixed disk drive (if the access-mode bit is set to 0)



## Interrupt 1AH—System-Timer and Real-Time Clock Services

The following is a summary of the system-timer and real-time clock services of Interrupt 1AH.

(AH) = 00H	– Read System-Timer Time Counter
(AH) = 01H	– Set System-Timer Time Counter
(AH) = 02H	– Read Real-Time Clock Time
(AH) = 03H	– Set Real-Time Clock Time
(AH) = 04H	– Read Real-Time Clock Date
(AH) = 05H	– Set Real-Time Clock Date
(AH) = 06H	– Set Real-Time Clock Alarm
(AH) = 07H	– Reset Real-Time Clock Alarm
(AH) = 08H	– Set Real-Time Clock Activated Power-On Mode
(AH) = 09H	– Read Real-Time Clock Alarm Time and Status
(AH) = 0AH	– Read System-Timer Day Counter
(AH) = 0BH	– Set System-Timer Day Counter
(AH) = 0CH to 7FH	– Reserved
(AH) = 80H	– Set Up Sound Multiplexer
(AH) = 81H to FFH	– Reserved

**Figure 2-28. INT 1AH System-Timer and Real-Time Clock Services Functions**

### Notes:

1. All reserved input fields must be set to 0.
2. Some models of Personal System/2 Model 25 do not support a real-time clock; therefore, functions (AH)=02H through (AH)=09H do not apply to those systems. The system-timer functions do apply.

### (AH)=00H—Read System-Timer Time Counter

On Return:

- (AL) = 0 - Timer has not passed 24 hours' worth of counts since power-on, last system reset, last system-timer time counter read, or last system-timer time counter set
- > 0 - Timer has passed 24 hours' worth of counts since power-on, last system reset, last system-timer time counter read, or last system-timer time counter set
- (CX) - High portion of count
- (DX) - Low portion of count

**Note:** Execution causes the timer overflow flag (hex 40:70) to be reset to 0. Counts occur at the rate of approximately 18.2 per second.

## **(AH) = 01H—Set System-Timer Time Counter**

(CX) - High portion of count

(DX) - Low portion of count

**Note:** Execution causes the timer overflow flag (hex 40:70) to be reset to 0. Counts occur at the rate of approximately 18.2 counts per second.

## **(AH) = 02H—Read Real-Time Clock Time**

For AT BIOS dated before 6/10/85:

On Return:

(CH) - Hours, in binary-coded decimal notation (BCD)

(CL) - Minutes, in BCD

(DH) - Seconds, in BCD

CF = 0 - Clock operating

= 1 - Clock not operating

For AT BIOS dated 6/10/85 and later, PC/XT Model 286, PC Convertible, and Personal System/2 products:

On Return:

(CH) - Hours, in BCD

(CL) - Minutes, in BCD

(DH) - Seconds, in BCD

(DL) = 01H - Daylight saving time option

= 00H - No daylight saving time option

CF = 0 - Clock operating

= 1 - Clock not operating

For all others, no action is performed.

## **(AH) = 03H—Set Real-Time Clock Time**

For AT, PC/XT Model 286, PC Convertible, and Personal System/2 products:

(CH) - Hours, in BCD

(CL) - Minutes, in BCD

(DH) - Seconds, in BCD

(DL) = 01H - Daylight saving time option

= 00H - No daylight saving time option

**Note:** For Personal System/2 Model 30, (DL) is not used.

For all others, no action is performed.

### **(AH) = 04H—Read Real-Time Clock Date**

For AT, PC/XT Model 286, PC Convertible, and Personal System/2 products:

On Return:

(CH) - Century (19 or 20), in BCD

(CL) - Year, in BCD

(DH) - Month, in BCD

(DL) - Day, in BCD

CF = 0 - Clock operating

= 1 - Clock not operating

For all others, no action is performed.

### **(AH) = 05H—Set Real-Time Clock Date**

For AT, PC/XT Model 286, PC Convertible, and Personal System/2 products:

(CH) - Century (19 or 20), in BCD

(CL) - Year, in BCD

(DH) - Month, in BCD

(DL) - Day, in BCD

For all others, no action is performed.

### **(AH) = 06H—Set Real-Time Clock Alarm**

For AT, PC/XT Model 286, PC Convertible, and Personal System/2 products:

(CH) - Hours, in BCD

(CL) - Minutes, in BCD

(DH) - Seconds, in BCD

On Return:

CF = 0 - Operation successfully completed

= 1 - Alarm already set or clock not operating

**Note:** The alarm interrupt occurs at the specified hour, minute, and second that are passed in (CH), (CL), and (DH), respectively. When the alarm interrupt occurs, software Interrupt 4AH is issued. The user must point software Interrupt 4AH to an alarm routine before setting the real-time clock alarm through Interrupt 1AH, Set Real-Time Clock Alarm function (AH)=06H). Only one alarm function can be active at any time. The alarm interrupt occurs every 24 hours at the specified time until it is reset.

For all others, no action is performed.

**(AH) = 07H—Reset Real-Time Clock Alarm**

For AT, PC/XT Model 286, PC Convertible, and Personal System/2 products, this function stops the real-time clock alarm interrupt from occurring.

For all others, no action is performed.

**(AH) = 08H—Set Real-Time Clock Activated Power-On Mode**

For PC Convertible:

(CH) - Hours, in BCD  
(CL) - Minutes, in BCD  
(DH) - Seconds, in BCD

On Return:

CF = 0 - Operation successfully completed  
= 1 - Alarm already set or clock not operating

For AT BIOS dated 6/10/85 and later, PC/XT Model 286, and Personal System/2 products, this function is not supported:

On Return:

CF = 1 - Invalid function request

For all others, no action is performed.

**(AH) = 09H—Read Real-Time Clock Alarm Time and Status**

For PC Convertible and Personal System/2 Model 30:

On Return:

(CH) - Hours, in BCD  
(CL) - Minutes, in BCD  
(DH) - Seconds, in BCD  
(DL) - Alarm status  
= 00H - Alarm not enabled  
= 01H - Alarm enabled but will not power-on the system  
= 02H - Alarm enabled and will power-on the system

**Note:** Personal System/2 Model 30 does not support the power-on system feature.

For AT BIOS dated 6/10/85 and later, PC/XT Model 286, and Personal System/2 products except Model 30, this function is not supported:

On Return:  
CF = 1 - Invalid function request

For all others, no action is performed.

**(AH) = 0AH—Read System-Timer Day Counter**

For AT and PC/XT Model 286, this function is not supported:

On Return:  
CF = 1 - Invalid function request

For PC/XT BIOS dated 1/10/86 and later and Personal System/2 products:

On Return:  
(CX) - Count of days since 1 January 1980 (1-1-80)

**Note:** The count of days since 1 January 1980 is initialized to 0 during POST.

For all others, no action is performed.

**(AH) = 0BH—Set System-Timer Day Counter**

For AT and PC/XT Model 286, this function is not supported:

On Return:  
CF = 1 - Invalid function request

For PC/XT BIOS dated 1/10/86 and later and Personal System/2 products:

(CX) - Count of days since 1 January 1980 (1-1-80)

**Note:** The count of days since 1 January 1980 is initialized to 0 during POST.

For all others, no action is performed.

**(AH) = 0CH to 7FH—Reserved**

## **(AH) = 80H—Set Up Sound Multiplexer**

For PCjr:

- (AL) - Source of sound ("Audio Out" or RF modulator)
  - = 00H - 8253 channel 2
  - = 01H - Cassette input
  - = 02H - "Audio In" line on I/O channel
  - = 03H - Complex sound generator chip

For AT BIOS dated 6/10/85 and later, PC/XT Model 286, PC Convertible, and Personal System/2 products, this function is not supported:

On Return:

CF = 1 - Invalid function request

For all others, no action is performed.

## **(AH) = 81H to FFH—Reserved**

## Interrupt 4BH—Advanced Services

The following is a summary of the services available through Interrupt 4BH.

(AH) = 00H to 7FH — Reserved
(AH) = 80H — SCSI Devices (page 2-IN4B-2)
(AH) = 81H — Operating-System DMA Services (page 2-IN4B-15)
(AH) = 82H to FFH — Reserved

*Figure 2-29. INT 4BH Advanced Services*

**Note:** All reserved input fields must be set to 0.

The BIOS services that are available through Interrupt 4BH provide an interface to certain functions. Before making calls to these routines, test the appropriate bits at address hex 40:7B to determine whether the routines are supported.

Services in the SCSI Devices function ((AH)=80H) provide a generic interface to small computer system interface (SCSI) devices.

Services in the Operating-System DMA Services function ((AH)=81H) provide bus masters and operating systems with an interface for controlling memory addresses.

## **(AH) = 80H—SCSI Devices**

These BIOS services are intended for use by device drivers. Application programs should use operating-system interfaces to access small computer system interface (SCSI) devices.

The following is a summary of the SCSI functions.

(AH) = 00H	— Get Device Count (page 2-IN4B-3)
(AH) = 01H	— Get Device Data (page 2-IN4B-4)
(AH) = 02H	— Allocate Device (page 2-IN4B-4)
(AH) = 03H	— Deallocate Device (page 2-IN4B-5)
(AH) = 04H	— Send Device SCB (page 2-IN4B-5)
(AH) = 05H	— Send Device Immediate (page 2-IN4B-8)
(AH) = 06H	— Reset Adapter (page 2-IN4B-8)
(AH) = 07H	— Set Time-Out Value (page 2-IN4B-9)
(AH) = 08H to FFH	— Reserved

*Figure 2-30. SCSI Device Functions*

IBM fixed disk drives are defined as having SCSI peripheral type 0, 512-byte sectors, and nonremovable media.

## **BIOS Data Area**

The following flags are located at address hex 40:7B in the BIOS data area. These flags indicate the services that are supported through Interrupt 4BH.

Address hex 40:7B

Bits 7 to 6 - Reserved

Bit 5 - Operating-system DMA services indicator

    = 0 - Not supported; all memory is mapped linear=physical

    = 1 - Supported

Bit 4 - Reserved

Bit 3 - Interrupt-4BH-intercepted indicator

    = 0 - Interrupt vector is not intercepted

    = 1 - Interrupt vector is intercepted

Bit 2 - Reserved

Bit 1 - Generic SCSI CBIOS services support

    = 0 - Interrupt 4BH does not support SCSI

    = 1 - Interrupt 4BH supports SCSI

Bit 0 - Reserved

If bit 5 is set to 1, DMA services are available. If bit 5 is set to 0, DMA services are not available.

Providers of the operating-system DMA services must set bit 3 when hooking into Interrupt 4BH.



If bit 3 is set to 1, a program that provides services through Interrupt 4BH must link into the interrupt chain by saving the interrupt vector for Interrupt 4BH and replacing the old vector with the pointer to the start of the program's code. Then, if the program does not provide the requested services, it can pass the call to the next link in the chain.

For Personal System/2 systems with Micro Channel architecture, if bit 3 is set to 0, the following test must be performed:

If bit 1 of 40:7B is set to 0,

— or —

the contents of the Interrupt 4BH vector are 0:0,

— or —

the Interrupt 4BH vector contains a segment value of hex E000 or hex F000,

the Interrupt vector must not be chained.

Else, the interrupt vector must be chained.

If this test indicates that the interrupt vector must not be chained, the program puts its pointer into the vector and sets bit 3 to 1; however, if it does not provide the requested services, the program must return status that indicates that the call is not supported ((AH)=86H and CF=1).

### **(AL) = 00H—Get Device Count**

This function returns the total count of SCSI devices in the system for the specified peripheral type.

(DL) - Device-type indicator

Bit 7 = 1 - Removable media

Bits 6 to 0 - SCSI peripheral type

On Return:

(AH) - Return code

(CL) - Device count

CF = 0 - Return code is 0

= 1 - Return code is non 0

## **(AL) = 01H—Get Device Data**

This function returns various parameters for the specified SCSI device.

- (DL) - Device-type indicator
  - Bit 7 = 1 - Removable media
  - Bits 6 to 0 - SCSI peripheral type
- (DH) - Device index (range is from 1 to the device count)

### **On Return:**

- (AL) - Device status
  - Bits 7 - Reserved
  - Bit 6 = 0 - Scatter/gather supported in hardware
    - = 1 - Scatter/gather not supported in hardware
  - Bit 5 = 1 - Device allocated
  - Bit 4 = 0 - Cache buffer not enabled
    - = 1 - Cache buffer enabled
  - Bit 3 = 1 - IBM fixed disk drive (nonremovable media, 512-byte sectors)
  - Bit 2 = 1 - Removable media
  - Bit 1 = 1 - Defective device (error during POST)
  - Bit 0 = 1 - Device not powered-on (during POST)
- (BH) - Level
  - Bits 7 to 4 - Reserved
  - Bits 3 to 0 - Adapter number
- (BL) - Unit number
  - Bits 7 to 4 - Logical unit number
  - Bits 3 to 0 - Physical unit number
- (CL) - Time-out value
  - Bit 7 = 0 - Time-out count, in seconds
    - = 1 - Time-out count, in minutes
  - Bits 6 to 0 - Time-out count
- (AH) - Return code
  - CF = 0 - Return code is 0
  - = 1 - Return code is non 0

## **(AL) = 02H—Allocate Device**

This function returns a device key for the specified SCSI device. This device key is unique for each device and is used with the Deallocate Device function ((AL) = 03H), the Send SCB to Device function ((AL) = 04H), the Send Immediate Command to Device function ((AL) = 05H), the Reset Adapter function ((AL) = 06H), and the Set Time-Out Value function ((AL) = 07H).

The value of the device key varies, depending on the configuration, the BIOS version, and other factors. Therefore, to ensure compatibility, use this function to obtain a valid device key. The device key is valid only when the carry flag is set to 0.

If the device has already been allocated, a device key is not returned.

(DH) - Device index  
= 0 - Allocate next available device  
= From 1 to the device count - Allocate specified device  
(DL) - Device-type indicator  
Bit 7 = 1 - Removable media  
Bits 6 to 0 - SCSI peripheral type

On Return:  
(AH) - Return code  
(DX) - Device key  
CF = 0 - Return code is 0  
= 1 - Return code is non 0

**Note:** IBM fixed disk drives are accessible through Interrupt 13H—Fixed Disk. Any device driver that allocates an IBM fixed disk drive through this function must also intercept Interrupt 13H requests for that device, or substitute for the operating-system device driver that issues the Interrupt 13H requests.

**(AL) = 03H—Deallocate Device**

This function frees the device that is associated with the specified device key so that it can be allocated by another device driver. After the device has been deallocated, an error occurs if that device key is used. The Allocate Device function must be used to regain access to the device.

(DX) - Device key  
  
On Return:  
(AH) - Return code  
CF = 0 - Return code is 0  
= 1 - Return code is non 0

**(AL) = 04H—Send SCB to Device**

This function sends a subsystem control block (SCB) command to the specified SCSI device. (Refer to the technical references for the SCSI adapter and the device for details on the SCB structure and other programming information.)

(CL) - SCB type indicator

Bits 7 to 4 - Reserved

Bit 3 = 0 - Original chain header or no chain header passed  
(see note on page 2-IN4B-7)

= 1 - Extended chain header passed

Bit 2 = 0 - 32-bit SCB physical address pointer not passed  
(SI and DI are undefined on input)

= 1 - 32-bit SCB physical pointer passed in (SI,DI)

Bit 1 = 0 - No chain header or extended chain header passed  
(see note on page 2-IN4B-7)

= 1 - Original chain header passed

Bit 0 = 0 - Send SCB (use with all commands except the  
Send Other SCSI SCB command)

= 1 - Send long SCB (use with Send Other SCSI SCB  
command)

(DX) - Device key

(ES:BX) - Logical pointer to SCB

(SI,DI) - 32-bit physical address pointer to SCB (SI is MSW;  
DI is LSW; valid when bit 2 of CL is set to 1)

On Return:

(AH) - Return code

(AL) - Error status (see "Return Codes" on page 2-IN4B-10 for a definition  
of this error status)

CF = 0 - Return code is 0

= 1 - Return code is non 0

## Notes:

1. The Read and Write SCB commands are the primary interface to devices such as IBM fixed disk drives. The Send Other SCSI command should not be used to read or write to IBM fixed disk drives because system data might be lost.
2. All addresses within the subsystem control block (SCB) are physical. Before making a call to BIOS, the calling program must translate logical addresses to physical addresses and ensure that all subsystem control blocks, termination status blocks, and data buffers are locked into memory. See Interrupt 4BH, Operating-System DMA Services function ((AH) = 81H).

3. When SCBs are chained, (ES:BX) points to the first SCB. Each SCB is built with a chain header, which immediately precedes the SCB. The format of the chain header is as follows:

Size	Offset	Description
Word	—10H	Reserved
DWord	—0CH	Logical-pointer data buffer
Word	—0AH	Reserved
Word	—08H	Reserved
DWord	—04H	Logical pointer to scatter/gather list (set to 0 if no scatter/gather list)
Word	—02H	Reserved
Word	00H	Reserved
DWord	02H	Logical pointer to next SCB header in chain, or chain-ending indicator (0)
Word	06H	Reserved
Word	08H	Reserved
DWord	0AH	Logical pointer to TSB associated with this SCB
Word	0EH	Reserved
	10H	Beginning of SCB

- A chain length of 1 is allowed.
- The chain must have an ending.
- A logical pointer to the next SCB header in the chain equal to 0 ends the chain.
- The chain header must immediately precede the SCB.

Bit 2 in (CL) can be set, and the physical address to the first SCB can be passed in (SI,DI). However, (ES:BX) and the chain headers must still be built and passed. If the linear address of the SCB does not equal the physical address, the physical address must be passed to BIOS. See Interrupt 4BH, Operating-System DMA Services function ((AH)=81H).

See "Send Device SCB Chaining" on page 2-IN4B-12 for chaining data structures and example.

To determine the chain-header type, the SCB-indicator bits 1 and 3 must be evaluated together. There are two types of chain headers as input to BIOS: the original header and the extended header. It is recommended that all software use the extended header as input. Using the extended header ensures portability to non-bus-master implementations of SCSI controllers. Bits 1 and 3 are evaluated as follows:

- When bit 3 is set to 0 and bit 1 is set to 0, no chain header is passed.
- When bit 3 is set to 0 and bit 1 is set to 1, the original chain header is passed.

- When bit 3 is set to 1 and bit 1 is set to 0, an extended chain header is passed.
- When bit 3 is set to 1 and bit 1 is set to 1, an invalid condition exists, and an error return code is returned.

#### **(AL) = 05H—Send Immediate Command to Device**

This function sends an immediate command to the specified SCSI device. (Refer to the technical references for the SCSI adapter and the device for details on the immediate commands and other programming information.)

(BX) - Word 0 of the immediate command

(CX) - Word 1 of the immediate command

(DX) - Device key

On Return:

(AH) - Return code

(AL) - Error status (see "Return Codes" on page 2-IN4B-10 for a definition of this error status)

CF = 0 - Return code is 0

= 1 - Return code is non 0

#### **(AL) = 06H—Reset Adapter**

This function issues a hardware reset to the adapter for the specified device. This causes the card to perform power-on diagnostic routines to determine whether various components on the adapter are functioning properly. All device operations in progress on the adapter are terminated. After the reset, BIOS restores the device assignments, device time-out values, and the device-cache status.

**Note:** This function should be used only as part of an error-recovery procedure. See "Return Codes" on page 2-IN4B-10 to determine when this function should be issued.

(DX) - Device key

On Return:

(AH) - Return code

(AL) - Error status (see "Return Codes" on page 2-IN4B-10 for a definition of this error status)

CF = 0 - Return code is 0

= 1 - Return code is non 0

### **(AL) = 07H—Set Time-Out Value**

This function specifies the maximum time that the adapter will wait for a device to complete a command. The default time-out value for each device is 45 seconds. The time-out value is specified in bits 6 to 0 of the CL register. Bit 7 determines whether the time is in seconds or in minutes.

If bits 6 to 0 are set to 0, BIOS programs the adapter not to time out any commands to the device. On subsequent commands to the device, BIOS waits up to 127 minutes for the operation to be completed before posting an error. If more time is required, the controlling program must chain into Interrupt 15H, screen the Wait (AX=9000H) and Post (AX=9100H) calls, and provide the correct wait time before allowing BIOS to post an error.

(CL) - Time-out value

Bit 7 = 0 - Time-out count, in seconds

= 1 - Time-out count, in minutes

Bits 6 to 0 - Time-out count

(DX) - Device key

On Return:

(AH) - Return code

(AL) - Error status (see "Return Codes" on page 2-IN4B-10 for a definition of this error status)

CF = 0 - Return code is 0

= 1 - Return code is non 0

### **(AL) = 08H to FFH—Reserved**

## Return Codes

The following figure lists the return codes and their meanings. Return codes are returned in the AH register.

Value	Description
00H	Operation Successfully Completed
01H	Invalid BIOS Function or Parameter
02H	Device Not Allocated
03H	Device Already Allocated
04H	Function Not Allowed (see note 1)
05H	Request Refused, Adapter Reset Required
06H	Request Ended In Error (see note 2)
07H	Adapter Reset Failed (see note 2)
08H	Adapter Reconfiguration Failed after Reset
09H	Adapter Time-Out Error
0AH	Adapter in Use (see note 3)
0BH	Extended Header Required to Complete the Request (see note 4)

**Figure 2-31. Return Codes – SCSI Devices**

### Notes:

1. This code is returned when the Set Feature Control immediate command is issued. Instead, the device driver must use the Set Time-Out Value function ((AL) = 07H).
2. The following defines the error status value in (AL):

(AL) - Error status

Bit 7 = 1 - Reset adapter before attempting the next command

Bit 6 = 1 - Request command-complete status to get additional error-status data

Bit 5 - Reserved

Bit 4 = 1 - Bits 3 to 0 contain interrupt ID of interrupt status register

Bits 3 to 0 - Interrupt ID of interrupt status register

3. This code is returned when the adapter is in use. The device driver should retry the operation at a later time.
4. In an operating-system DMA-services environment, this error occurs when:
  - No chain header (original or extended) was passed and the caller's termination status block must be filled in, or
  - No extended chain header was passed and the caller requested a scatter/gather operation on hardware that does not support scatter/gather.



## Programming Considerations

- BIOS tests the device key parameter and the device index parameter to protect the integrity of the system. Any calls that are directed to BIOS for SCSI devices and have an unrecognized function value generate a return code of 01H (Invalid BIOS Function or Parameter).
- SCSI devices are controlled by immediate commands or subsystem control blocks (SCBs) that are issued to the device. These commands contain a function number and other necessary parameters.

If the device driver specifies an SCB command, the device performs the command pointed to by (ES:BX).

If the device driver specifies an immediate command, the device performs the 32-bit command that is contained in (CX,BX) (CX=MSW; BX=LSW).

The device driver must ensure that subsystem control blocks and any data buffers or termination status block areas that are needed for command processing are locked into memory. The device driver must also ensure that the correct physical addresses are used.

- Before any of the BIOS for SCSI Devices functions are invoked, the device driver must test bit 1 of address hex 40:7B. If this bit is set to 1, the SCSI services are available, and the bit no longer has to be tested.
- The AX register and the carry flags are used to return the status of the operation. The carry flag signals the status of each operation on return from BIOS. If the carry flag is set to 0, the operation was successful, and (AH) is set to 0. If the carry flag is set to 1, the operation was not successful, and (AH) contains a return code as defined in "Return Codes" on page 2-IN4B-10. If more information is needed, use the Request Sense command.

## Programming Example

The following is an example of a procedure that uses these BIOS routines. The scenario describes a peripheral-type device driver that does not have information about how many devices of that type are attached.

**1. Determine how many devices of that type are attached:**

Get Device Count function

(DL) = 81H - Specifies removable-media tape drive

On Return:

CF = 0

(CL) = 03H - Three devices

**2. Get the device key for the specific drive. The device key is unique for each drive in the system:**

Allocate Device function

(DL) = 81H

(DH) = 01H - First tape drive

On Return:

CF = 0

(DX) = Device key for first tape drive in the system

**3. Send a Device Inquiry command using BIOS routines with (DX) = device key, and check the device-dependent information to determine whether it is the correct device. If it is not:**

Deallocate Device function

(DX) = Device key

Allocate Device function

(DL) = 81H

(DH) = 02H - Next tape

**Send Device SCB Chaining**

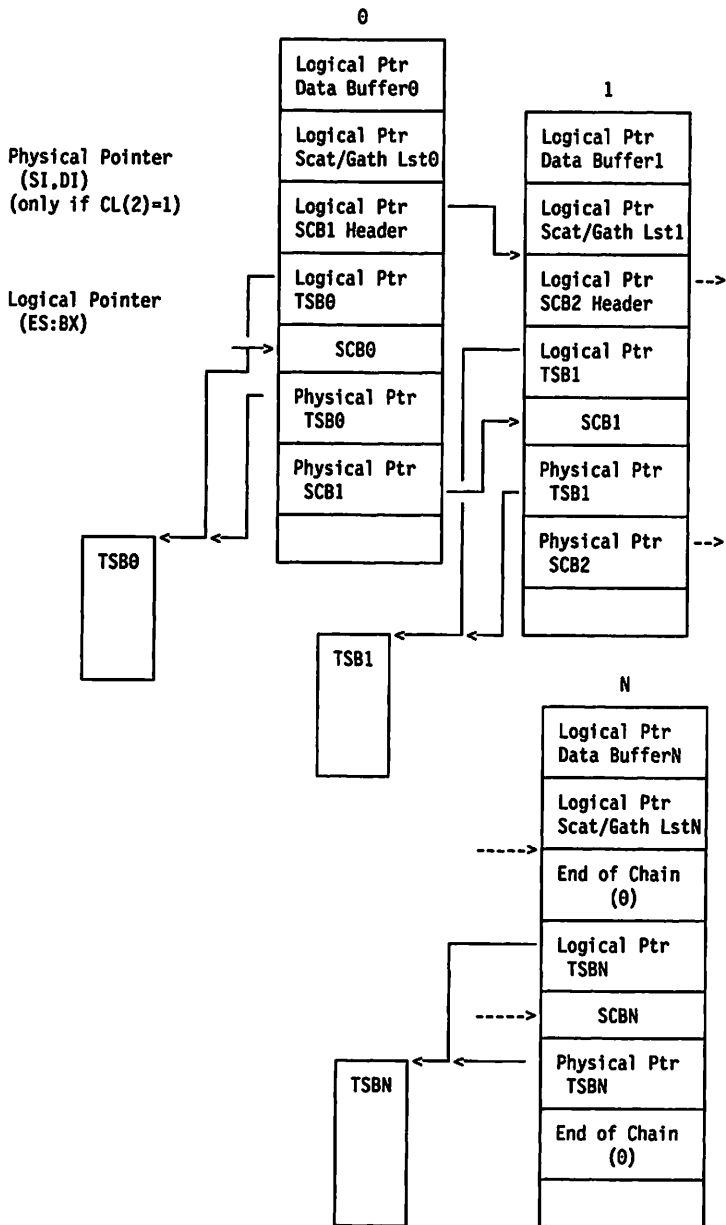
**Before chaining subsystem control blocks:**

- When bit 3 of the CL register is set to 1, the extended SCB chain header is being used.
- (ES:BX) is the logical pointer to the first SCB. This pointer must always exist. It points to the SCB, not to the SCB chain header.
- (SI,DI) is the 32-bit physical pointer to the first SCB. Bit 2 of the CL register is set to 1. If the SCB is not in linear-equals-physical memory, this information must be provided.

- The following describes the data structures that must be used when SCBs are chained:

Size	Offset	Description
Word	-10H	Reserved
DWord	-0CH	Logical-pointer data buffer
Word	-0AH	Reserved
Word	-08H	Reserved
DWord	-04H	Logical pointer to scatter/gather list (set to 0 if no scatter/gather list)
Word	-02H	Reserved
Word	00H	Reserved
DWord	02H	Logical pointer to next SCB header in chain or chain-ending indicator (0)
Word	06H	Reserved
Word	08H	Reserved
DWord	0AH	Logical pointer to TSB that is associated with this SCB
Word	0EH	Reserved
	10H	Beginning of SCB

## BIOS Send Device SCB Chain Example



## **(AH) = 81H—Operating-System DMA Services**

This interface is provided by the operating system, not by BIOS or ABIOS. It enables communication of physical address information between the operating system and other programs. This interface provides a means for:

- Identifying the version and functional level of the support that is provided
- Identifying the physical address of an I/O buffer
- Determining whether the memory for an I/O buffer is a contiguous block
- Locking and unlocking the I/O buffer into a specific memory space
- Remapping memory to put the I/O buffer into a contiguous memory space
- Allocating a contiguous block of memory for I/O buffers
- Copying data to or from the allocated buffers.

The following is a summary of the Operating-System DMA Services functions.

(AH) = 00H to 001H – Reserved
(AH) = 02H – Return DMA-Service Parameters (page 2-IN4B-20)
(AH) = 03H – Lock I/O Buffer (page 2-IN4B-20)
(AH) = 04H – Unlock I/O Buffer (page 2-IN4B-22)
(AH) = 05H – Scatter/Gather Lock Buffer (page 2-IN4B-23)
(AH) = 06H – Scatter/Gather Unlock Buffer (page 2-IN4B-25)
(AH) = 07H – Request Operating-System Buffer (page 2-IN4B-9)
(AH) = 08H – Release Operating-System Buffer (page 2-IN4B-26)
(AH) = 09H – Copy Data into Operating-System Buffer (page 2-IN4B-27)
(AH) = 0AH – Copy from Operating-System Buffer (page 2-IN4B-27)
(AH) = 0BH – Disable DMA Translation (page 2-IN4B-28)
(AH) = 0CH – Enable DMA Translation (page 2-IN4B-28)
(AH) = 0DH to FFH – Reserved

*Figure 2-32. Operating-System DMA Services Functions*

Micro Channel bus-master adapters can access memory locations without assistance from the operating system and without using the system microprocessor or DMA controller.

When the system microprocessor is operating in the protected (virtual address) mode, the device drivers for these adapters communicate with the operating system to exchange address information. In this mode, the physical address of the data buffers that are used in I/O operations is not available to BIOS or device-driver routines that initiate the I/O operations.

## **Operating-System Requirements**

Any operating system that supports operating-system DMA services must provide the following minimum requirements:

- Functions hex 02 through hex 0C describe the minimum subset of DMA services that must be implemented by the provider.
- The operating system must support either I/O buffer allocation or automatic page remapping, but it can provide both services. If I/O buffer allocation is supported, the minimum size of the buffer is 16KB.
- During initialization of an operating system that provides DMA services, the device drivers must remain at the previously-allocated memory addresses. Operating systems that provide DMA services can implement an interface that allows for the device drivers to be relocated. However, all in-process I/O transactions must be completed before the device drivers are relocated.
- The operating system must indicate that these DMA services are provided in the BIOS data area by setting bit 5 at address hex 40:7B to 1.
- BIOS locks and unlocks the extended BIOS data area as required. When DMA services are provided (bit 5 at address hex 40:7B is set to 1), the provider must ensure that requests to lock the extended BIOS data area are successful. When the extended BIOS data area is locked, it is contiguous, and the physical address is available in the Physical Address field of the DMA descriptor structure (DDS).
- Interrupt 15H, Return Extended BIOS Data Area Segment Address function ((AH)=C1H) returns the address segment for the extended BIOS data area. The first byte (offset hex 00) is the length of the extended BIOS data area, in KB. The extended BIOS data area has an assumed offset of hex 0. Interrupt 15H, Return System Configuration Parameters function ((AH)=C0H) indicates whether the extended BIOS data area has been allocated.

## BIOS Data Area

The following flags are located at address hex 40:7B in the BIOS data area. These flags indicate the services that are supported through Interrupt 4BH.

Address hex 40:7B

Bits 7 to 6 - Reserved

Bit 5 - Operating-system DMA services indicator

= 0 - Not supported; all memory is mapped linear=physical

= 1 - Supported

Bit 4 - Reserved

Bit 3 - Interrupt-4BH-intercepted indicator

= 0 - Interrupt vector is not intercepted

= 1 - Interrupt vector is intercepted

Bit 2 - Reserved

Bit 1 - Generic SCSI CBIOS services support

= 0 - Interrupt 4BH does not support SCSI

= 1 - Interrupt 4BH supports SCSI

Bit 0 - Reserved

If bit 5 is set to 1, DMA services are available. If bit 5 is set to 0, DMA services are not available.

Providers of the operating-system DMA services must set bit 3 when hooking into Interrupt 4BH.

If bit 3 is set to 1, a program that provides services through Interrupt 4BH must link into the interrupt chain by saving the interrupt vector for Interrupt 4BH and replacing the old vector with the pointer to the start of the code for the program. Then, if the program does not provide the requested services, it can pass the call to the next link in the chain.

For Personal System/2 systems with Micro Channel architecture, if bit 3 is set to 0, the following test must be performed:

If bit 1 of 40:7B is set to 0,

— or —

the contents of the Interrupt 4BH vector are 0:0,

— or —

the Interrupt 4BH vector contains a segment value of hex E000 or hex F000,

the interrupt vector must not be chained.

Else, the interrupt vector must be chained.

If this test indicates that the interrupt vector must not be chained, the program puts its pointer into the vector and sets bit 3 to 1; however, if it does not provide the requested services, the program must return status that indicates that the call is not supported ((AH)=86H and CF=1).

### **Programming Considerations**

- Before attempting to physically address memory, device drivers and other programs that use the operating-system DMA services must determine whether physical-address services are available. At initialization, the device driver tests bit 5 (physical-address services indicator) at address hex 40:7B to make this determination.

If bit 5 is set to 1, the services are available, and the device driver must use the services to lock the required buffers into memory and to get the physical addresses of the buffers. Any locked buffers must be unlocked after the I/O operation is complete. This bit is retested before each operation that accesses memory.

If bit 5 is set to 0, the services are not available, and the device driver requests a block of memory to use as buffer space. After the buffer space has been allocated, that space remains locked into memory as long as the driver remains installed. The device driver does not retest bit 5 for accesses to buffers that have been allocated in this manner. However, if the device driver receives an address that is not within its boundaries, the device driver must retest bit 5 and use the physical-address services, if they are available.

- With the exception of the AX register, the contents of all registers and the values in the DMA descriptor structure must be preserved, unless otherwise noted. If the carry flag is set to 1, the AL register contains an error code. If the carry flag is set to 0, the AH and AL registers are undefined, unless otherwise noted.



## The DMA Descriptor Structure (DDS)

The DMA descriptor structure (DDS) is used to pass parameters between the operating system and the calling program. The DDS can be constructed in the memory of the calling program or on the stack. Reserved fields in the DDS must be set to 0.

The format of the DMA descriptor structure is as follows:

Size	Offset	Description
DWord	00H	Buffer size (In bytes)
DWord	04H	Offset This is a 32-bit offset. If a 16-bit offset is specified, the calling program must set the word at offset hex 06 to 0. If the Segment/Selector field is set to 0, the offset is a 32-bit linear offset.
Word	08H	Segment/selector This is the segment or selector of the buffer. If this field is set to 0, the Offset field contains the 32-bit linear address of the buffer.
Word	0AH	Buffer ID This field is filled in by the provider of DMA services. When this field is set to 0, a buffer was not allocated.
Word	0CH	Physical address This is the 32-bit value of the physical address that can be used to program the DMA controller or Bus-Master Address register. This field is filled in by the provider of DMA services.

### (AL) = 00H to 01H—Reserved

On Return:

(AL) - Error code

= 0FH - Function not supported

(CF) = 1 - Operation failed

## **(AL) = 02H—Return DMA-Service Parameters**

This function returns the version level of DMA services, the operating-system identification, the DMA-service revision level, hardware address restrictions, the maximum buffer size that is available from the operating system, and the automatic memory-remapping-support indicator.

### **(DX) - Flags**

All bits are reserved and must be set to 0

### **On Return:**

(AH) - Major specification version number (binary)

(AL) - Minor specification version number (binary)

(BX) - Product number

(CX) - Product revision number

(SI:DI) - Maximum single operating-system buffer size that can be requested from the operating system (in bytes). A value of 0 indicates that no operating-system buffer is available or supported.

### **(DX) - Flags**

Bit 0 - Memory address space size

= 0 - Memory addresses greater than 1MB exist

= 1 - All memory addresses are less than 1MB

Bit 1 - Physical buffer/remap buffer location

= 0 - Physical buffer/remap buffer can be above 1MB

= 1 - Physical buffer/remap buffer is below 1MB

Bit 2 - Automatic page remapping support

= 0 Automatic page remapping is not supported

= 1 Automatic page remapping is supported

Bit 3 - Memory-paging indicator

= 0 - Memory is not physically contiguous

= 1 - All memory is physically contiguous

Bits 4 to 15 - Reserved (set to 0)

CF = 0 - Operation successfully completed

## **(AL) = 03H—Lock I/O Buffer**

This function determines whether the I/O buffer for the calling program is in contiguous physical memory. If it is, the physical address of the I/O buffer is returned in the DMA descriptor structure (DDS), and the buffer is locked in memory.

A locked buffer must be unlocked after the I/O operation is completed, either successfully or unsuccessfully.

Automatic page remapping, if supported, can be enabled or disabled. If automatic page remapping is enabled, the operating system moves the buffer pages to force the I/O buffer into contiguous physical memory. Automatic remapping might reduce performance because of the movement of data in memory.

An option is provided to allocate a buffer in the memory of the operating system if the buffer for the calling program is not in contiguous memory. The calling program can also specify that the data is to be copied from the buffer of the calling program into the allocated buffer. The physical address that is returned in the DDS points to the allocated buffer that contains the data for the calling program.

**(DX) - Flags**

**Bit 0 - Reserved**

**Bit 1 - Automatic data copy indicator**

= 0 - Do not copy data into operating-system buffer

= 1 - Copy data into operating-system buffer

(Ignored if bit 2 = 1)

**Bit 2 - Automatic buffer-allocation indicator**

= 0 - Allocate buffer

= 1 - Do not allocate buffer

**Bit 3 - Automatic remapping request indicator**

= 0 - Automatic remapping requested

= 1 - Do not remap memory

**Bit 4 - 64KB I/O buffer boundary restriction indicator**

= 0 - No 64KB physical address boundary restrictions

= 1 - Buffer must not cross 64KB physical address boundary

**Bit 5 - 128KB I/O buffer boundary restriction indicator**

= 0 - No 128KB physical address boundary restrictions

= 1 - Buffer must not cross 128KB physical address boundary

**Bits 6 to 15 - Reserved (set to 0)**

**(ES:DI) - Pointer to DMA descriptor structure (DDS)**

The calling program must fill in the Buffer Size field, the Offset field, and the Segment/Selector field in the DDS.

**On Return:**

**(AL) - Error code**

= 01H - Specified buffer is not in contiguous memory, or automatic remapping was not successful

= 02H - Buffer crossed a physical alignment boundary

= 03H - Unable to lock pages; insufficient memory (virtual-memory systems only)

= 05H - Requested buffer size too large

= 06H - Buffer currently in use

= 07H - Invalid memory buffer specified

= 10H - Reserved flag bits set in (DX)

**CF = 0 - Operation successfully completed.**

Buffer memory is locked.

The Physical Address field of the DDS contains the starting physical address of the buffer.

The Buffer ID field contains the ID of the allocated buffer, or the Buffer ID field contains 0 if no buffer was allocated.

**= 1 - Operation failed.**

Buffer memory is not locked

The Buffer Size field of the DDS contains the maximum contiguous length, in bytes.

## Notes:

1. If the DMA services return a physical address that is not compatible with the DMA controller or the bus-master addressing capabilities, the calling program can unlock the I/O buffer and request that the DMA services allocate a buffer from the free memory of the operating system (see (AL) = 07H on page 2-IN4B-25).
2. If an operating-system buffer is not available, the calling program can enable interrupts, and a wait loop can repeatedly attempt to allocate a buffer.
3. After the I/O operation has been initiated, a wait loop with interrupts enabled can be used to wait for the completion of the I/O operation. The buffer can be released on the hardware-interrupt level, enabling other tasks that are waiting for buffer space to continue.
4. This function can be called repeatedly.

### **(AL) = 04H—Unlock I/O Buffer**

This function unlocks a previously-locked I/O buffer.

#### **(DX) - Flags**

Bits 15 to 2 - Reserved (set to 0)

Bit 1 - Automatic data copy indicator

= 0 - Do not copy data

= 1 - Copy data from the operating-system buffer to a buffer of the calling program

Bit 0 - Reserved (set to 0)

**(ES:DI) - Pointer to DMA descriptor structure (DDS)**

The calling program must return a DDS with the buffer size, physical address, and buffer ID of a buffer that is currently locked.

#### **On Return:**

**(AL) - Error code**

= 08H - Memory was not locked

= 0AH - Invalid buffer ID

= 10H - Reserved flag bits set in (DX)

**CF = 0 - Operation successfully completed.**

Memory is unlocked; the buffer is no longer valid.

= 1 - Operation failed.

All memory remains locked

## **(AL) = 05H—Scatter/Gather Lock Buffer**

This function determines the physical addresses of the memory areas that make up the buffer of the calling program and locks each of these memory areas. The list of physical addresses and the size of each area are returned to the calling program in the DMA descriptor structure (DDS). The calling program can use these parameters to program bus masters that support the scatter/gather functions or to break up a bus-master or DMA-I/O operation into multiple operations.

### **(DX) - Flags**

**Bit 6 - Return physical address/page table entries**

= 0 - Return physical addresses

= 1 - Return page table entries

**Bit 7 - Allocate page indicator (only if bit 6 is set to 1)**

= 0 - All pages must be allocated and present

= 1 - Lock only those pages that are present.

All page-table entries must be returned in the DDS, whether they are present or not.

The calling program must determine the presence or absence of pages in memory from bit 0 of the page-table entry:

Bits 31 to 12 - Page-frame address if bit 0 = 1.

All set to 0 if bit 0 = 0.

Bits 11 to 1 - Reserved (set to 0)

**Bit 0 - Presence indicator**

= 0 - Page is absent from memory

= 1 - Page is present in memory

All other bits are reserved and must be set to 0.

**(ES:DI) - Pointer to scatter/gather DDS**

When bit 6 is set to 0, the DMA descriptor structure returns the physical addresses of the memory areas:

<b>Size</b>	<b>Offset</b>	<b>Description</b>
DWord	00H	Buffer size, in bytes
DWord	04H	Offset
Word	08H	Segment/selector
Word	0AH	Reserved
Word	0CH	Number of physical buffers available
Word	0EH	Number of physical buffers that are used
DWord	10H	Buffer 0 physical address
DWord	14H	Buffer 0 size, in bytes
DWord	18H	Buffer 1 physical address
DWord	1CH	Buffer 1 size, in bytes
.	.	.
.	.	.
.	.	.
DWord	10H + (n×8)	Buffer n physical address
DWord	14H + (n×8)	Buffer n size, in bytes

When bit 6 is set to 1, the DMA descriptor structure returns the page-table entries of the memory areas:

Size	Offset	Description
DWord	00H	Buffer size, in bytes
DWord	04H	Offset
Word	08H	Segment/selector
Word	0AH	Reserved
Word	0CH	Number of page-table entries available
Word	0EH	Number of page-table entries that are used
DWord	10H	Page-table entry 0
DWord	14H	Page-table entry 1
.	.	.
.	.	.
DWord	10H + (n x 4)	Page-table entry n

The calling program must fill in the Buffer Size field, the Segment/Selector field, the Offset field, and the Number of Physical Buffers/Page-Table Entries Available field.

The maximum number of physical buffers that are required can be computed as:

$$\frac{(\text{Linear Address AND } 0\text{FFFH}) + \text{Buffer Size in hexadecimal} + 0\text{FFFH}}{1000\text{H}}$$

On Return:

(AL) = Error code

03H = Unable to lock pages, insufficient memory

07H = Invalid memory buffer specified

09H = Number of physical buffers/pages is greater than the table length. Offset hex 0E of the DDS contains the number of entries that are actually required to represent the buffer of the calling program.

10H = Reserved flag bits set in (DX)

CF = 0 - Operation successfully completed.

Offset hex 0E of the DDS contains the number of physical buffers or page-table entries that are filled in.

If bit 6 of (DX) is set to 1, the low 12 bits of (BX) contain the offset in the first page to the start of the buffer.

The high 4 bits are set to 0.

= 1 - Operation failed.

Memory is not locked.

The Buffer Size field of the DDS contains the maximum length, in bytes, that can be locked and described in the DDS.

Either the list of physical addresses and their sizes or the page-table entries are undefined.

### **((AL) = 06H—Scatter/Gather Unlock Buffer**

This function unlocks a buffer that was locked by the Scatter/Gather Lock Buffer function ((AL) = 05H).

(DX) - Flags

All bits are reserved and must be set to 0.

(ES:DI) - Pointer to the extended DDS that was used when the Scatter/Gather Lock Buffer function was called

On Return:

(AL) - Error code

= 08H - Memory was not locked

= 10H - Reserved flag bits set in (DX)

CF = 0 - Operation successfully completed; memory is unlocked

= 1 - Operation failed

### **((AL) = 07H—Request Operating-System Buffer**

This function allocates and locks a buffer of physically-contiguous memory from the operating-system free memory. If requested, data is copied from the buffer of the calling program into the buffer that was allocated by the operating system.

The data must be copied to or from the operating-system buffer before the Release Operating-System Buffer function ((AL) = 08H) is called. The physical address of the buffer is invalid after the buffer is released.

(DX) - Flags

Bit 1 - Automatic data copy indicator

= 0 - Do not copy data

= 1 - Copy data into the operating-system buffer

All other bits are reserved and must be set to 0.

(ES:DI) - Pointer to DMA descriptor structure (DDS)

The calling program must fill in the Buffer Size field. If automatic copy is selected, the calling program must also fill in the Segment/Selector field and the Offset field to point to the data that is to be copied into the allocated buffer.

On Return:

(AL) - Error code

- = 04H - No buffer available
- = 05H - Buffer size too large
- = 06H - Buffer currently in use
- = 07H - Invalid memory buffer specified
- = 10H - Reserved flag bits set in (DX)

CF = 0 - Operation successfully completed

The Physical Address field of the DDS contains the starting physical address of the buffer.

The Buffer Size field of the DDS specifies the size of the buffer.

The Buffer ID field of the DDS contains the ID of the allocated buffer.

If automatic copy is selected, data is copied into the operating-system buffer.

= 1 - Operation failed

### Notes:

1. If an operating-system buffer is not available, the calling program can enable interrupts, and a wait loop can repeatedly attempt to allocate a buffer.
2. After the I/O operation has been initiated, a wait loop with interrupts enabled can be used to wait for the completion of the I/O operation. The buffer can be released on the hardware-interrupt level, enabling other tasks that are waiting for buffer space to continue.
3. This function can be called repeatedly.

### **(AL) = 08H—Release Operating-System Buffer**

This function releases an operating-system buffer that was previously allocated. The physical address is invalid after the buffer is released.

(DX) - Flags

Bit 1 - Automatic data copy indicator

= 0 - Do not copy data

= 1 - Copy data out of the operating-system buffer

All other bits are reserved and must be set to 0.

(ES:DI) - Pointer to DMA descriptor structure (DDS)

The calling program must return a pointer to a DDS that contains a Buffer ID field of the previously-allocated buffer.

On Return:

(AL) - Error code

= 0AH - Invalid buffer ID

= 10H - Reserved flag bits set in (DX)

CF = 0 - Operation successfully completed

= 1 - Operation failed



### **(AL) = 09H—Copy Data Into Operating-System Buffer**

This function copies data from the buffer of the calling program into the buffer that has been allocated by the operating system to prepare for an I/O write operation.

(DX) - Flags

All bits are reserved and must be set to 0.

(ES:DI) - Pointer to DMA descriptor structure (DDS)

The calling program must fill in the Buffer ID field, the Buffer Size field, the Segment/Selector field, and the Offset field to specify the source address to copy from. The Buffer Size field of the DDS determines the number of bytes that are to be copied.

(BX:CX) - Starting offset in operating-system buffer to be copied

On Return:

(AL) - Error code

= 0AH - Invalid buffer ID

= 0BH - (Copy count + offset) is greater than the buffer size

= 10H - Reserved flag bits set in (DX)

CF = 0 - Operation successfully completed

= 1 - Operation failed

### **(AL) = 0AH—Copy from Operating-System Buffer**

This function copies data from the buffer that has been allocated by the operating system into the buffer of the calling program after an I/O read operation.

(DX) - Flags

All bits are reserved and must be set to 0.

(ES:DI) - Pointer to DMA descriptor structure (DDS)

The calling program must fill in the Buffer ID field, the Buffer Size field, the Segment/Selector field, and the Offset field to specify the destination address to copy to. The Buffer Size field of the DDS determines the number of bytes that are to be copied.

(BX:CX) - Starting offset in operating-system buffer to be copied

On Return:

- (AL) - Error code
  - = 0AH - Invalid buffer ID
  - = 0BH - (Copy count + offset) is greater than the buffer size
  - = 10H - Reserved flag bits set in (DX)
- CF = 0 - Operation successfully completed
- = 1 - Operation failed

### **(AL) = 0BH—Disable DMA Translation**

This function disables the operating-system function that traps the DMA I/O ports and remaps the buffer address to a physical address. BIOS and device drivers that use the DMA services to determine the physical address of I/O buffers must disable the automatic DMA translation by calling this function.

A disable count is maintained. Each disable call must have a corresponding enable call before automatic DMA translation is enabled.

(BX) - DMA channel number

(DX) - Flags

All bits are reserved and must be set to 0.

On Return:

- (AL) - Error code
  - = 0CH - Invalid DMA channel number
  - = 0DH - Disable-count overflow
  - = 10H - Reserved flag bits set in (DX)
- CF = 0 - Operation successfully completed
- = 1 - Operation failed

### **(AL) = 0CH—Enable DMA Translation**

This function enables automatic DMA translation.

A disable count is maintained. Each disable call must have a corresponding enable call before automatic DMA translation is enabled.

(BX) - DMA channel number

(DX) - Flags

All bits are reserved and must be set to 0.

On Return:

(AL) - Error code

= 0CH - Invalid DMA channel number

= 0EH - Disable-count underflow (was not previously disabled; count remains 0)

= 10H - Reserved flag bits set in (DX)

CF = 0 - Operation successfully completed

= 1 - Operation failed

ZF = 0 - Disable count is not equal to 0

= 1 - Disable count is equal to 0

### **(AL) = 0DH to FFH—Reserved**

All registers are reserved (set to 0).

On Return:

(AL) - Error code

= 0FH - Function not supported

CF = 1 - Operation failed

## Error Codes and Option Flags

The following figure lists the error codes and their meanings. Error codes are returned in the AL register.

Value	Description
01H	Specified buffer was not in contiguous memory
02H	Buffer crossed a physical alignment boundary
03H	Unable to lock pages
04H	No buffer available
05H	Buffer too large
06H	Buffer currently in use
07H	Invalid memory buffer specified
08H	Buffer was not locked
09H	Number of physical pages was greater than table length
0AH	Invalid buffer ID
0BH	Copy out of buffer range
0CH	Invalid DMA channel number
0DH	Disable-count overflow
0EH	Disable-count underflow
0FH	Function not supported
10H	Reserved flag bits set in (DX)

**Figure 2-33. Error Codes — Operating-System DMA Services**

The following figure lists the option flags. Option flags are returned in the DX register.

Bit 7	Lock and return presently-available pages
Bit 6	Return physical address/page table entries
Bit 5	128KB I/O-buffer boundary restriction indicator
Bit 4	64KB I/O-buffer boundary restriction indicator
Bit 3	Automatic remapping request indicator
Bit 2	Automatic buffer allocation indicator
Bit 1	Automatic data-copy indicator

**Figure 2-34. Option Flags — Operating-System DMA Services**

## Interrupt 70H—Real-Time Clock Interrupt

For AT, PC/XT Model 286, and Personal System/2 products except Model 25:

This interrupt handler services the periodic and alarm interrupt functions from the real-time clock.

**Periodic function** — When activated, the interrupt occurs approximately 1024 times per second. The interrupt handler decreases the doubleword microsecond counter by 976 microseconds (1/1024 of a second). When the counter becomes less than or equal to 0, bit 7 of the designated location is set to hex 80. For Interrupt 15H, Event Wait function ((AH)=83H), the designated location is provided by the user. For Interrupt 15H, Wait function ((AH)=86H), the designated location is bit 7 of BIOS data area hex 40:A0 (Wait Active Flag).

**Alarm function** — When activated, the interrupt occurs at the specified time, and a software Interrupt 4AH is issued. The user must point Interrupt 4AH to an alarm routine before setting Interrupt 1AH, Real-Time Clock Alarm function ((AH)=06H).

For all others, the real-time clock interrupt is not supported.

### Notes:

1. The PC Convertible supports the Periodic and Alarm Interrupt functions, but the real-time clock interrupt generates a nonmaskable interrupt, rather than invoking Interrupt 70H. Additionally, the PC Convertible uses the real-time-clock update ended interrupt function (one interrupt per second) when certain system profiles are enabled.
2. For Personal System/2 Model 30, the Periodic Interrupt function is not supported.

## Notes:



## Section 3. BIOS Data Areas

The BIOS data areas are allocated specifically as work areas for system BIOS and adapter BIOS. The BIOS routines use 256 bytes of memory from absolute address hex 400 to hex 4FF. A description of the BIOS data areas follows.

Address (Hex)	Function	Size
40:00	RS-232C communication line 1 port base address	Word
40:02	RS-232C communication line 2 port base address	Word
40:04	RS-232C communication line 3 port base address	Word
40:06	RS-232C communication line 4 port base address	Word
<b>Note:</b> The RS-232C communication line port base address fields can be initialized to 0 by the POST if the system configuration contains less than four serial ports. The POST never puts 0 in the RS-232C communication line port base address table between two valid RS-232C communication line port base addresses.		

**Figure 3-1. RS-232C Port Base Address Data Area**

Address (Hex)	Function	Size
40:08	Printer 1 port base address	Word
40:0A	Printer 2 port base address	Word
40:0C	Printer 3 port base address	Word
40:0E	Reserved	Word
Exception: 40:0E	Printer 4 port base address (PC, PC/XT, AT, and PC Convertible)	Word

**Note:** The printer port base address fields can be initialized to 0 by the POST if the system configuration contains less than four parallel ports. The POST never puts 0 in the printer port base address table between two valid printer port base addresses.

**Figure 3-2. Printer Port Base Address Data Area**

Address (Hex)	Function	Size
40:10	Installed hardware	Word
Bits 15, 14	Number of printer adapters	
Bit 13	Reserved	
Bit 12	Reserved	
Bits 11 to 9	Number of RS-232C adapters	
Bit 8	Reserved	
Bits 7, 6	Number of diskette drives (0-based)	
Bits 5, 4	Video mode type (values are binary)	
	= 00 - Reserved	
	= 01 - 40x25 color	
	= 10 - 80x25 color	
	= 11 - 80x25 monochrome	
Bit 3	Reserved	
Bit 2	Pointing device	
Bit 1	Math coprocessor	
Bit 0	IPL diskette	
Exceptions:		
Bit 13	Internal modem (for PC Convertible and Personal System/2 Model 55 LS)	
Bit 2	Reserved (for PC, PC/XT, AT, and PC Convertible)	

**Note:** Refer to Interrupt 11H for equipment return information.

**Figure 3-3. System Equipment Data Area**



<b>Address (Hex)</b>	<b>Function</b>	<b>Size</b>
40:12	Reserved	Byte
<b>Exceptions:</b>		
40:12	Power-on self-test status (for PC Convertible only)	Byte
	POST system flag (for Personal System/2 Model 25 only)	Byte
Bit 1	Real-time clock (RTC) installed	
Bit 0	Optional memory failed; memory remapped	

*Figure 3-4. Miscellaneous Data Area 1*

<b>Address (Hex)</b>	<b>Function</b>	<b>Size</b>
40:13	Memory size, in KB (range of 0KB to 640KB)	Word
40:15, 40:16	Reserved	Byte

*Figure 3-5. Memory Size Data Area*

<b>Address (Hex)</b>	<b>Function</b>	<b>Size</b>
<b>40:17</b>	<b>Keyboard control</b>	<b>Byte</b>
Bit 7	Insert locked	
Bit 6	Caps Lock locked	
Bit 5	Num Lock locked	
Bit 4	Scroll Lock locked	
Bit 3	Alt key pressed	
Bit 2	Ctrl key pressed	
Bit 1	Left Shift key pressed	
Bit 0	Right Shift key pressed	
<b>Note:</b> For 101/102-key keyboard support, bits 2 and 3 are set if either or both the Alt key and the Ctrl key are pressed.		
<b>40:18</b>	<b>Keyboard control</b>	<b>Byte</b>
Bit 7	Insert key pressed	
Bit 6	Caps Lock key pressed	
Bit 5	Num Lock key pressed	
Bit 4	Scroll Lock key pressed	
Bit 3	Pause locked	
Bit 2	SysRq key pressed	
Bit 1	Left Alt key pressed	
Bit 0	Left Ctrl key pressed	
<b>40:19</b>	<b>Alternate keypad entry</b>	<b>Byte</b>
<b>40:1A</b>	<b>Keyboard buffer head pointer</b>	<b>Word</b>
<b>40:1C</b>	<b>Keyboard buffer tail pointer</b>	<b>Word</b>
<b>40:1E</b>	<b>Keyboard buffer</b>	<b>32 bytes</b>

**Figure 3-6. Keyboard Data Area 1**

Address (Hex)	Function	Size
40:3E	Recalibrate status	Byte
Bit 7	Interrupt flag	
Bit 6	Reserved	
Bit 5	Reserved	
Bit 4	Reserved	
Bit 3	Recalibrate drive 3	
Bit 2	Recalibrate drive 2	
Bit 1	Recalibrate drive 1	
Bit 0	Recalibrate drive 0	
40:3F	Motor status	Byte
Bit 7	Write/read operation	
Bit 6	Reserved	
Bits 5, 4	Diskette drive select status (values in binary)	
	= 00 - Diskette drive 0 selected	
	= 01 - Diskette drive 1 selected	
	= 10 - Diskette drive 2 selected	
	= 11 - Diskette drive 3 selected	
Bit 3	Diskette drive 3 motor-on status	
Bit 2	Diskette drive 2 motor-on status	
Bit 1	Diskette drive 1 motor-on status	
Bit 0	Diskette drive 0 motor-on status	
40:40	Motor off counter	Byte
40:41	Last diskette drive operation status	Byte
	= 00H - No error	
	= 01H - Invalid diskette drive parameter	
	= 02H - Address mark not found	
	= 03H - Write-protect error	
	= 04H - Requested sector not found	
	= 06H - Diskette change line active	
	= 08H - DMA overrun on operation	
	= 09H - Attempt to DMA across a 64KB boundary	
	= 0CH - Media type not found	
	= 10H - CRC error on diskette read	
	= 20H - General controller failure	
	= 30H - Drive does not support media sense	
	= 31H - No media in drive	
	= 32H - Drive does not support media type	
	= 40H - Seek operation failed	
	= 80H - Time-out	
	= AAH - Diskette drive not ready	

**Figure 3-7 (Part 1 of 2). Diskette Drive Data Area**

<b>Address (Hex)</b>	<b>Function</b>	<b>Size</b>
40:42	Diskette-drive controller status bytes	7 bytes

*Figure 3-7 (Part 2 of 2). Diskette Drive Data Area*

<b>Address (Hex)</b>	<b>Function</b>	<b>Size</b>
40:49	Display mode set	Byte
40:4A	Number of columns	Word
40:4C	Length of regen buffer, in bytes	Word
40:4E	Starting address in regen buffer	Word
40:50	Cursor position page 1	Word
40:52	Cursor position page 2	Word
40:54	Cursor position page 3	Word
40:56	Cursor position page 4	Word
40:58	Cursor position page 5	Word
40:5A	Cursor position page 6	Word
40:5C	Cursor position page 7	Word
40:5E	Cursor position page 8	Word
40:60	Cursor type	Word
40:62	Display page	Byte
40:63	Display controller base address	Word
40:65	Current setting of 3x8 register	Byte
40:66	Current setting of 3x9 register	Byte

*Figure 3-8. Video-Control Data Area 1*

<b>Address (Hex)</b>	<b>Function</b>	<b>Size</b>
40:67	Reserved	DWord
40:6B	Reserved	Byte
Exception: 40:67	Pointer to reset code upon system reset with memory preserved (for Personal System/2 products except Model 25 and Model 30) Reset flag at hex 40:72 = 4321H	DWord

**Figure 3-9. System Data Area 1**

<b>Address (Hex)</b>	<b>Function</b>	<b>Size</b>
40:6C	Timer counter	DWord
40:70	Timer overflow (if nonzero, the timer has counted past 24 hours.)	Byte

**Figure 3-10. System-Timer Data Area**

<b>Address (Hex)</b>	<b>Function</b>	<b>Size</b>
40:71 Bit 7	Break key state = 1 - Break key pressed = 0 - Break key not pressed	Byte
40:72	Reset flag = 1234H - Bypass memory test = 4321H - Preserve memory (for Personal System/2 products except Model 25 and Model 30) = 5678H - System suspended (for PC Convertible) = 9ABCH - Manufacturing test mode (for PC Convertible) = ABCDH - System POST loop mode (for PC Convertible)	Word

**Figure 3-11. System Data Area 2**

<b>Address (Hex)</b>	<b>Function</b>	<b>Size</b>
<b>40:74</b>	<b>Last fixed disk drive operation status</b> 00H = No error 01H = Invalid function request 02H = Address mark not found 03H = Write protect error 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 detected 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 = Time-out 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	<b>Byte</b>
<b>40:75</b>	<b>Number of fixed disk drives attached</b>	<b>Byte</b>
<b>40:76</b>	<b>Reserved</b>	<b>Byte</b>
<b>40:77</b>	<b>Reserved</b>	<b>Byte</b>
<b>Exceptions:</b>		
<b>40:74</b>	<b>Reserved (for devices using ESDI-type commands)</b>	<b>Byte</b>
<b>40:76</b>	<b>Fixed disk drive control (for PC/XT)</b>	<b>Byte</b>
<b>40:77</b>	<b>Fixed disk drive controller port (for PC/XT)</b>	<b>Byte</b>

**Figure 3-12. Fixed Disk Drive Data Area**

Address (Hex)	Function	Size
40:78	Printer 1 time-out value	Byte
40:79	Printer 2 time-out value	Byte
40:7A	Printer 3 time-out value	Byte
40:7B	Generic SCSI CBIOS and bus-master filter flags	
Bits 7, 6	Reserved	
Bit 5	Operating-system DMA services indicator = 0 - Not supported; all memory is mapped linear = physical = 1 - Supported	
Bit 4	Reserved	
Bit 3	Interrupt-4BH-intercepted indicator = 0 - Interrupt vector is not intercepted = 1 - Interrupt vector is intercepted	
Bit 2	Reserved	
Bit 1	Generic SCSI CBIOS services support = 0 - Interrupt 4BH does not support SCSI = 1 - Interrupt 4BH supports SCSI	
Bit 0	Reserved	
Exception: 40:7B	Printer 4 time-out value (for PC, PC/XT, AT, and Personal System/2 Model 25 and Model 30)	Byte

**Figure 3-13. Printer Time-Out Value Data Area**

Address (Hex)	Function	Size
40:7C	RS-232C communication line 1 time-out value	Byte
40:7D	RS-232C communication line 2 time-out value	Byte
40:7E	RS-232C communication line 3 time-out value	Byte
40:7F	RS-232C communication line 4 time-out value	Byte

**Figure 3-14. RS-232C Time-Out Value Data Area**

<b>Address (Hex)</b>	<b>Function</b>	<b>Size</b>
40:80	Keyboard buffer start offset pointer	Word
40:82	Keyboard buffer end offset pointer	Word

*Figure 3-15. Keyboard Data Area 2*

<b>Address (Hex)</b>	<b>Function</b>	<b>Size</b>
40:84	Number of rows on the screen (minus 1)	Byte
40:85	Character height (bytes/character)	Word
40:87	Video control states	Byte
40:88	Video control states	Byte
40:89	Reserved	Byte
40:8A	Reserved	Byte

*Figure 3-16. Video-Control Data Area 2*



Address (Hex)	Function	Size
40:8B	Media control	Byte
Bits 7, 6	Last diskette drive data rate selected (values in binary) = 00 - 500KB per second = 01 - 300KB per second = 10 - 250KB per second = 11 - 1MB per second	
Bits 5, 4	Last diskette drive step rate selected = 00 - 0CH = 01 - 0DH = 10 - 0EH = 11 - 0AH	
Bit 3	Reserved	
Bit 2	Reserved	
Bit 1	Reserved	
Bit 0	Reserved	
40:8C	Fixed disk drive controller status	Byte
40:8D	Fixed disk drive controller error status	Byte
40:8E	Fixed disk drive interrupt control	Byte
40:8F	Reserved	Byte

**Figure 3-17 (Part 1 of 5). Diskette Drive/Fixed Disk Drive Control Data Area**

Address (Hex)	Function	Size
40:90	Drive 0 media state	Byte
Bits 7, 6	Data transfer rate (values in binary) = 00 - 500KB per second = 01 - 300KB per second = 10 - 250KB per second = 11 - 1MB per second	
Bit 5	Double stepping = 1 - Double stepping required = 0 - Double stepping not required	
Bit 4	Media/drive established = 1 - Media/drive has been established = 0 - Media/drive has not been established	
Bit 3	4MB media capability = 1 - Drive is capable of supporting 4MB media = 0 - Drive is not capable of supporting 4MB media	
Bits 2 to 0	Drive/media state (values in binary) = 000 - 360KB diskette in 360KB drive is not established = 001 - 360KB diskette in 1.2MB drive is not established = 010 - 1.2MB diskette in 1.2MB drive is not established = 011 - 360KB diskette in 360KB drive is established = 100 - 360KB diskette in 1.2MB drive is established = 101 - 1.2MB diskette in 1.2MB drive is established = 110 - Reserved = 111 - None of the above	

**Figure 3-17 (Part 2 of 5). Diskette Drive/Fixed Disk Drive Control Data Area**

Address (Hex)	Function	Size
40:91	Drive 1 media state	Byte
Bits 7, 6	Data transfer rate (values in binary) = 00 - 500KB per second = 01 - 300KB per second = 10 - 250KB per second = 11 - 1MB per second	
Bit 5	Double stepping = 1 - Double stepping required = 0 - Double stepping not required	
Bit 4	Media/drive established = 1 - Media/drive has been established = 0 - Media/drive has not been established	
Bit 3	4MB media capability = 1 - Drive is capable of supporting 4MB media = 0 - Drive is not capable of supporting 4MB media	
Bits 2 to 0	Drive/media state (values in binary) = 000 - 360KB diskette in 360KB drive is not established = 001 - 360KB diskette in 1.2MB drive is not established = 010 - 1.2MB diskette in 1.2MB drive is not established = 011 - 360KB diskette in 360KB drive is established = 100 - 360KB diskette in 1.2MB drive is established = 101 - 1.2MB diskette in 1.2MB drive is established = 110 - Reserved = 111 - None of the above	

**Figure 3-17 (Part 3 of 5). Diskette Drive/Fixed Disk Drive Control Data Area**

Address (Hex)	Function	Size
40:92	Drive 2 media state	Byte
Bits 7, 6	Data transfer rate (values in binary) = 00 - 500KB per second = 01 - 300KB per second = 10 - 250KB per second = 11 - 1MB per second	
Bit 5	Double stepping = 1 - Double stepping required = 0 - Double stepping not required	
Bit 4	Media/drive established = 1 - Media/drive has been established = 0 - Media/drive has not been established	
Bit 3	4MB media capability = 1 - Drive is capable of supporting 4MB media = 0 - Drive is not capable of supporting 4MB media	
Bit 2	Multiple data rate capability determined = 1 - Multiple data rate capability is determined = 0 - Multiple data rate capability is not determined	
Bit 1	Multiple data rate capability = 1 - Multiple data rate capability = 0 - No multiple data rate capability	
Bit 0	80-track capability (drive 2) = 1 - Drive 2, 80 tracks = 0 - Drive 2, 40 tracks	

**Figure 3-17 (Part 4 of 5). Diskette Drive/Fixed Disk Drive Control Data Area**

<b>Address (Hex)</b>	<b>Function</b>	<b>Size</b>
<b>40:93</b>	<b>Drive 3 media state</b>	<b>Byte</b>
Bits 7, 6	Data transfer rate (values in binary) = 00 - 500KB per second = 01 - 300KB per second = 10 - 250KB per second = 11 - 1MB per second	
Bit 5	Double stepping = 1 - Double stepping required = 0 - Double stepping not required	
Bit 4	Media/drive established = 1 - Media/drive has been established = 0 - Media/drive has not been established	
Bit 3	4MB media capability = 1 - Drive is capable of supporting 4MB media = 0 - Drive is not capable of supporting 4MB media	
Bit 2	Multiple data rate capability determined = 1 - Multiple data rate capability is determined = 0 - Multiple data rate capability is not determined	
Bit 1	Multiple data rate capability = 1 - Multiple data rate capability = 0 - No multiple data rate capability	
Bit 0	80-track capability (drive 3) = 1 - Drive 3, 80 tracks = 0 - Drive 3, 40 tracks	
<b>40:94</b>	<b>Drive 0 current cylinder</b>	<b>Byte</b>
<b>40:95</b>	<b>Drive 1 current cylinder</b>	<b>Byte</b>
<b>Exception:</b>		
<b>40:8B to 40:95</b>	<b>Reserved (for PC, PCjr, PC/XT BIOS dated 11/8/82, and PC Convertible)</b>	<b>Byte</b>

**Figure 3-17 (Part 5 of 5). Diskette Drive/Fixed Disk Drive Control Data Area**

<b>Address (Hex)</b>	<b>Function</b>	<b>Size</b>
<b>40:96</b>	<b>Keyboard mode state and type flags</b>	<b>Byte</b>
Bit 7	Read ID in progress	
Bit 6	Last character was first ID character	
Bit 5	Force num lock if read ID and KBX	
Bit 4	101/102-key keyboard installed	
Bit 3	Right Alt key pressed	
Bit 2	Right Ctrl key pressed	
Bit 1	Last code was E0 hidden code	
Bit 0	Last code was E1 hidden code	
<b>40:97</b>	<b>Keyboard LED flags</b>	<b>Byte</b>
Bit 7	Keyboard transmit error flag	
Bit 6	Mode indicator update	
Bit 5	Resend receive flag	
Bit 4	Acknowledgment received	
Bit 3	Reserved (must be set to 0)	
Bits 2 to 0	Keyboard LED state bits	

**Figure 3-18. Keyboard Data Area 3**

<b>Address (Hex)</b>	<b>Function</b>	<b>Size</b>
<b>40:98</b>	<b>Offset address to user wait complete flag</b>	<b>Word</b>
<b>40:9A</b>	<b>Segment address to user wait complete flag</b>	<b>Word</b>
<b>40:9C</b>	<b>User wait count - low word (in microseconds)</b>	<b>Word</b>
<b>40:9E</b>	<b>User wait count - high word (in microseconds)</b>	<b>Word</b>
<b>40:A0</b>	<b>Wait active flag</b>	<b>Byte</b>
Bit 7	Wait time elapsed and Post	
Bits 6 to 1	Reserved	
Bit 0	Interrupt 15H, Wait function ((AH) = 86H) has occurred	
<b>40:A1 to 40:A7</b>	<b>Reserved</b>	<b>Byte</b>

**Figure 3-19. Real-Time Clock Data Area**

For Personal System/2 products and systems with EGA capability, the save pointer table contains pointers that define specific dynamic overrides for Interrupt 10H, Set Mode function ((AH)=00H).

Address (Hex)	Function	Size
40:A8	Pointer to video parameters and overrides	DWord
DWord 1	Video parameter table pointer Initialized to the BIOS video parameter table; this value must contain a valid pointer.	
DWord 2	Dynamic save area pointer (for all systems except Personal System/2 Model 25 and Model 30) Initialized to hex 00:00, this value is optional. When nonzero, this value points to an area in RAM where certain dynamic values are saved. This area holds the 16 EGA palette register values plus the overscan value in bytes (0 16), respectively. A minimum of 256 bytes must be allocated for this area.	
DWord 3	Alphanumeric mode auxiliary character generator Initialized to hex 00:00, this value is optional. When nonzero, this value points to a table that is described as follows:	
	Bytes/character	Byte
	Block to be loaded	Byte
	= 0 - Normal operation	
	Count to be stored	Word
	= 256 - Normal operation	
	Character offset	Word
	= 0 - Normal operation	
	Pointer to a font table	DWord
	Displayable rows	Byte
	If 0FFH, the maximum calculated value is used. Otherwise, this value is used.	
	Consecutive bytes of mode values for this font description. The end of this stream is indicated by a byte code of 0FFH.	Byte
<b>Note:</b> Use of the DWord 3 pointer might cause unexpected cursor type operation. For an explanation of cursor type, see Interrupt 10H, Set Cursor Type function ((AH)=01H).		

Figure 3-20 (Part 1 of 2). Save Pointer Data Area

Address (Hex)	Function	Size
DWord 4	Graphics mode auxiliary character generator pointer Initialized to hex 00:00, this value is optional. When nonzero, this value points to a table that is described as follows: Displayable rows Bytes per character Pointer to a font table Consecutive bytes of mode values for this font description. The end of this stream is indicated by a byte code of 0FFH.	Byte Word DWord Byte
DWord 5	Secondary save pointer (for all systems except EGA and Personal System/2 Model 25 and Model 30) Initialized to the BIOS secondary save pointer; this value must contain a valid pointer.	
DWord 6	Reserved and set to hex 00:00	
DWord 7	Reserved and set to hex 00:00	

**Figure 3-20 (Part 2 of 2). Save Pointer Data Area**



Address (Hex)	Function	Size
DWord 1	Table length Initialized to the BIOS secondary save pointer table length	
DWord 2	Display combination code (DCC) table pointer Initialized to ROM DCC table. This value must exist. It points to a table described as follows:	
	Number of entries in table	Byte
	DCC table version number	Byte
	Maximum display type code	Byte
	Reserved	Byte
	00,00 - Entry 0 No displays	
	00,01 - Entry 1 Monochrome display and printer adapter (MDPA)	
	00,02 - Entry 2 Color/graphics monitor adapter (CGA)	
	02,01 - Entry 3 MDPA + CGA	
	00,04 - Entry 4 Enhanced graphics adapter (EGA)	
	04,01 - Entry 5 EGA + MDPA	
	00,05 - Entry 6 EGA with monochrome display (MEGA)	
	02,05 - Entry 7 MEGA + CGA	
	00,06 - Entry 8 Professional graphics controller (PGC)	
	01,06 - Entry 9 PGC + MDPA	
	05,06 - Entry 10 PGC + MEGA	
	00,08 - Entry 11 Video graphics array (VGA) based with color display (CVGA)	
	01,08 - Entry 12 CVGA + MDPA	
	00,07 - Entry 13 VGA based with monochrome display (MVGA)	
	02,07 - Entry 14 MVGA + CGA	
	02,06 - Entry 15 MVGA + PGC	

**Figure 3-21 (Part 1 of 2). Secondary Save Pointer Data Area**

Address (Hex)	Function	Size
DWord 3	<p>Second alphanumeric mode auxiliary character generator pointer            Initialized to hex 00:00, this value is optional. When nonzero, this value points to a table that is described as follows:</p> <p>Bytes/character <span style="float:right">Byte</span>            Block to be loaded <span style="float:right">Byte</span>            = 0 - Normal operation            Reserved <span style="float:right">Byte</span>            Pointer to a font table <span style="float:right">DWord</span>            Consecutive bytes of mode values for this font description. The end of this stream is indicated by a byte code of 0FFH. <span style="float:right">Byte</span></p>	
<p><b>Note:</b> Attribute bit 3 is used to switch between primary and secondary fonts. It might be desirable to use the user palette profile to define a palette of consistent colors independent of attribute bit 3.</p>		
DWord 4	<p>User palette profile table pointer            Initialized to hex 00:00, this value is optional. When nonzero, this value points to a table that is described as follows:</p> <p>Underlining flag <span style="float:right">Byte</span>            = 1 - On            = 0 - Ignore            = -1 - Off            = 0 - Normal operation            Reserved <span style="float:right">Byte</span>            Reserved <span style="float:right">Word</span>            Internal palette count (0 to 17) <span style="float:right">Word</span>            = 17 - Normal operation            Internal palette index (0 to 16) <span style="float:right">Word</span>            = 0 - Normal operation            Pointer to internal palette <span style="float:right">DWord</span>            External palette count (0 to 256) <span style="float:right">Word</span>            = 256 - Normal operation            External palette index (0 to 255) <span style="float:right">Word</span>            = 0 - Normal operation            Pointer to external palette <span style="float:right">DWord</span>            Consecutive bytes of mode values for this font description. The end of this stream is indicated by a byte code of 0FFH. <span style="float:right">Byte</span></p>	
DWord 5 to DWord 7	Reserved (set to hex 00:00)	

**Figure 3-21 (Part 2 of 2). Secondary Save Pointer Data Area**

<b>Address (Hex)</b>	<b>Function</b>	<b>Size</b>
40:AC to 40:FF	Reserved	Byte
50:00	Print screen status (Interrupt 05H status)	Byte

*Figure 3-22. Miscellaneous Data Area 2*

## **Extended BIOS Data Areas**

The extended BIOS data area is supported on Personal System/2 products only. POST allocates the highest possible  $n$ KB of memory below 640KB to be used as this data area. The word value at address hex 40:13 (memory size), which indicates the number of KB below the 640KB limit, is decreased by  $n$ . The first byte in the extended BIOS data area is initialized to the length of the allocated area, in KB.

To access the extended BIOS data area segment, call Interrupt 15H, Return Extended BIOS Data Area Segment Address function ((AH)=C1H). To determine whether an extended BIOS data area is allocated, call Interrupt 15H, Return System Configuration Parameters function ((AH)=C0H).



## Section 4. ROM Tables

The following ROM tables are used by BIOS to define the characteristics of the hardware devices that are supported by the system or adapter BIOS.

### Fixed Disk Drive Parameter Table

The fixed disk drive parameter table is defined as follows:

Offset	Length	Description
0	Word	Maximum number of cylinders
2	Byte	Maximum number of heads
3	Word	For PC/XT: Starting reduced write current cylinder For all others: Not used
5	Word	Starting write precompensation cylinder
7	Byte	For PC/XT: Maximum ECC data-burst length For all others: Not used
8	Byte	Control byte For PC/XT: Bit 7 - Disable disk-access retries Bit 6 - Disable ECC retries Bits 5 to 3 = 0 Bits 2 to 0 - Drive option For all others: Bit 7 - Disable retries — or — Bit 6 - Disable retries Bit 5 - Manufacturer's defect map present at maximum cylinders + 1 Bit 3 - More than eight heads Bits 2 to 0 - Reserved
9	Byte	For PC/XT: Standard time-out value For all others: Not used
10	Byte	For PC/XT: Time-out value for format drive For all others: Not used
11	Byte	For PC/XT: Time-out value for check drive For all others: Not used
12	Word	For PC/XT: Reserved For all others: Landing zone
14	Byte	For PC/XT: Reserved For all others: Number of sectors per track
15	Byte	Reserved

Figure 4-1. Fixed Disk Drive Parameter Table

For AT and Personal System/2 products, the following table lists the fixed disk drive parameters for the various fixed disk drive types. Values are decimal unless noted otherwise.

Type	Number of Cylinders	Number of Heads	Number Write Precompensation	Landing Zone	Defect Map	Number of Sectors
0	— No fixed disk drive installed —					
1	306	4	128	305	No	17
2	615	4	300	615	No	17
3	615	6	300	615	No	17
4	940	8	512	940	No	17
5	940	6	512	940	No	17
6	615	4	0FFFFH (None)	615	No	17
7	462	8	256	511	No	17
8	733	5	0FFFFH (None)	733	No	17
9	900	15	0FFFFH (None)	901	No	17
10	820	3	0FFFFH (None)	820	No	17
11	855	5	0FFFFH (None)	855	No	17
12	855	7	0FFFFH (None)	855	No	17
13	306	8	128	319	No	17
14	733	7	0FFFFH (None)	733	No	17
15	— Reserved —					
16	612	4	0 (All cylinders)	663	No	17
17	977	5	300	977	No	17
18	977	7	0FFFFH (None)	977	No	17
19	1024	7	512	1023	No	17
20	733	5	300	732	No	17
21	733	7	300	732	No	17
22	733	5	300	733	No	17
23	306	4	0 (All cylinders)	336	No	17
24	612	4	305	663	No	17
25	306	4	0FFFFH (None)	340	No	17
26	612	4	0FFFFH (None)	670	No	17
27	698	7	300	732	Yes	17
28	976	5	488	977	Yes	17
29	306	4	0 (All cylinders)	340	No	17
30	611	4	306	663	Yes	17
31	732	7	300	732	Yes	17
32	1023	5	0FFFFH (None)	1023	Yes	17
33	614	4	0FFFFH (None)	663	Yes	25

Types 34 – 255 are reserved.

**Figure 4-2. Fixed Disk Drive Parameters (for AT and Personal System/2 Products)**

**Notes:**

1. Software Interrupt 41H points to the entry in the table for drive 0. Software Interrupt 46H points to the entry in the table for drive 1.
2. AT BIOS dated 1/10/84 supports types 0 through 14.
3. AT BIOS dated 6/10/85 or 11/15/85 supports types 0 through 23.
4. PC/XT Model 286 supports types 0 through 24.
5. Personal System/2 products except Model 25 and Model 30 support types 0 through 32.
6. Personal System/2 Model 30 supports types 0 through 26.
7. For Personal System/2 Model 70 and Model 80 BIOS dated 10/7/87 and later and self-identifying drives, the fixed disk drive parameters in Figure 4-2 on page 4-2 and Figure 4-3 do not apply; software Interrupts 41H and 46H are reserved.
8. To obtain the fixed disk drive parameters, use Interrupt 13H, Read Drive Parameters function ((AH)=08H).

For Personal System/2 products except Model 25 and Model 30, the following fixed disk drive parameters apply:

Offset	Length	Value	Description
0	2 bytes	41	Length of fixed disk drive table
2	22 bytes	(ID)	ASCII string 'IBM HARDFILE TYPE xxx', where xxx is the type number, in ASCII
24	1 byte	yyy	Type number (in binary)
25	2 bytes	*	Maximum number of cylinders
27	1 byte	*	Maximum number of heads
28	2 bytes	0	Reserved
30	2 bytes	*	Start write precompensation cylinder
32	1 byte	0	Reserved
33	1 byte	*	Control byte
			Bit 7 or 6 - Disable retries
			Bit 5 - Defect map installed
			Bit 3 - More than eight heads (AT only)
34	3 bytes	0	Reserved
37	2 bytes	*	Landing zone
39	1 byte	*	Number of sectors per track
40	1 byte	0	Reserved

**Note:** This information is located at head 0, track 0, sector 2; it applies only to ST412 and ST506 drive types.

**Figure 4-3. Fixed Disk Drive Parameters (for Personal System/2 Products except Model 25 and Model 30)**

For PC/XT BIOS dated 11/10/82, the following fixed disk drive parameters apply:

Size	Value	Description
DW	306	Maximum cylinders
DB	2	Maximum heads
DW	306	Start reduced write current cylinder
DW	0	Start write precompensation cylinder
DB	0BH	Maximum ECC burst data length
DB	00H	Control byte
DB	0CH	Standard time-out
DB	0B4H	Time-out for format drive
DB	028H	Time-out for check drive
DB	0,0,0,0	Reserved

**Figure 4-4. Fixed Disk Drive Parameter Table 00 (for PC/XT BIOS Dated 11/10/82)**

Size	Value	Description
DW	375	Maximum cylinders
DB	8	Maximum heads
DW	375	Start reduced write current cylinder
DW	0	Start write precompensation cylinder
DB	0BH	Maximum ECC burst data length
DB	05H	Control byte
DB	0CH	Standard time-out
DB	0B4H	Time-out for format drive
DB	028H	Time-out for check drive
DB	0,0,0,0	Reserved

**Figure 4-5. Fixed Disk Drive Parameter Table 01 (for PC/XT BIOS Dated 11/10/82)**



Size	Value	Description
DW	306	Maximum cylinders
DB	6	Maximum heads
DW	128	Start reduced write current cylinder
DW	256	Start write precompensation cylinder
DB	0BH	Maximum ECC burst data length
DB	05H	Control byte
DB	0CH	Standard time-out
DB	0B4H	Time-out for format drive
DB	028H	Time-out for check drive
DB	0,0,0,0	Reserved

**Figure 4-6. Fixed Disk Drive Parameter Table 02 (for PC/XT BIOS Dated 11/10/82)**

Size	Value	Description
DW	306	Maximum cylinders
DB	4	Maximum heads
DW	306	Start reduced write current cylinder
DW	0	Start write precompensation cylinder
DB	0BH	Maximum ECC burst data length
DB	05H	Control byte
DB	0CH	Standard time-out
DB	0B4H	Time-out for format drive
DB	028H	Time-out for check drive
DB	0,0,0,0	Reserved

**Figure 4-7. Fixed Disk Drive Parameter Table 03 (for PC/XT BIOS Dated 11/10/82)**

**Note:** Software Interrupt 41H points to the beginning of the table. The switch settings on the adapter are used as an index into the table.

For PC/XT BIOS dated 1/8/86 and later, the following fixed disk drive parameters apply:

Size	Value	Description
DW	306	Maximum cylinders
DB	4	Maximum heads
DW	306	Start reduced write current cylinder
DW	0	Start write precompensation cylinder
DB	0BH	Maximum ECC burst data length
DB	05H	Control byte
DB	0CH	Standard time-out
DB	0B4H	Time-out for format drive
DB	028H	Time-out for check drive
DB	0,0,0,0	Reserved

*Figure 4-8. Fixed Disk Drive Parameter Table 00 – Type 1 (for PC/XT BIOS Dated 1/8/86)*

Size	Value	Description
DW	612	Maximum cylinders
DB	4	Maximum heads
DW	612	Start reduced write current cylinder
DW	0	Start write precompensation cylinder
DB	0BH	Maximum ECC burst data length
DB	05H	Control byte
DB	20H	Standard time-out
DB	0B4H	Time-out for format drive
DB	028H	Time-out for check drive
DB	0,0,0,0	Reserved

*Figure 4-9. Fixed Disk Drive Parameter Table 01 – Type 16 (for PC/XT BIOS Dated 1/8/86)*

Size	Value	Description
DW	615	Maximum cylinders
DB	4	Maximum heads
DW	615	Start reduced write current cylinder
DW	300	Start write precompensation cylinder
DB	0BH	Maximum ECC burst data length
DB	05H	Control byte
DB	18H	Standard time-out
DB	0B4H	Time-out for format drive
DB	028H	Time-out for check drive
DB	0,0,0,0	Reserved

**Figure 4-10. Fixed Disk Drive Parameter Table 02 – Type 2 (for PC/XT BIOS Dated 1/8/86)**

Size	Value	Description
DW	306	Maximum cylinders
DB	8	Maximum heads
DW	306	Start reduced write current cylinder
DW	128	Start write precompensation cylinder
DB	0BH	Maximum ECC burst data length
DB	05H	Control byte
DB	0CH	Standard time-out
DB	0B4H	Time-out for format drive
DB	028H	Time-out for check drive
DB	0,0,0,0	Reserved

**Figure 4-11. Fixed Disk Drive Parameter Table 03 – Type 13 (for PC/XT BIOS Dated 1/8/86)**

**Note:** Software Interrupt 41H points to the beginning of the table. The switch settings on the adapter are used as an index into the table.

---

## Diskette Drive Parameter Table

The diskette drive parameter table is defined as follows:

Offset	Length	Description
0	Byte	First specification byte
1	Byte	Second specification byte
2	Byte	Number of timer ticks to wait before turning the diskette drive motor off
3	Byte	Number of bytes per sector = 02H - 512 bytes per sector
4	Byte	Sectors per track
5	Byte	Gap length
6	Byte	Dtl (data length)
7	Byte	Gap length for format
8	Byte	Fill byte for format
9	Byte	Head settle time (in milliseconds)
10	Byte	Motor startup time (in $\frac{1}{8}$ -second units) (For example, 8 = 1-second wait)

*Figure 4-12. Diskette Drive Parameter Table*

**Note:** Software Interrupt 1EH points to the beginning of the diskette drive parameter table.

| If the caller changes the values of the head settle time (byte 9) and  
| the motor startup time (byte 10) to values that are inconsistent with  
| the diskette drive specifications, BIOS enforces the minimum values  
| for these parameters as specified for the diskette drive. The values  
| of these parameters can be increased to allow for correcting possible  
| future problems if some diskette drives require more than the  
| nominal values for these parameters.

---

## Section 5. Additional Information

Interrupt Sharing	5-3
Considerations	5-3
Interrupt Request (IRQ n) Reset	5-4
Interrupt-Sharing Software Requirements	5-4
Interrupt-Sharing Chaining Structure and Signature	5-6
ROM Considerations	5-7
Implementation Information	5-7
Adapter ROM	5-11
Video-Function Compatibility	5-13
Video-Presence Test	5-13
Video-Mode Switching	5-14
Multitasking Provisions	5-15
Application Guidelines	5-17
Math-Coprocessor Testing	5-17
Hardware Interrupts	5-17
Programming Considerations	5-18
BIOS and Operating-System Function Calls	5-19

**Notes:**



---

## Interrupt Sharing

This section defines an interrupt-sharing protocol that enables multiple hardware adapters to share a single interrupt-request line.

### Considerations

When implementing interrupt sharing, consider the following:

- The interrupt-sharing protocol is intended to run only in the real address mode. It is not intended to run in the protected (virtual address) mode.
- The interrupt-sharing protocol does not apply to the sharing of an interrupt level between an interrupt handler that is running in the real mode and an interrupt handler that is running in the protected mode.
- The interrupt-sharing protocol is not necessarily compatible with all operating systems.
- Interrupts must be disabled before control is passed to the next handler on the chain. Disabling of the interrupts enables the next handler to receive control as if a hardware interrupt had caused it to receive control.
- Interrupts must be disabled before the nonspecific End of Interrupt (EOI) instruction is issued. To ensure that the Return from Interrupt (IRET) instruction is executed, interrupts must not be reenabled in the interrupt handler. The flags are restored and the interrupts are reenabled before another interrupt is serviced, protecting the stack from excessive build-up.
- Each interrupt handler must have a routine that can be executed after power-on to disable its adapters' interrupts. Executing this routine and resetting the interrupt-sharing hardware ensures that adapters are deactivated if the user resets the system.
- Interrupt-handler implementations must store data in memory using Intel\*\* data format; that is, word hex 424B is stored as 4B42 in memory.

---

\*\* Intel is a trademark of Intel Corporation.

## **Interrupt Request (IRQ n) Reset**

The Micro Channel interrupt mechanism is level sensitive, whereas the PC-type I/O channel interrupt mechanism is edge sensitive. The level-sensitive Micro Channel mechanism simplifies the interrupt hardware that is needed for the adapters.

An interrupt request in the PC-type I/O channel is implicitly reset because of the edge-sensitive characteristic of the signal. In the Micro Channel architecture, because of the level-sensitive characteristic of the signal, an interrupt request must be explicitly reset by the bus-slave interrupt-handler software. This is not the case when the bus-slave hardware implicitly resets the interrupt request. The system timer is an example of a bus-slave device that implicitly resets an interrupt.

## **Interrupt-Sharing Software Requirements**

All interrupt-sharing software that is developed for Micro Channel bus slaves must reset the interrupt request. The interrupt-sharing chaining structure must be provided by all interrupt handlers. The 16-byte interrupt-sharing chaining structure must begin at the third byte from the entry point of the interrupt handler. Pointers and flags that are stored in the interrupt-sharing chaining structure must be stored in Intel data format (see "Interrupt-Sharing Chaining Structure and Signature" on page 5-6). These requirements are specified to support the portability of interrupt handlers across hardware operating environments.

The interrupt-handling software for all adapters that share an interrupt-request line must implement this interrupt-sharing software standard. Interrupt-sharing software that operates in a multitasking environment must support the linking of a task's interrupt handler to a chain of interrupt handlers; the sharing of the interrupt level while that task is active; and the unlinking of the interrupt handler from the chain when the task is complete.

To link an interrupt handler, the interrupt handler of a newly-activated task replaces the interrupt vector in low memory with a pointer to the interrupt handler. (See "ROM Considerations" on page 5-7 for information about interrupt handlers that are stored in ROM.) The interrupt handler must preserve the interrupt vector that it is replacing and use it as a forward pointer to the next interrupt handler in the chain. The old interrupt vector must be stored at a fixed offset from the entry point of the new task's interrupt handler.



When the system acknowledges an interrupt request, each interrupt handler must determine whether it is the appropriate interrupt handler for the adapter that is presenting the interrupt request. To make this determination, the interrupt handler reads the contents of the interrupt-status register of the adapter.

When an interrupt handler determines that its device caused the interrupt, the interrupt handler must service the interrupt, reset the interrupt-status bit, clear the interrupts, issue a nonspecific EOI instruction to the interrupt controller, then execute an IRET instruction.

When an interrupt handler determines that its device did not cause the interrupt, the interrupt handler passes control to the next interrupt handler in the chain, using the previously-stored forward pointer.

To unlink an interrupt handler from a chain, a task first locates the position of its interrupt handler in the chain. The task searches the chain, starting at the interrupt vector in low memory, and using the offset of each interrupt handler's forward pointer to locate the entry point of each interrupt handler. The task repeats this process until it finds its own interrupt handler. The signature (hex 424B) of each interrupt handler must be checked to ensure that a valid forward pointer exists. The forward pointer of the task replaces the forward pointer of the previous interrupt handler in the chain, thus removing the interrupt handler from the chain.

**Note:** If the interrupt handler cannot locate its position in the chain, the interrupt handler cannot be unlinked.

An application-dependent unlinking error-recovery procedure must be incorporated into the unlinking routine for situations in which the unlinking routine determines that the interrupt chain contains an interrupt handler that is linked but does not have a valid signature. To avoid this error condition, all interrupt-sharing handlers, except those in ROM (see "ROM Considerations" on page 5-7), must use hex 424B as their signature.

In a system-reset condition, a routine for each interrupt handler must be executed after power-on to disable the interrupts from their devices.

Operating-system environments that support dynamic relocation of software must manage the entire interrupt-sharing process. Interrupt-handler software that is written exclusively for dynamic-relocation operating-system environments does not have to provide an interrupt-sharing chaining structure. These interrupt

handlers do not have to provide linking and unlinking support, but they must provide support for disabling the interrupt capability of the bus slave that they support.

## Interrupt-Sharing Chaining Structure and Signature

The interrupt-sharing software chaining structure has a 16-byte format that contains a 4-byte forward pointer (FPTR), a 2-byte signature, and 8 reserved bytes (RES\_BYTES), as shown in the following example:

ENTRY:	JMP	SHORT	PAST	; Jump around structure
	FPTR	DD	0	; Forward pointer
	SIGNATURE	DW	424BH	; Used when unlinking to identify
				; compatible interrupt handlers
	FLAGS	DB	0	; Flags
	FIRST	EQU	80H	; Flag for being first in chain
	JMP	SHORT	RESET	
	RES_BYTES	DB	DUP 7(0)	; -Reserved-
PAST:	...			; Actual start of code

The interrupt-sharing software chaining structure begins at the third byte from the entry point of the interrupt handler. The first instruction of each interrupt handler is a short jump around the structure, placing the structure at a known offset from the beginning of the interrupt-handler routine. Because the position of the chaining structure of each interrupt handler is known (except for the interrupt handlers on adapter ROM), the forward pointers can be updated during linking and unlinking.

The FIRST flag is used to determine the position of the interrupt handler in the chain during linking and unlinking for shared interrupt levels. The value of the FLAGS byte is changed to the value of the FIRST flag (hex 80) to indicate that the interrupt handler is the first interrupt handler that is linked in the chain. Each interrupt handler that is not stored in ROM must store the FIRST flag (hex 80) in the FLAGS byte when it is the first interrupt handler in the chain.

The Reset routine, an entry point for the operating system, must disable the adapter interrupt and return to the operating system.

## ROM Considerations

An adapter with an interrupt handler in ROM must implement chaining by storing the forward pointer in a latch or in a port on the adapter. If the adapter is sharing interrupt level 7 or 15, it must also store the FIRST flag, which indicates whether it is positioned first in the chain of interrupt handlers. Storage of this information is required because it cannot be guaranteed that interrupt handlers in ROM will always link first and never unlink. The forward pointer in ROM interrupt handlers is not stored at the third byte from the entry point of the interrupt handler; therefore the ROM interrupt handler must contain the signature, 0000H, beginning at the seventh byte from the entry point of the interrupt handler.

## Implementation Information

The Interrupt Mask register is located at I/O port hex 21. Specific End of Interrupt (EOI) values for the various interrupt levels are listed (hex 67 for level 7). To execute a specific EOI, issue an OUT instruction to the Programmable Interrupt Controller Operation Command register (port hex 20), using operation-command byte 2. To execute a nonspecific EOI, issue an OUT value of hex 20 to the Programmable Interrupt Controller Operation Command register (port hex 20).

The following are examples of code that implements interrupt sharing:

### Linking

```
        PUSH     ES
        CLI                      ;Clear interrupts
;Set forward pointer to value of interrupt vector in low memory
        ASSUME   CS:CODESEG,DS:CODESEG
        PUSH     ES
        MOV      AX,350FH        ;DOS get interrupt vector
        INT      21H
        MOV      SI,OFFSET CS:FPTR ;Get offset of your forward pointer
                                   ; in an indexable register
        MOV      CS:[SI],BX      ;Store the old interrupt vector
        MOV      CS:[SI+2],ES    ; in your forward pointer for
                                   ; chaining
        CMP      ES:BYTE PTR[BX],0CFH ;Test for IRET
if iret_test_only_is_needed ; See NOTE below
        JNE      SETVECTR
else
        JE       FRSTVCTR
        CMP      ES:WORD PTR[BX+6],424BH ; Is signature present?
        JE       SETVECTR
        MOV      AX,ES
```

```

        CMP        AX,0F000H        ;See if pointing to dummy handler
        JNE        SETVCTR
        CMP        BX,WORD PTR ES:[0FF01H] ; Dummy vector pointer?
        JNE        SETVECTR        ;If dummy, then first
FRSTVCTR:
        endif
        MOV        CS:FLAGS,FIRST    ;Set up first in chain flag
SETVECTR: POP        ES
        PUSH       DS
;Make interrupt vector in low memory point to your handler
        MOV        DX,OFFSET ENTRY   ;Make interrupt vector point to
                                        ; your handler
        MOV        AX,SEG ENTRY      ;If DS ≠ CS, get it and
        MOV        DS,AX             ; put it in DS
        MOV        AX,250FH         ;DOS set interrupt vector
        INT        21H
        POP        DS
;Unmask (enable) interrupts for your level
        IN         AL,IMR            ;Read interrupt mask register
        JMP        $+2              ;I/O delay
        AND        AL,07FH          ;Unmask interrupt level 7
        OUT        IMR,AL           ;Write new interrupt mask
        MOV        AL,SPC_EOI       ;Issue specific EOI for level 7
        JMP        $+2              ; to allow pending level 7 interrupts
        OUT        OCR,AL           ; (if any) to be serviced
        STI                     ;Enable interrupts
        POP        ES

```

## Notes:

1. The operating system must ensure that SEG:OFF points to a valid interrupt handler or to an IRET (hex 0CF) for levels 7 and 15.
2. Each ROM interrupt handler during ROMSCAN (before the operating system is loaded) and each interrupt handler on other than IRQ 7 must test SEG:OFF as shown in the "else" clause in this listing to determine whether it is the first interrupt handler in the chain. Checking SEG:OFF to determine whether it points to an IRET as the sole determination of FIRST is allowed only on IRQ 7, and then only after the operating system is loaded.

## Interrupt Handler

```

YOUR_CARD EQU        xxxx          ;Location of your card interrupt
                                        ; control/status register
ISB        EQU        xx           ;Interrupt bit in your card
REARM      EQU        2F7H         ;Interrupt control/status register
                                        ;Global Rearm location for
                                        ; interrupt level 7
SPC_EOI    EQU        67H          ;Specific EOI for programmable
                                        ; interrupt controller interrupt level 7
EOI        EQU        20H          ;Nonspecific EOI
OCR        EQU        20H          ;Location of programmable interrupt
                                        ; controller operation command register

```

IMR	EQU	21H	;Location of programmable interrupt ; controller interrupt mask
MYCSEG	SEGMENT	PARA	
	ASSUME	CS:MYCSEG,DS:DSEG	
ENTRY	PROC	FAR	
	JMP	SHORT PAST	;Entry point of handler
FPTR	DD	0	;Forward pointer
SIGNATURE	DW	424BH	;Used when unlinking to identify ; compatible interrupt handlers
FLAGS	DB	0	;Flags
FIRST	EQU	80H	
JMP	SHORT	RESET	
RES_BYTES	DB	7 DUP (0)	;Future expansion
PAST:	STI		;Actual start of handler code
	PUSH	...	;Save needed registers
	MOV	DX,YOUR_CARD	;Select your status register
	IN	AL,DX	;Read the status register
	TEST	AL,ISB	;Your card caused interrupt?
	JNZ	SERVICE	;Yes, branch to service logic
	TEST	CS:FLAGS,FIRST	;Are we the first ones in?
	JNZ	EXIT	;If yes, branch for EOI and Rearm
	POP	...	;Restore registers
	CLI		;Clear interrupts
	JMP	DWORD PTR CS:FPTR	;Pass control to next handler on chain
SERVICE:	...		;Service the interrupt
EXIT:	CLI		;Clear interrupts
	MOV	AL,EOI	
	OUT	OCR,AL	;Issue nonspecific EOI to programmable ; interrupt controller
	MOV	DX,REARM	;Rearm the cards
	OUT	DX,AL	
	POP	...	;Restore registers
	IRET		
RESET:	...		;Disable your card
	RET		;Return FAR to operating system
ENTRY	ENDP		
MYCSEG	ENDS		
	END	ENTRY	

## Unlinking

	PUSH	DS	
	PUSH	ES	
	CLI		;Clear interrupts
	MOV	AX,350FH	;DOS get interrupt vector
	INT	21H	;ES:BX points to first of chain
	MOV	CX,ES	;Pick up segment part of interrupt vector
	;Are we the first handler in the chain?		
	MOV	AX,CS	;Get code seg into comparable register
	CMP	BX,OFFSET ENTRY	;Interrupt vector in low memory ; pointing to your handler offset?
	JNE	UNCHAIN_A	;No, branch
	CMP	AX,CX	;Vector pointing to your handler ; segment?
	JNE	UNCHAIN_A	;No, branch

```

;Set interrupt vector in low memory to point to the handler pointed to
; by your pointer
    PUSH    DS
    MOV     DX,WORD PTR CS:FPTR
    MOV     DS,WORD PTR CS:FPTR[2]
    MOV     AX,250FH          ;DOS set interrupt vector
    INT     21H
    POP     DS
    JMP     UNCHAIN_X
UNCHAIN_A:  ; BX = FPTR offset, ES = FPTR segment, CX = CS
    CMP     ES:[BX+6],4842H    ;Is handler using the appropriate
                                ; conventions (is SIGNATURE present in
                                ; the interrupt chaining structure)?
    JNE     exception          ;No, invoke error exception handler
    LDS     SI,ES:[BX+2]       ;Get FPTR segment and offset
    CMP     SI,OFFSET ENTRY    ;Is this forward pointer pointing to
                                ; your handler offset?
    JNE     UNCHAIN_B          ;No, branch
    MOV     CX,DS              ;Move to compare
    CMP     AX,CX              ;Is this forward pointer pointing to
                                ; your handler segment?
    JNE     UNCHAIN_B          ;No, branch
;Locate your handler in the chain
    MOV     AX,WORD PTR CS:FPTR ; Get your FPTR offset
    MOV     ES:[BX+2],AX       ;Replace offset of FPTR of handler
                                ; that points to you
    MOV     AX,WORD PTR CS:FPTR[2] ; Get your FPTR segment
    MOV     ES:[BX+4],AX       ;Replace segment of FPTR of handler
                                ; that points to you
    MOV     AL,CS:FLAGS        ;Get your flags
    AND     AL,FIRST           ;Isolate FIRST flag
    OR      ES:[BX+6],AL       ;Set your first flag into prior routine
    JMP     UNCHAIN_X
UNCHAIN_B:  MOV     BX,SI       ;Move new offset to BX
    PUSH    DS                 ;Set pointer to next in chain
    POP     ES
    JMP     UNCHAIN_A          ;Examine next handler in chain
UNCHAIN_X:  STI               ;Enable interrupts
    POP     ES
    POP     DS

```

---

## Adapter ROM

BIOS provides a method for integrating adapters with on-board ROM code into the system. During power-on self-test (POST), interrupt vectors are established for BIOS calls. When the default vectors are in place, a scan for adapter-ROM modules occurs. At this point, an adapter-ROM routine can gain control. The routine can establish or intercept interrupt vectors to hook into the system.

Early in POST, the absolute addresses hex C0000 through hex C7FFF are scanned in 2KB blocks to search for adapter-ROM modules that need to be initialized (for example, valid video adapter ROM).

Later in POST, the absolute addresses hex C8000 through hex DFFFF are scanned in 2KB blocks to search for devices with valid adapter-ROM modules. Valid adapter ROM is defined as follows:

**Byte 0:** Hex 55

**Byte 1:** Hex AA

**Byte 2:** A length indicator, representing the number of 512-byte blocks (the limit is hex 7F) in the ROM (length divided by 512). A checksum tests the integrity of the ROM module. Each byte in the defined ROM is summed modulo hex 100. This sum must be 0 for the module to be valid.

When POST identifies valid adapter ROM, it executes a far call to byte 3 of the ROM (which should contain executable code). The device can now perform power-on initialization. The adapter ROM should return control to POST by executing a far return.

For the PC Convertible, if the adapter ROM detects a self-test error, it should perform the following steps before returning:

- Set bit 4 of hex 40:12 (POST status) to 1.
- Set the device number for the supported adapter in (AH).
- Set a two-digit error code in (AL).

If no self-test error is found, the adapter ROM should reset bit 4 of hex 40:12 (POST status) to 0 before returning.

For Personal System/2 products with Micro Channel architecture, a video adapter in the channel has a ROM signature code that identifies it. During POST, when CMOS is not valid (abnormal condition), the signature code is used to find the first video adapter and set up its ROM programmable option select (POS) parameters.

The code starts at hex 0C in the ROM address space and consists of:

77H, CCH, 'VIDEO '

The POS parameters are accessed from offset hex 30 in the ROM address space and are in the following order:

POS Byte 102, POS Byte 103, POS Byte 104, POS Byte 105

Video ROM scan remains from hex C0000 to hex C7FFF.

For Personal System/2 BIOS dated 10/7/87 and later, the adapter-ROM integration method is as follows:

Early in POST, the absolute addresses hex C0000 through DFFFF are scanned in 2KB blocks to search for adapter-ROM modules that have video-adapter signature code, as previously described. Only adapters with a video signature code are initialized early in POST.

Later in POST, the absolute addresses hex C0000 through DFFFF are scanned in 2KB blocks to initialize other adapter-ROM modules that were not initialized in the early scan.

For the PC Convertible, during early ROM scan, the following protocol is established to determine the video support:

On return from a call to a video-adapter-ROM module, (BH) indicates the following:

- (BH) = 00H - Not a video adapter
- = 02H - Video adapter supporting video in the color/graphics adapter range
- = 04H - Video adapter supporting video in the monochrome adapter range



---

## Video-Function Compatibility

The following procedures are recommended to provide video-function compatibility for application software.

### Video-Presence Test

Use this video-presence test to determine which IBM video functions are present.

1. Issue Interrupt 10H, Read/Write Display-Combination Code function with ((AH)=1AH), Read Display-Combination Code ((AL)=00H).

On return, if the value of (AL) is not hex 1A, the Read/Write Display-Combination Code function is not supported, and step 2 should be followed to determine video presence.

On return, if the value of (AL) is hex 1A, the information that is returned in (BX) defines the video environment. The active display code is returned in (BL). The alternative display code, if any, is returned in (BH). Refer to Interrupt 10H, Read/Write Display-Combination Code function ((AH)=1AH) for display-code definitions.

2. To determine the presence of an IBM Enhanced Graphics Adapter (EGA) when the Read/Write Display-Combination Code function is not supported, issue Interrupt 10H, Alternative Selection function ((AH)=12H), Return EGA Information ((BL)=10H).

On return, if the value of (BL) is hex 10, an EGA is not present, and step 3 should be followed.

On return, if the value of (BL) is not hex 10, an EGA is present. Note that an IBM Color/Graphics Monitor Adapter or an IBM Monochrome Display and Printer Adapter might also be present, depending on the EGA switch settings.

3. Complete steps 1 and 2 before performing this step. The video functions that might be present at this point are the IBM Color/Graphics Monitor Adapter, the IBM Monochrome Display and Printer Adapter, or both. Perform a presence test on video buffer addresses hex 0B8000 and hex 0B0000 to determine which video functions are present.

## Video-Mode Switching

Use the following video-mode switching procedure when applications will switch between monochrome and color video modes. A correct video-presence test, as previously described, is required. The following three system-video environments are possible:

1. A single video function that supports either monochrome or color video modes. If a monochrome function is present, only monochrome video modes are available. If a color function is present, only color video modes are available.
2. Two video functions, one supporting color video modes, and the other supporting monochrome video modes. In this case, both monochrome and color video modes are available. To switch from monochrome to color or from color to monochrome, the application program must change bits 5 and 4 (video mode type) of data area hex 40:10 to monochrome or color and issue Interrupt 10H, Set Mode function ((AH)=00H).
3. A single video function that supports both monochrome and color video modes. To determine whether a single video function supports both monochrome and color video modes, the application program should issue Interrupt 10H, Return Functionality/State Information function ((AH)=1BH).

On return, if the value of (AL) is not hex 1B, the Return Functionality/State Information function is not supported, and support for both monochrome and color video modes on a single video function is not available.

On return, if the value of (AL) is hex 1BH, use the returned information to determine whether bit 0 (all modes on all displays active) of byte (DI+2DH) is set to 1. If it is set to 1, both monochrome and color modes are available, and the application program should change the video-modes bits for monochrome or color and issue Interrupt 10H, Set Mode function ((AH)=00H). If it is set to 0, only monochrome modes or color modes are available, depending on the results of the video-presence test.

---

## Multitasking Provisions

BIOS provides hooks to assist in multitasking information. Whenever a Busy (Wait) loop occurs in BIOS, a hook is provided for the program to break out of the loop. Also, when BIOS services an interrupt, a corresponding Wait loop is ended, and another hook is provided. A program can be written that employs most of the device-driver code. The following information is valid only in the microprocessor real-address mode, and the code must use the following procedures to enable this support.

The program is responsible for matching corresponding Wait and Post calls and for the serialization of access to the device driver. The BIOS code is not reentrant.

The multitasking dispatcher uses the following four interfaces:

**Startup:** The startup code hooks Interrupt 15H. The dispatcher is responsible for checking for function codes (AH)=90H and (AH)=91H (see the following descriptions of Wait and Post). The dispatcher must pass all other functions to the previous user of Interrupt 15H (use a JMP or CALL instruction). If the value of (AH) is hex 90 or hex 91, the dispatcher must perform the appropriate processing and return by the IRET instruction.

**Serialization:** The multitasking system must ensure that the device driver code is used serially. Multiple entries into the code can result in errors.

**Wait (Busy):** When BIOS is about to enter a Wait loop, it first issues Interrupt 15H, (AH)=90H. This signals a Wait condition. At this point, the dispatcher should save the task status and dispatch another task. This enables overlapped execution of tasks when the hardware is busy. The following is an outline of the code that has been added to BIOS to perform this function.

```
MOV AX, 90xxH      ;Wait code in AH and
                   ; type code in AL
INT 15H            ;Issue call
JC TIMEOUT         ;Optional: for time-out or
                   ; if carry is set, time-out
                   ; occurred
NORMAL TIMEOUT LOGIC ;Normal time-out
```

**Post (Interrupt):** When BIOS has set an interrupt flag for a corresponding Busy loop, it issues Interrupt 15H, (AH)=91H. This signals a Post condition. At this point, the dispatcher must set the task status to "ready to run" and return to the interrupt routine. The following is an outline of the code that has been added to BIOS to perform this function.

```
MOV AX, 91xxH      ;Post code AH and  
                   ; type code AL  
INT 15H            ;Issue call
```

Three Wait loop function-code classes are supported:

- The first (hex 0 to hex 7F) is serially reusable. This means that for devices that use these codes, access to BIOS must be restricted to one task at a time, and the operating system must serialize access.
- The second (hex 80 to hex BF) is for reentrant devices. There is no restriction on the number of tasks that can use a device. (ES:BX) is used to distinguish between different calls.
- The third (hex C0 to hex FF) is noninterrupt (wait-only calls). There is no corresponding interrupt for the Wait loop. The dispatcher must take the appropriate action to satisfy this condition and exit from the loop. There is no complementary Post for these Wait conditions. They are time-out only, and the times are function-number dependent.

To support time-outs properly, the multitasking dispatcher must be aware of time. If a device enters a Busy loop, generally, it should remain there for a specific length of time before indicating an error. The dispatcher must return to the BIOS Wait loop with the carry flag set to 1 if a time-out occurs.

---

## Application Guidelines

Use the following information to develop application programs for IBM Personal System/2 and Personal Computer products. Whenever possible, BIOS should be used as an interface to hardware to provide maximum compatibility and portability of applications across systems.

### Math-Coprocessor Testing

Interrupt 11H (Equipment Determination) should be used where possible as the method for detecting the presence of the math coprocessor.

### Hardware Interrupts

Hardware interrupts are level sensitive for systems that use the Micro Channel architecture; hardware interrupts are edge triggered for systems that use the PC-type I/O channel. In edge-triggered-interrupt systems, the interrupt controller clears its internal 'interrupt in progress' latch when the interrupt routine sends an End of Interrupt (EOI) command to the controller. The EOI command is sent, regardless of whether the incoming interrupt request to the controller is active or inactive.

In level-sensitive-interrupt systems, the 'interrupt in progress' latch is readable at an I/O-address bit position. This latch is read during the Interrupt Service routine; it can be reset by the read operation, or it might require an explicit reset.

**Note:** For performance and latency considerations, designers might want to limit the number of devices that share an interrupt level.

The interrupt controller on level-sensitive interrupt systems requires the interrupt request to be inactive at the time the EOI command is sent; otherwise, a "new" interrupt request will be detected, and another microprocessor interrupt will be caused.

| To avoid this problem, a level-sensitive interrupt handler must clear the interrupt condition, usually by a Read or Write command to an I/O port on the device that is causing the interrupt.

| Because of the speed of the processor, a delay must be added after every repetitive I/O operation to a controller or device to ensure that

| I/O operations are synchronized with the processor. The JMP \$+2 instruction is not recommended for this purpose.

| Before the interrupt controllers are programmed, interrupts should be disabled through a Clear Interrupt Flag (CLI) instruction. This includes programming of the Mask register and issuing EOI instructions, initialization-command bytes, and operation-command bytes.

In level-sensitive-interrupt systems, hardware prevents the interrupt controllers from being set to the edge-triggered mode.

Hardware interrupt IRQ 9 is defined as the replacement interrupt level for the cascade level IRQ 2. Program interrupt sharing should be implemented on IRQ 2, Interrupt 0AH. The following processing occurs to maintain compatibility with the IRQ 2 that is used by IBM Personal Computer products:

1. A device drives the interrupt request that is active on IRQ 2 of the channel.
2. This interrupt request is mapped in hardware to IRQ 9 input on the slave interrupt controller.
3. When the interrupt occurs, the system microprocessor passes control to the IRQ 9 (Interrupt 71H) interrupt handler.
4. This interrupt handler sends an EOI command to the slave interrupt controller and passes control to the IRQ 2 (Interrupt 0AH) interrupt handler.
5. When handling the interrupt, the IRQ 2 interrupt handler causes the device to reset the interrupt request before sending an EOI command to the master interrupt controller that finishes servicing the IRQ 2 request.

## **Programming Considerations**

The IBM-supported languages of IBM C, BASIC, FORTRAN, COBOL, and Pascal are the best choices for writing compatible programs. If a program uses specific features of the hardware, the program might not be compatible with all IBM Personal System/2 and Personal Computer products.

Any program that requires precise timing information should obtain it through an operating system or language interface (for example, TIME\$ in BASIC). The use of programming loops can prevent a

program from being compatible with other Personal System/2 and IBM Personal Computer products and software.

## **BIOS and Operating-System Function Calls**

For maximum portability, programs should perform all I/O operations through operating-system function calls. In environments where the operating system does not provide the necessary programming interfaces, programs should access the hardware through BIOS function calls, if it is permissible. When writing programs, consider the following:

- In some environments, program interrupts are used for access to these functions. This practice removes the absolute addressing from the program. Only the interrupt number is required.
- The system can mask hardware sensitivity. New devices can change BIOS to accept the same programming interface on the new device.
- When BIOS provides parameter tables, such as for video or diskette, a program can substitute new parameter values by building a new copy of the table and changing the vector to point to that table. The program should copy the current table, using the current vector, and then modify those locations in the table that need to be changed. In this way, the program does not inadvertently change any values that should be left the same.
- The diskette parameters table that is pointed to by Interrupt 1EH consists of 11 parameters that are required for diskette operation. It is recommended that the values that are supplied in ROM be used. If it becomes necessary to modify any of the parameters, build another parameter block, and modify the address at Interrupt 1EH (hex 00:78) to point to the new block.

The parameters were established to allow:

- Some models of the IBM Personal Computer to operate both the 5.25-inch high-capacity diskette drive (96 tracks per inch) and the 5.25-inch double-sided diskette drive (48 tracks per inch)
- Some models of the Personal System/2 system to operate both the 3.5-inch 1.44MB diskette drive and the 3.5-inch 720KB diskette drive.

The Gap Length parameter is not always retrieved from the parameter block. The gap length that is used during diskette read, write, and verify operations is derived from within diskette

BIOS. The gap length for format operations is still obtained from the parameter block.

If a parameter block contains a Head Settle Time parameter value of 0 milliseconds, BIOS enforces a minimum head settle time of 15 milliseconds. Read and verify operations use the head settle time that is provided by the parameter block.

For any function that requires a parameter block that contains a Motor Start Wait parameter of less than 500 milliseconds (1 second for a Personal Computer product), diskette BIOS enforces a minimum time of 500 milliseconds (1 second for a Personal Computer product).

- Programs can be designed to reside on both 5.25-inch and 3.5-inch diskettes. Because not all programs are operating-system dependent, the following procedure can be used to determine which type of media is in a diskette drive:
  1. Verify track 0, head 0, sector 1 (1 sector). This enables diskette BIOS to determine whether the format of the media is a recognizable type.

If the verify operation fails, issue the Reset function ((AH)=00H) to diskette BIOS and try the operation again. If it fails again, the media needs to be formatted or is defective.

2. Verify track 0, head 0, sector 16 (1 sector).

If the verify operation fails, either a 5.25-inch (48 TPI) or a 3.5-inch 720KB diskette is installed. To determine the type, verify track 78, head 1, sector 1 (1 sector). A successful verification of track 78 indicates that a 3.5-inch 720KB diskette is installed; a failed verification indicates that a 5.25-inch (48 TPI) diskette is installed.

**Note:** Refer to the *DOS Technical Reference* for the file-allocation-table parameters for single-sided and double-sided diskettes.

3. Read the diskette-controller status in BIOS, starting with address hex 40:42. The fifth byte defines the head that the operation ended with. If the operation ended with head 1, the diskette is a 5.25-inch high-capacity (96 TPI) diskette; if the operation ended with head 0, the diskette is a 3.5-inch 1.44MB diskette.
4. Newer Personal System/2 products might support Interrupt 13H—Diskette, Get Media Type function ((AH)=20H).



## Section 6. System Identification

Each BIOS ROM module has a model byte at address hex F000:FFFE in ROM. In some cases, a submodel byte and a BIOS revision-level byte are used to further distinguish between the various BIOS ROM modules. To gain access to this information, see Interrupt 15H, Return System Configuration Parameters function ((AH) = C0H).

A model byte changes when there are major processing-unit architecture changes. For example, 80286-based systems have model byte hex FC and have the following additional features:

- Protected virtual address mode
- 24 bits of addressing.

The 80386- and 80486-based systems have model byte hex F8 and have the following additional features:

- 32-bit registers
- 32-bit addressing
- 32-bit flat model mode
- Virtual 8086 mode.

A submodel byte changes when a system implements a software-detectable change from previous systems with the same model byte.

A BIOS revision level changes when the ROM image changes for a model or submodel.

The following table lists the system-identification information for Personal Computer and Personal System/2 products.

Product	BIOS Date	Model Byte	Submodel Byte	Revision
PC	4/24/81	FF	00	00
PC	10/19/81	FF	00	01
PC	10/27/82	FF	00	02
PC/XT	11/8/82	FE	00	00
PC/XT	1/10/86	FB	00	01
PC/XT	5/9/86	FB	00	02
PCjr	6/1/83	FD	00	00

Figure 6-1 (Part 1 of 3). System Identification

<b>Product</b>	<b>BIOS Date</b>	<b>Model Byte</b>	<b>Submodel Byte</b>	<b>Revision</b>
AT	1/10/84	FC	00	00
AT	6/10/85	FC	00	01
AT	11/15/85	FC	01	00
PC/XT Model 286	4/21/86	FC	02	00
PC Convertible	9/13/85	F9	00	00
PS/2 Model 25	6/26/87	FA	01	00
PS/2 Model 25	11/2/88	FA	01	01
PS/2 Model 30	9/2/86	FA	00	00
PS/2 Model 30	1/31/89	FA	00	04
PS/2 Model 30 286	8/25/88	FC	09	00
PS/2 Model 30 286	11/30/88	FC	09	01
PS/2 Model 30 286	5/30/89	FC	09	02
PS/2 Model 30 286		FC	09	03
PS/2 Model 40 SX/35 SX	3/15/91	F8	19	05
PS/2 Model 40 SX/35 SX	4/4/91	F8	19	06
PS/2 Model 40 SX/35 SX	6/4/91	F8	19	07
PS/2 Model L40 SX		F8	23	00
PS/2 Model 50 (Type 1)	2/13/87	FC	04	00
PS/2 Model 50		FC	04	01
PS/2 Model 50 (Type 1)	11/2/89	FC	04	02
PS/2 Model 50 (Type 2)	1/28/88	FC	04	03
PS/2 Model 50	5/12/88	FC	04	04
PS/2 Model 55 SX	11/2/88	F8	0C	00
PS/2 Model 55 SX		F8	0C	01
PS/2 Model 55 LS	2/8/90	F8	1E	00
PS/2 Model 57 SX	5/10/91	F8	26	00
PS/2 Model 60	2/13/87	FC	05	00
PS/2 Model 65 SX	2/8/90	F8	1C	00
PS/2 Model 70 (Type 2)	4/11/88	F8	04	02
PS/2 Model 70	3/17/89	F8	04	03
PS/2 Model 70	12/15/89	F8	04	04
PS/2 Model 70 (Type 1)	4/11/88	F8	09	02
PS/2 Model 70	3/17/89	F8	09	03
PS/2 Model 70	12/15/89	F8	09	04
PS/2 Model 70 (Type 3)	6/8/88	F8	0D	00
PS/2 Model 70	2/20/88	F8	0D	01
PS/2 Model 70 486 (Type 4)	12/1/89	F8	1B	00
PS/2 Model P70		F8	0B	00
PS/2 Model P70		F8	0B	02
PS/2 Model 80 (Type 1)	3/30/87	F8	00	00
PS/2 Model 80		F8	00	01
PS/2 Model 80	6/19/89	F8	00	02
PS/2 Model 80 (Type 2)	10/7/87	F8	01	00
PS/2 Model 80	11/21/89	F8	80	01
PS/2 Model 80	2/15/90	F8	80	02

**Figure 6-1 (Part 2 of 3). System Identification**

<b>Product</b>	<b>BIOS Date</b>	<b>Model Byte</b>	<b>Submodel Byte</b>	<b>Revision</b>
PS/2 Model 90 (Type 1)	10/1/90	F8	11	00
PS/2 Model 90 (Type 2)	10/1/90	F8	13	00
PS/2 Model 90 (Type 3)	4/24/91	F8	2D	01
PS/2 Model 90 (Type 3)	4/24/91	F8	2F	01
PS/2 Model 95 (Type 1)	10/1/90	F8	14	00
PS/2 Model 95 (Type 2)	10/1/90	F8	16	00
PS/2 Model 95 (Type 3)	4/24/91	F8	2C	01
PS/2 Model 95 (Type 3)	4/24/91	F8	2E	01

**Figure 6-1 (Part 3 of 3). System Identification**

**Note:** Information about a specific system board can be found in the technical reference for that model.

## Notes:

---

## Section 7. Scan Code/Character Code Combinations

**Note:** For scan code/character code combinations for the PC Space Saving (84-/85-key) Keyboard, refer to the *Personal System/2 Model 25 Technical Reference* or the "Keyboards" section of the *Personal System/2 Hardware Interface Technical Reference—Common Interfaces*.

The Keyboard Read function ((AH)=00H) and the Keyboard Status function ((AH)=01H) of Interrupt 16H apply to the 83-/84-key keyboard, the 101-/102-key keyboard (standard and extended function), and the 122-key keyboard. However, these functions return only the scan code/character code combinations that are supported by the 83-/84-key keyboard and the 101-/102-key standard-function keyboard.

The Extended-Keyboard Read function ((AH)=10H) and the Extended Keystroke Status function ((AH)=11H) apply to the 83-/84-key keyboard, the 101-/102-key keyboard (standard and extended function), and the 122-key keyboard. However, these functions return only the scan code/character code combinations that are supported by the 101-/102-key extended-function keyboard.

The Keyboard Read for the 122-Key Keyboard function ((AH)=20H) and the Keystroke Status for the 122-Key Keyboard function ((AH)=21H) apply to the 83-/84-key keyboard, the 101-/102-key keyboard (standard and extended function), and the 122-key keyboard. See the table on page 7-13 for the additional keystrokes and scan code/character code combinations that are supported by the 122-key keyboard.

The following table lists the keyboard keystrokes and the scan code/character code combinations that are returned through Interrupt 16H.

<b>Keystroke</b>	<b>83-/84-Key Standard Function</b>	<b>101-/102-Key Standard Function</b>	<b>101-/102-Key Extended Function</b>
Esc	01/1B	01/1B	01/1B
1	02/31	02/31	02/31
2	03/32	03/32	03/32
3	04/33	04/33	04/33
4	05/34	05/34	05/34
5	06/35	06/35	06/35
6	07/36	07/36	07/36
7	08/37	08/37	08/37
8	09/38	09/38	09/38
9	0A/39	0A/39	0A/39
0	0B/30	0B/30	0B/30
-	0C/2D	0C/2D	0C/2D
=	0D/3D	0D/3D	0D/3D
Backspace	0E/08	0E/08	0E/08
Tab	0F/09	0F/09	0F/09
q	10/71	10/71	10/71
w	11/77	11/77	11/77
e	12/65	12/65	12/65
r	13/72	13/72	13/72
t	14/74	14/74	14/74
y	15/79	15/79	15/79
u	16/75	16/75	16/75
i	17/69	17/69	17/69
o	18/6F	18/6F	18/6F
p	19/70	19/70	19/70
[	1A/5B	1A/5B	1A/5B
]	1B/5D	1B/5D	1B/5D
Return	1C/0D	1C/0D	1C/0D
Ctrl	..	..	..
a	1E/61	1E/61	1E/61
s	1F/73	1F/73	1F/73
d	20/64	20/64	20/64
f	21/66	21/66	21/66

**Figure 7-1 (Part 1 of 3). Keyboard Keystrokes**

<b>Keystroke</b>	<b>83-/84-Key Standard Function</b>	<b>101-/102-Key Standard Function</b>	<b>101-/102-Key Extended Function</b>
g	22/67	22/67	22/67
h	23/68	23/68	23/68
j	24/6A	24/6A	24/6A
k	25/6B	25/6B	25/6B
l	26/6C	26/6C	26/6C
;	27/3B	27/3B	27/3B
'	28/27	28/27	28/27
Shift	29/60	29/60	29/60
\	2B/5C	2B/5C	2B/5C
z	2C/7A	2C/7A	2C/7A
x	2D/78	2D/78	2D/78
c	2E/63	2E/63	2E/63
v	2F/76	2F/76	2F/76
b	30/62	30/62	30/62
n	31/6E	31/6E	31/6E
m	32/6D	32/6D	32/6D
.	33/2C	33/2C	33/2C
/	34/2E	34/2E	34/2E
*	35/2F	35/2F	35/2F
	37/2A	37/2A	37/2A
Alt	**	**	**
Space	39/20	39/20	39/20
Caps Lock	**	**	**
F1	3B/00	3B/00	3B/00
F2	3C/00	3C/00	3C/00
F3	3D/00	3D/00	3D/00
F4	3E/00	3E/00	3E/00
F5	3F/00	3F/00	3F/00
F6	40/00	40/00	40/00
F7	41/00	41/00	41/00
F8	42/00	42/00	42/00
F9	43/00	43/00	43/00

**Figure 7-1 (Part 2 of 3). Keyboard Keystrokes**

<b>Keystroke</b>	<b>83-/84-Key Standard Function</b>	<b>101-/102-Key Standard Function</b>	<b>101-/102-Key Extended Function</b>
F10	44/00	44/00	44/00
F11	(no key)	--	85/00
F12	(no key)	--	86/00
Num Lock	**	**	**
Scroll Lock	**	**	**
Home	47/00	47/00	47/00
Up Arrow	48/00	48/00	48/00
PgUp	49/00	49/00	49/00
-	4A/2D	4A/2D	4A/2D
Left Arrow	4B/00	4B/00	4B/00
Center Key	--	--	4C/00
Right Arrow	4D/00	4D/00	4D/00
+	4E/2B	4E/2B	4E/2B
End	4F/00	4F/00	4F/00
Down Arrow	50/00	50/00	50/00
PgDn	51/00	51/00	51/00
Ins	52/00	52/00	52/00
Del	53/00	53/00	53/00
SysRq	**	(no key)	(no key)
Key 45	(no key)	56/5C	56/5C
Enter	(no key)	1C/0D	E0/0D
/	(no key)	35/2F	E0/2F
PrtSc	(no key)	**	**
Pause	(no key)	**	**
Home	(no key)	47/00	47/E0
Up Arrow	(no key)	48/00	48/E0
PageUp	(no key)	49/00	49/E0
Left Arrow	(no key)	4B/00	4B/E0
Right Arrow	(no key)	4D/00	4D/E0
End	(no key)	4F/00	4F/E0
Down Arrow	(no key)	50/00	50/E0
PageDown	(no key)	51/00	51/E0
Insert	(no key)	52/00	52/E0
Delete	(no key)	53/00	53/00

\*\* These combinations do not provide a keystroke for the application that is currently running, but they perform some other action. They are not put into the Interrupt 16H queue.

-- These combinations have no function and are ignored.

**Figure 7-1 (Part 3 of 3). Keyboard Keystrokes**



The following table lists the Shift keyboard keystrokes and the scan code/character code combinations that are returned through Interrupt 16H.

<b>Keystroke</b>	<b>83-/84-Key Standard Function</b>	<b>101-/102-Key Standard Function</b>	<b>101-/102-Key Extended Function</b>
Shift Esc	01/1B	01/1B	01/1B
Shift !	02/21	02/21	02/21
Shift @	03/40	03/40	03/40
Shift #	04/23	04/23	04/23
Shift \$	05/24	05/24	05/24
Shift %	06/25	06/25	06/25
Shift ^	07/5E	07/5E	07/5E
Shift &	08/26	08/26	08/26
Shift *	09/2A	09/2A	09/2A
Shift (	0A/28	0A/28	0A/28
Shift )	0B/29	0B/29	0B/29
Shift _	0C/5F	0C/5F	0C/5F
Shift +	0D/2B	0D/2B	0D/2B
Shift Backspace	0E/08	0E/08	0E/08
Shift Tab (Backtab)	0F/00	0F/00	0F/00
Shift Q	10/51	10/51	10/51
Shift W	11/57	11/57	11/57
Shift E	12/45	12/45	12/45
Shift R	13/52	13/52	13/52
Shift T	14/54	14/54	14/54
Shift Y	15/59	15/59	15/59
Shift U	16/55	16/55	16/55
Shift I	17/49	17/49	17/49
Shift O	18/4F	18/4F	18/4F
Shift P	19/50	19/50	19/50
Shift {	1A/7B	1A/7B	1A/7B
Shift }	1B/7D	1B/7D	1B/7D
Shift Return	1C/0D	1C/0D	1C/0D
Shift Ctrl	**	**	**
Shift A	1E/41	1E/41	1E/41
Shift S	1F/53	1F/53	1F/53
Shift D	20/44	20/44	20/44
Shift F	21/46	21/46	21/46
Shift G	22/47	22/47	22/47
Shift H	23/48	23/48	23/48
Shift J	24/4A	24/4A	24/4A
Shift K	25/4B	25/4B	25/4B
Shift L	26/4C	26/4C	26/4C
Shift :	27/3A	27/3A	27/3A
Shift "	28/22	28/22	28/22
Shift ~	29/7E	29/7E	29/7E

Figure 7-2 (Part 1 of 3). Shift Keyboard Keystrokes

<b>Keystroke</b>	<b>83-/84-Key Standard Function</b>	<b>101-/102-Key Standard Function</b>	<b>101-/102-Key Extended Function</b>
Shift	2B/7C	2B/7C	2B/7C
Shift Z	2C/5A	2C/5A	2C/5A
Shift X	2D/58	2D/58	2D/58
Shift C	2E/43	2E/43	2E/43
Shift V	2F/56	2F/56	2F/56
Shift B	30/42	30/42	30/42
Shift N	31/4E	31/4E	31/4E
Shift M	32/4D	32/4D	32/4D
Shift <	33/3C	33/3C	33/3C
Shift >	34/3E	34/3E	34/3E
Shift ?	35/3F	35/3F	35/3F
Shift *	37/2A	37/2A	37/2A
Shift Alt	**	**	**
Shift Space	39/20	39/20	39/20
Shift Caps Lock	**	**	**
Shift F1	54/00	54/00	54/00
Shift F2	55/00	55/00	55/00
Shift F3	56/00	56/00	56/00
Shift F4	57/00	57/00	57/00
Shift F5	58/00	58/00	58/00
Shift F6	59/00	59/00	59/00
Shift F7	5A/00	5A/00	5A/00
Shift F8	5B/00	5B/00	5B/00
Shift F9	5C/00	5C/00	5C/00
Shift F10	5D/00	5D/00	5D/00
Shift F11	(no key)	--	87/00
Shift F12	(no key)	--	88/00
Shift Num Lock	**	**	**
Shift Scroll Lock	**	**	**
Shift 7	47/37	47/37	47/37
Shift 8	48/38	48/38	48/38
Shift 9	49/39	49/39	49/39
Shift -	4A/2D	4A/2D	4A/2D
Shift 4	4B/34	4B/34	4B/34
Shift 5	4C/35	4C/35	4C/35
Shift 6	4D/36	4D/36	4D/36
Shift +	4E/2B	4E/2B	4E/2B
Shift 1	4F/31	4F/31	4F/31
Shift 2	50/32	50/32	50/32
Shift 3	51/33	51/33	51/33
Shift 0	52/30	52/30	52/30
Shift .	53/2E	53/2E	53/2E
Shift SysRq	**	(no key)	(no key)
Shift Key 45	(no key)	56/7C	56/7C
Shift Enter	(no key)	1C/0D	E0/0D
Shift /	(no key)	35/2F	E0/2F

**Figure 7-2 (Part 2 of 3). Shift Keyboard Keystrokes**

Keystroke	83-/84-Key Standard Function	101-/102-Key Standard Function	101-/102-Key Extended Function
Shift PrtSc	(no key)	**	**
Shift Pause	(no key)	**	**
Shift Home	(no key)	47/00	47/E0
Shift Up Arrow	(no key)	48/00	48/E0
Shift PgUp	(no key)	49/00	49/E0
Shift Left Arrow	(no key)	4B/00	4B/E0
Shift Right	(no key)	4D/00	4D/E0
Shift End	(no key)	4F/00	4F/E0
Shift Down Arrow	(no key)	50/00	50/E0
Shift PgDn	(no key)	51/00	51/E0
Shift Insert	(no key)	52/00	52/E0
Shift Delete	(no key)	53/00	53/E0

\*\* These combinations do not provide a keystroke for the application that is currently running, but they perform some other action. They are not put into the Interrupt 16H queue.

-- These combinations have no function and are ignored.

**Figure 7-2 (Part 3 of 3). Shift Keyboard Keystrokes**

The following table lists the Ctrl keyboard keystrokes and the scan code/character code combinations that are returned through Interrupt 16H.

Keystroke	83-/84-Key Standard Function	101-/102-Key Standard Function	101-/102-Key Extended Function
Ctrl Esc	01/1B	01/1B	01/1B
Ctrl 1	--	--	--
Ctrl 2 (NUL)	03/00	03/00	03/00
Ctrl 3	--	--	--
Ctrl 4	--	--	--
Ctrl 5	--	--	--
Ctrl 6 (RS)	07/1E	07/1E	07/1E
Ctrl 7	--	--	--
Ctrl 8	--	--	--
Ctrl 9	--	--	--
Ctrl 0	--	--	--
Ctrl _	0C/1F	0C/1F	0C/1F
Ctrl =	--	--	--
Ctrl Backspace (DEL)	0E/7F	0E/7F	0E/7F
Ctrl Tab	--	--	94/00

**Figure 7-3 (Part 1 of 3). Ctrl Keyboard Keystrokes**

<b>Keystroke</b>	<b>83-/84-Key Standard Function</b>	<b>101-/102-Key Standard Function</b>	<b>101-/102-Key Extended Function</b>
Ctrl q (DC1)	10/11	10/11	10/11
Ctrl w (ETB)	11/17	11/17	11/17
Ctrl e (ENQ)	12/05	12/05	12/05
Ctrl r (DC2)	13/12	13/12	13/12
Ctrl t (DC4)	14/14	14/14	14/14
Ctrl y (EM)	15/19	15/19	15/19
Ctrl u (NAK)	16/15	16/15	16/15
Ctrl i (HT)	17/09	17/09	17/09
Ctrl o (SI)	18/0F	18/0F	18/0F
Ctrl p (DLE)	19/10	19/10	19/10
Ctrl [ (ESC)	1A/1B	1A/1B	1A/1B
Ctrl ] (GS)	1B/1D	1B/1D	1B/1D
Ctrl Return (LF)	1C/0A	1C/0A	1C/0A
Ctrl a (SOH)	1E/01	1E/01	1E/01
Ctrl s (DC3)	1F/13	1F/13	1F/13
Ctrl d (EOT)	20/04	20/04	20/04
Ctrl f (ACK)	21/06	21/06	21/06
Ctrl g (BEL)	22/07	22/07	22/07
Ctrl h (Backspace)	23/08	23/08	23/08
Ctrl j (LF)	24/0A	24/0A	24/0A
Ctrl k (VT)	25/0B	25/0B	25/0B
Ctrl l (FF)	26/0C	26/0C	26/0C
Ctrl ;	--	--	--
Ctrl ' (	--	--	--
Ctrl ' (	--	--	--
Ctrl Shift	--	--	--
Ctrl \ (FS)	2B/1C	2B/1C	2B/1C
Ctrl z (SUB)	2C/1A	2C/1A	2C/1A
Ctrl x (CAN)	2D/18	2D/18	2D/18
Ctrl c (ETX)	2E/03	2E/03	2E/03
Ctrl v (SYN)	2F/16	2F/16	2F/16
Ctrl b (STX)	30/02	30/02	30/02
Ctrl n (SO)	31/0E	31/0E	31/0E
Ctrl m (CR)	32/0D	32/0D	32/0D
Ctrl ,	--	--	--
Ctrl .	--	--	--
Ctrl /	--	--	--
Ctrl *	--	--	96/00
Ctrl Alt	--	--	--
Ctrl Space	39/20	39/20	39/20
Ctrl Caps Lock	--	--	--
Ctrl F1	5E/00	5E/00	5E/00
Ctrl F2	5F/00	5F/00	5F/00
Ctrl F3	60/00	60/00	60/00
Ctrl F4	61/00	61/00	61/00
Ctrl F5	62/00	62/00	62/00

*Figure 7-3 (Part 2 of 3). Ctrl Keyboard Keystrokes*

Keystroke	83-/84-Key Standard Function	101-/102-Key Standard Function	101-/102-Key Extended Function
Ctrl F6	63/00	63/00	63/00
Ctrl F7	64/00	64/00	64/00
Ctrl F8	65/00	65/00	65/00
Ctrl F9	66/00	66/00	66/00
Ctrl F10	67/00	67/00	67/00
Ctrl F11	(no key)	--	89/00
Ctrl F12	(no key)	--	8A/00
Ctrl Num Lock	--	--	--
Ctrl Scroll Lock	--	--	--
Ctrl Home	77/00	77/00	77/00
Ctrl Up Arrow	--	--	8D/00
Ctrl PgUp	84/00	84/00	84/00
Ctrl Keypad -	--	--	8E/00
Ctrl Left Arrow	73/00	73/00	73/00
Ctrl Center	--	--	8F/00
Ctrl Right Arrow	74/00	74/00	74/00
Ctrl Keypad +	--	--	90/00
Ctrl End	75/00	75/00	75/00
Ctrl Down Arrow	--	--	91/00
Ctrl PgDn	76/00	76/00	76/00
Ctrl Ins	--	--	92/00
Ctrl Del	--	--	93/00
Ctrl SysRq	**	(no key)	(no key)
Ctrl Key 45	(no key)	--	--
Ctrl Enter	(no key)	1C/0A	E0/0A
Ctrl /	(no key)	--	95/00
Ctrl PrtSc	(no key)	72/00	72/00
Ctrl Break	(no key)	00/00	00/00
Ctrl Home	(no key)	77/00	77/E0
Ctrl Up	(no key)	--	8D/E0
Ctrl PageUp	(no key)	84/00	84/E0
Ctrl Left	(no key)	73/00	73/E0
Ctrl Right	(no key)	74/00	74/E0
Ctrl End	(no key)	75/00	75/E0
Ctrl Down	(no key)	--	91/E0
Ctrl PageDown	(no key)	76/00	76/E0
Ctrl Insert	(no key)	--	92/E0
Ctrl Delete	(no key)	--	93/E0

\*\* These combinations do not provide a keystroke for the application that is currently running, but they perform some other action. They are not put into the Interrupt 16H queue.

-- These combinations have no function and are ignored.

Figure 7-3 (Part 3 of 3). Ctrl Keyboard Keystrokes

The following table lists the Alt keyboard keystrokes and the scan code/character code combinations that are returned through Interrupt 16H.

Keystroke	83-/84-Key Standard Function	101-/102-Key Standard Function	101-/102-Key Extended Function
Alt Esc	--	--	01/00
Alt 1	78/00	78/00	78/00
Alt 2	79/00	79/00	79/00
Alt 3	7A/00	7A/00	7A/00
Alt 4	7B/00	7B/00	7B/00
Alt 5	7C/00	7C/00	7C/00
Alt 6	7D/00	7D/00	7D/00
Alt 7	7E/00	7E/00	7E/00
Alt 8	7F/00	7F/00	7F/00
Alt 9	80/00	80/00	80/00
Alt 0	81/00	81/00	81/00
Alt -	82/00	82/00	82/00
Alt =	83/00	83/00	83/00
Alt Backspace	--	--	0E/00
Alt Tab	--	--	A5/00
Alt q	10/00	10/00	10/00
Alt w	11/00	11/00	11/00
Alt e	12/00	12/00	12/00
Alt r	13/00	13/00	13/00
Alt t	14/00	14/00	14/00
Alt y	15/00	15/00	15/00
Alt u	16/00	16/00	16/00
Alt i	17/00	17/00	17/00
Alt o	18/00	18/00	18/00
Alt p	19/00	19/00	19/00
Alt [	--	--	1A/00
Alt ]	--	--	1B/00
Alt Return	--	--	1C/00
Alt Ctrl	--	--	--
Alt a	1E/00	1E/00	1E/00
Alt s	1F/00	1F/00	1F/00
Alt d	20/00	20/00	20/00
Alt f	21/00	21/00	21/00
Alt g	22/00	22/00	22/00
Alt h	23/00	23/00	23/00
Alt j	24/00	24/00	24/00
Alt k	25/00	25/00	25/00
Alt l	26/00	26/00	26/00
Alt ;	--	--	27/00
Alt '	--	--	28/00
Alt `	--	--	29/00

Figure 7-4 (Part 1 of 3). Alt Keyboard Keystrokes

<b>Keystroke</b>	<b>83-/84-Key Standard Function</b>	<b>101-/102-Key Standard Function</b>	<b>101-/102-Key Extended Function</b>
Alt Shift	**	**	**
Alt \	--	--	2B/00
Alt z	2C/00	2C/00	2C/00
Alt x	2D/00	2D/00	2D/00
Alt c	2E/00	2E/00	2E/00
Alt v	2F/00	2F/00	2F/00
Alt b	30/00	30/00	30/00
Alt n	31/00	31/00	31/00
Alt m	32/00	32/00	32/00
Alt ,	--	--	33/00
Alt .	--	--	34/00
Alt /	--	--	35/00
Alt *	--	--	37/00
Alt Space	39/20	39/20	39/20
Alt Caps Lock	**	**	**
Alt F1	68/00	68/00	68/00
Alt F2	69/00	69/00	69/00
Alt F3	6A/00	6A/00	6A/00
Alt F4	6B/00	6B/00	6B/00
Alt F5	6C/00	6C/00	6C/00
Alt F6	6D/00	6D/00	6D/00
Alt F7	6E/00	6E/00	6E/00
Alt F8	6F/00	6F/00	6F/00
Alt F9	70/00	70/00	70/00
Alt F10	71/00	71/00	71/00
Alt F11	(no key)	--	8B/00
Alt F12	(no key)	--	8C/00
Alt Num Lock	**	**	**
Alt Scroll Lock	**	**	**
Alt Keypad -	--	--	4A/00
Alt Keypad +	--	--	4E/00
Alt Keypad Numbers	#	#	#
Alt Del	**	--	--
Alt SysRq	**	(no key)	(no key)
Alt Key 45	(no key)	--	--
Alt Enter	(no key)	--	A6/00
Alt /	--	--	A4/00
Alt Print Screen	(no key)	**	**
Alt Pause	(no key)	**	**
Alt Home	(no key)	--	97/00
Alt Up	(no key)	--	98/00
Alt PageUp	(no key)	--	99/00
Alt Left	(no key)	--	9B/00
Alt Right	(no key)	--	9D/00
Alt End	(no key)	--	9F/00
Alt Down	(no key)	--	A0/00

**Figure 7-4 (Part 2 of 3). Alt Keyboard Keystrokes**

Keystroke	83-/84-Key Standard Function	101-/102-Key Standard Function	101-/102-Key Extended Function
Alt PageDown	(no key)	—	A1/00
Alt Insert	(no key)	—	A2/00
Alt Delete	(no key)	—	A3/00
<p>** These combinations do not provide a keystroke for the application that is currently running, but they perform some other action. They are not put into the Interrupt 16H queue.</p> <p>— These combinations have no function and are ignored.</p> <p># See the note below for use of the Alt key with the keypad number keys.</p>			

**Figure 7-4 (Part 3 of 3). Alt Keyboard Keystrokes**

**Note:** For all keyboards, the numeric keypad can be used in combination with the Alt key to type any ASCII character. The scan code (always 00) and character code are returned after the Alt key is released. For example, pressing the Alt key and keypad 1, then releasing the Alt key returns scan code/character code combination hex 00/01; pressing the Alt key and keypad 255, then releasing the Alt key returns scan code/character code combination hex 00/FF.



The following table lists the keyboard keystrokes and the scan code/character code combinations that the 122-keyboard supports in addition to those listed previously. The Keyboard Read for the 122-Key Keyboard function ((AH)=20H) and the Keystroke Status for the 122-Key Keyboard function ((AH)=21H) must be used to return scan code/character code combinations for these keystrokes.

<b>Keystroke</b>	<b>122-Key Extended Extended Function</b>
F13	EC/00
F14	ED/00
F15	EE/00
F16	EF/00
F17	F4/00
F18	F5/00
F19	F6/00
F20	F7/00
F21	F8/00
F22	F9/00
F23	FA/00
F24	C0/00
PA1	E8/00
EraseEOF	F0/00
Clear	FB/00
Shift F13	C1/00
Shift F14	C3/00
Shift F15	C4/00
Shift F16	C5/00
Shift F17	C6/00
Shift F18	C7/00
Shift F19	C8/00
Shift F20	C9/00
Shift F21	CA/00
Shift F22	CB/00
Shift F23	CC/00
Shift F24	CD/00
Shift PA1	E9/00
Shift EraseEOF	F1/00
Shift Clear	FB/00
Ctrl F13	CE/00
Ctrl F14	CF/00
Ctrl F15	D0/00
Ctrl F16	D1/00
Ctrl F17	D2/00
Ctrl F18	D3/00
Ctrl F19	D4/00
Ctrl F20	D5/00
Ctrl F21	D6/00
Ctrl F22	D7/00
Ctrl F23	D8/00

**Figure 7-5 (Part 1 of 2). Keyboard Keystrokes – 122-Key Keyboard**

<b>Keystroke</b>	<b>122-Key Extended Extended Function</b>
Ctrl F24	D9/00
Ctrl PA1	EA/00
Ctrl EraseEOF	F2/00
Ctrl Clear	FC/00
Alt F13	DA/00
Alt F14	DB/00
Alt F15	DC/00
Alt F16	DD/00
Alt F17	DE/00
Alt F18	DF/00
Alt F19	E2/00
Alt F20	E3/00
Alt F21	E4/00
Alt F22	E5/00
Alt F23	E6/00
Alt F24	E7/00
Alt PA1	EB/00
Alt EraseEOF	F3/00
Alt Clear	FD/00

**Figure 7-5 (Part 2 of 2). Keyboard Keystrokes – 122-Key Keyboard**

The following keys on the leftmost keypad of the 122-key keyboard do not produce scan code/character code combinations:

- Attn
- CrSel
- ExSel
- Copy Play
- The two blank keys.

---

# Contents—ABIOS

<b>Section 1. Introduction to Advanced BIOS</b>	1-1
Introduction	1-3
Data Structures	1-4
Initialization	1-5
Transfer Conventions	1-6
Interrupt Processing	1-7
Extending ABIOS	1-8
 <b>Section 2. Data Structures</b>	2-1
Introduction	2-3
Common Data Area	2-3
Function Transfer Table	2-7
Device Block	2-9
 <b>Section 3. Initialization</b>	3-1
Introduction	3-3
Build System Parameters Table—Operating System	3-4
Build System Parameters Table—BIOS	3-4
Build Initialization Table—Operating System	3-6
Build Initialization Table—BIOS	3-6
Build Common Data Area—Operating System	3-8
Initialize Pointers—Operating System	3-9
Initialize Data Structures—ABIOS	3-10
Logical ID 2 Initialization	3-12
Build Protected-Mode Tables	3-13
 <b>Section 4. Transfer Conventions</b>	4-1
Request Block	4-3
Functional Parameters	4-5
Service-Specific Parameters	4-5
ABIOS Transfer Convention	4-13
Operating-System Transfer Convention	4-15
 <b>Section 5. Additional Information</b>	5-1
Interrupt Processing	5-3
Interrupt Flow	5-3
Interrupt Sharing	5-3
Default Interrupt Handler	5-5
Adding, Patching, Extending, and Replacing	5-6
Adapter-ROM Structure	5-7
RAM-Extension Structure	5-9
Adding	5-11

Patching .....	5-12
Extending .....	5-13
Replacing .....	5-15
Considerations for RAM Extensions .....	5-16
Operating-System Implementation Considerations .....	5-18
ABIOS Rules .....	5-18
Considerations for Bimodal Implementations .....	5-20

<b>Section 6. Interfaces</b> .....	6-1
Device ID 01H—Diskette .....	6-ID01-1
Device ID 02H—Fixed Disk .....	6-ID02-1
Device ID 03H—Video .....	6-ID03-1
Device ID 04H—Keyboard .....	6-ID04-1
Device ID 05H—Parallel Port .....	6-ID05-1
Device ID 06H—Asynchronous Communication .....	6-ID06-1
Device ID 07H—System Timer .....	6-ID07-1
Device ID 08H—Real-Time Clock .....	6-ID08-1
Device ID 09H—System Services .....	6-ID09-1
Device ID 0AH—Nonmaskable Interrupt (NMI) .....	6-ID0A-1
Device ID 0BH—Pointing Device .....	6-ID0B-1
Device ID 0EH—Nonvolatile Random Access Memory (NVRAM) .....	6-ID0E-1
Device ID 0FH—Direct Memory Access (DMA) .....	6-ID0F-1
Device ID 10H—Programmable Option Select (POS) .....	6-ID10-1
Device ID 16H—Keyboard Security .....	6-ID16-1
Device ID 17H—SCSI Subsystem Interface .....	6-ID17-1
Device ID 18H—SCSI Peripheral Type .....	6-ID18-1

---

# Section 1. Introduction to Advanced BIOS

Introduction .....	1-3
Data Structures .....	1-4
Initialization .....	1-5
Transfer Conventions .....	1-6
Interrupt Processing .....	1-7
Extending A BIOS .....	1-8

## **Notes:**

---

## Introduction

Advanced BIOS (ABIOS) is firmware that isolates an operating system from the low-level system hardware interface in IBM Personal System/2 systems that use 80286 or 80386 microprocessors. The operating system makes functional requests of ABIOS (read or write) instead of directly manipulating the I/O ports and control words of the system hardware. This enables details of the hardware attachments and the timings of the hardware interfaces to be altered without disturbing the operating-system components above the ABIOS interface.

ROM BIOS operates as a single-tasking component in which addressing capabilities are limited to less than 1 megabyte of memory and only in the real-address mode (real mode) of the Intel microprocessor. ABIOS supports addressing above 1 megabyte, using the protected virtual address mode (protected mode) of the Intel microprocessor. ABIOS is contained in ROM but does not prevent a RAM implementation. ABIOS can be operated in the real mode, in the protected mode, or in a bimodal environment using both the real mode and the protected mode. ABIOS provides a data structure for implementing a protected-mode or bimodal (real and protected modes) operating system. In addition, ABIOS can run in virtual 8086 mode.

Requests to ABIOS that are made by an operating system fall into three categories: single-staged, discrete multistaged, and continuous multistaged. Single-staged requests perform the requested function before returning to the caller. Discrete multistaged requests start an action or operation that involves a delay before the operation is completed. Continuous multistaged requests start an action or operation that also involves a delay but never ends. For multistaged operations, control is returned to the caller during these delays so that the processing time can be used. An interrupt from an I/O device usually indicates completion of a stage of the operation.

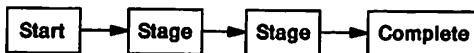
The following figure shows the three categories of BIOS requests.

**Single-Staged**

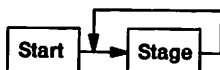
Start - Complete



**Discrete Multistaged**



**Continuous Multistaged**



**Figure 1-1. Types of Requests**

---

## **Data Structures**

Requests to BIOS that are made by an operating system are made through transfer conventions that are provided by the BIOS structure. Transfer conventions require data structures that link the operating system to the device-function routines of each supported device. These data structures are the common data area, function transfer tables, and device blocks. They reside in system memory and are initialized during BIOS initialization.

Transfer conventions that are provided by BIOS are defined to enable operations that use the real mode, the protected mode, or both modes of an Intel microprocessor. To provide flexibility in implementing a real-mode, protected-mode, or bimodal operating system, the common data area links all BIOS pointers into a single structure (see "Common Data Area" on page 2-3). This structure contains the function-transfer-table pointers, the device-block pointers, and the BIOS data pointers.

BIOS entry points are stored in vector tables called function transfer tables (see "Function Transfer Table" on page 2-7). Each supported BIOS device has an associated function transfer table. The first three entries of the function transfer table are structured entry routines: the Start routine, the Interrupt routine, and the Time-Out routine.



ABIOS routines require a permanent work area, called a device block, for each device (see "Device Block" on page 2-9). Hardware port addresses, interrupt levels, and device-status information are stored in a device block.

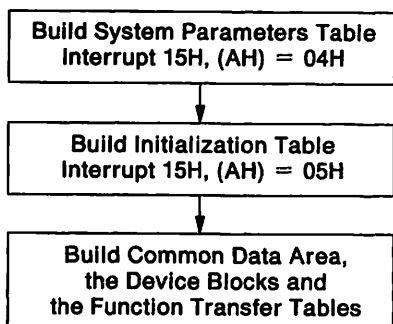
---

## **Initialization**

Initialization is a defined protocol between BIOS and an operating system. The operating system plays a major role in the initialization process, including starting the process. Until the operating system starts the initialization process, BIOS cannot be used (see Section 3, "Initialization"). The initialization process must occur in the real mode of the microprocessor. It consists of three steps:

1. The operating system calls BIOS to build the system parameters table. This table describes the number of devices that are available in the system, common entry points, and system-stack requirements.
2. The operating system calls BIOS to build the initialization table. This table defines the initialization information for each device that the system supports. This information is used to initialize device blocks and function transfer tables.
3. The operating system allocates memory for the common data area, using the initialization information that was returned in step 2. The memory for device blocks and function transfer tables is allocated, and the device-block pointers and function-transfer-table pointers are initialized in the common data area. Then the operating system calls BIOS to build a device block and function transfer table for each device.

The flow of the initialization process is shown below.



*Figure 1-2. Flow of the Initialization Process*

---

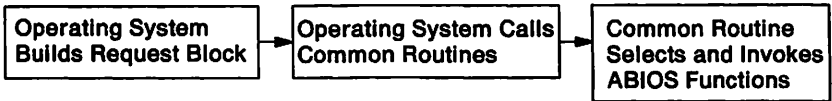
## Transfer Conventions

After BIOS is initialized, requests are presented through a parameter block, called a request block. A request block has fields that identify the target device, the requested operation, details of the request, memory locations that are involved in a data transfer, and the status of the staged or completed request. Request blocks are described in detail in Section 4, "Transfer Conventions."

BIOS is implemented as a call-return programming model, using either the BIOS transfer convention or the operating-system transfer convention. These two calling conventions give an operating system flexibility in calling BIOS. Both calling conventions use stacks to pass request information to the target BIOS device routine.

The BIOS transfer convention is the simplest calling sequence for the operating system. The operating system passes the common-data-area pointer and the request-block pointer to one of three common entry points: the Common Start routine, the Common Interrupt routine, and the Common Time-Out routine. The pointers to these common routines are returned to the operating system during initialization. The common routines use the request-block information and the common-data-area pointer to get the device-block pointer and the function-transfer-table pointer from the common data area. The common routine then transfers control to the requested BIOS routine whose pointer is in the function transfer table.

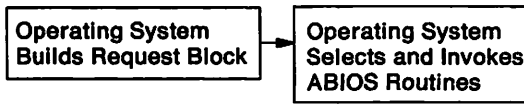
The flow of the BIOS transfer convention is shown below.



*Figure 1-3. Flow of BIOS Transfer Convention*

The operating-system transfer convention requires the operating system to determine the address for the requested BIOS routine. This gives the operating system flexibility in maintaining BIOS-routine addresses that are frequently called. This method is useful for handling interrupts from character and programmed-I/O devices that repeatedly call a single routine. The common-data-area pointer, the request-block pointer, the function-transfer-table pointer, and the device-block pointer are required on entry to the BIOS routine.

The flow of the operating-system transfer convention is shown below.



*Figure 1-4. Flow of Operating-System Transfer Convention*

The BIOS transfer convention and the operating-system transfer convention are described in detail in Section 4, "Transfer Conventions."

---

## Interrupt Processing

For multistaged requests, interrupts from hardware devices cause the microprocessor to branch to predefined addresses in the interrupt vector table. When an interrupt occurs, BIOS expects the operating system to receive control. BIOS provides interrupt routines for the processing of BIOS interrupts. Interrupt processing is described in "Interrupt Processing" on page 5-3.

---

## **Extending BIOS**

The ability to add, patch, extend, and replace BIOS routines is necessary for supporting new devices or device features on the system. For more information, see Section 5, "Additional Information."

---

# Section 2. Data Structures

Introduction .....	2-3
Common Data Area .....	2-3
Function Transfer Table .....	2-7
Device Block .....	2-9

## Notes:

---

## **Introduction**

ABIOS uses data structures to link the operating system to the device-function routines for each ABIOS device. These data structures are the common data area, the function transfer table, and the device block. They reside in system memory and are initialized during ABIOS initialization.

Transfer conventions that are provided by ABIOS are defined to enable operations that use the real mode, the protected mode, or both modes of an Intel microprocessor. ABIOS provides the common data area for implementing a real-mode, protected-mode, or bimodal operating system. This structure contains the function-transfer-table pointers, the device-block pointers, and the ABIOS data pointers. The common data area links all ABIOS pointers in a single structure to enable an operating system to manage ABIOS requests in both operating environments of the Intel microprocessors.

---

## **Common Data Area**

The common data area contains data pointers that facilitate the ABIOS operation in a bimodal environment. These data pointers are established during ABIOS initialization and contain information for each device that the system supports. The common data area is required in all three operating modes.

Each ABIOS device has a physical-device identifier, called a device ID. A device ID has one or more logical IDs that serve as device handlers and are used by the operating system to make requests of ABIOS. The common data area is made up of two arrays: an array of logical-ID entries and an array of data-pointer entries. Each logical-ID entry contains a pointer to a device block and a pointer to a function transfer table. Each data-pointer entry contains memory addresses that are used by ABIOS services.

On each request to ABIOS, a segment or selector that has an assumed offset of 0 and points to the common data area is passed to ABIOS. This pointer is referred to as the anchor pointer to the common data area.

The following diagram shows the common data area and its relationship to the other BIOS data structures.

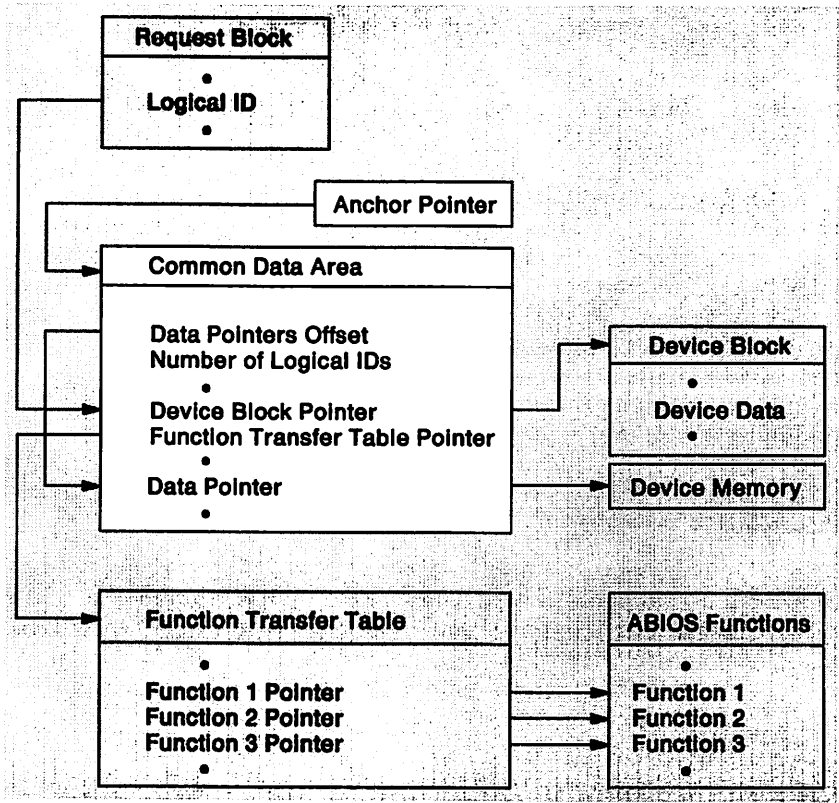


Figure 2-1. Flow of Common Data Area



The following figure shows the format of the common data area.

Size	Offset	Description
Word	00H	Offset to data pointer 0
Word	02H	Count of logical IDs
DWord	04H	Reserved
DWord	08H	Device-block pointer logical ID
DWord	0CH	Function-transfer-table pointer logical ID 1
DWord	10H	Device-block pointer logical ID 2
DWord	14H	Function-transfer-table pointer logical ID 2
.	.	.
.	.	.
.	.	.
DWord	$08H \times n$	Device-block pointer logical ID $n$
DWord	$(08H \times n) + 04H$	Function-transfer-table pointer logical ID $n$
Word	$(08H \times n) + 08H$	Data pointer $p$ length
Word	$(08H \times n) + 0AH$	Data pointer $p$ offset
Word	$(08H \times n) + 0CH$	Data pointer $p$ segment
Word	$(08H \times n) + 0EH$	Data pointer $p-1$ length
Word	$(08H \times n) + 10H$	Data pointer $p-1$ offset
Word	$(08H \times n) + 12H$	Data pointer $p-1$ segment
.	.	.
.	.	.
.	.	.
Word	$(08H \times n) + (06H \times p) + 08H$	Data pointer 0 length
Word	$(08H \times n) + (06H \times p) + 0AH$	Data pointer 0 offset
Word	$(08H \times n) + (06H \times p) + 0CH$	Data pointer 0 segment
Word	$(08H \times n) + (06H \times p) + 0EH$	Data pointer count

$n$  = the number of logical IDs  
 $p$  = the number of data pointers minus 1

Figure 2-2. Common Data Area

The common-data-area entries are:

**Offset to Data Pointer 0:** This field, combined with the anchor pointer, produces a pointer to the Data Pointer 0 Length field.

**Count of Logical IDs:** This field contains the number of device-block and function-transfer-pointer pairs.

**Device-Block Pointers:** These fields contain the pointers to the device blocks for the specified logical IDs.

**Function-Transfer-Table Pointers:** These fields contain the pointers to the function transfer tables for the specified logical IDs.

**Data-Pointer Lengths:** These fields contain the lengths of the data areas that are pointed to by the associated data pointer.

**Data-Pointer Offsets:** These fields contain the offsets of the data areas. Each offset is combined with its associated data-pointer segment to produce a pointer to the data area.

**Data-Pointer Segments:** These fields contain the segments of the data areas. Each segment is combined with its associated data-pointer offset to produce a pointer to the data area.

**Data-Pointer Count:** This field contains the number of data pointers.

If the Function-Transfer-Table Pointer field and the Device-Block Pointer field are both set to hex 0:0 after BIOS initialization, the associated logical ID is disregarded by the operating system as a null common-data-area entry. The entry is used as a temporary placeholder during initialization for BIOS extensibility. For more information, refer to Section 5, "Additional Information."

## Function Transfer Table

ABIOS entry points are stored in vector tables, called function transfer tables. These tables contain the doubleword address pointer for each ABIOS function. Reserved function pointers are initialized to hex 0:0. Each logical ID (entry in the common data area) has a function-transfer-table pointer. Multiple logical IDs can have function-transfer-table pointers that point to the same function transfer table.

The following figure shows a function transfer table and its relationship to the common data area.

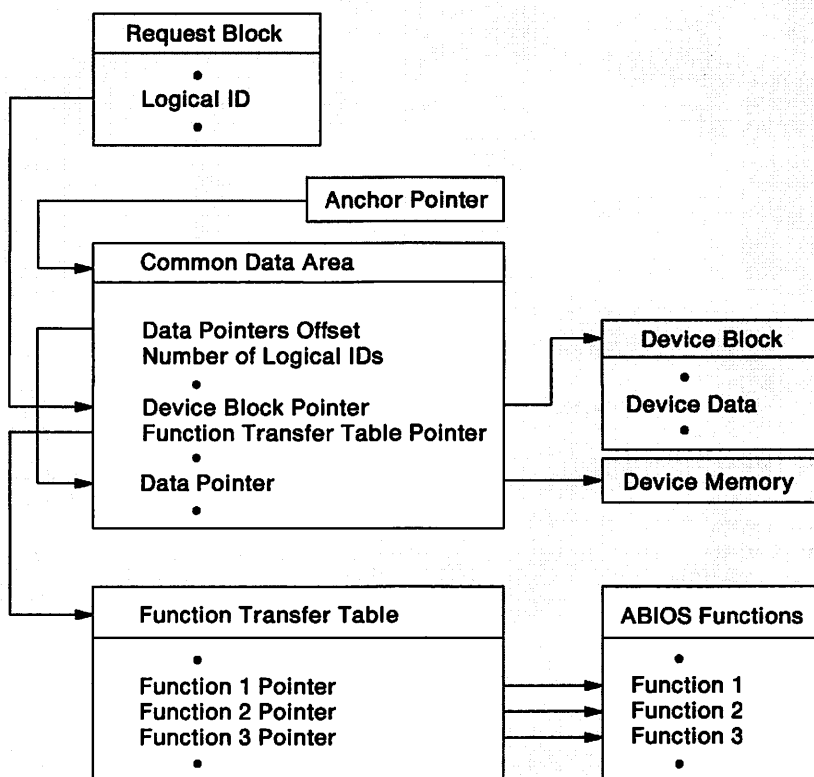


Figure 2-3. Flow of Function Transfer Table

The operating system builds a request block, including the logical ID, which defines the device, and the function. Based on the information in the request block, the function-transfer-table pointer and device-block pointer can be located in the common data area for the requested device. The operating system uses the function-transfer-table pointer to start requests, process interrupts, and handle any time-outs that occur. Each pointer in the function transfer table is a doubleword pointer to a Function routine.

The following figure shows the format of the function transfer table.

Size	Offset	Description
DWord	00H	Start-routine pointer
DWord	04H	Interrupt-routine pointer
DWord	08H	Time-Out routine pointer
Word	0CH	Function count
Word	0EH	Reserved
DWord	10H	Function 1 routine pointer
DWord	14H	Function 2 routine pointer
.	.	.
.	.	.
.	.	.
DWord	$(4 \times n) + 0CH$	Function $n$ routine pointer


$n$  = the number of functions

**Figure 2-4. Function Transfer Table**

The function-transfer-table entries are:

**Start-Routine Pointer:** The Start-routine pointer is a doubleword pointer. It is called (using Call Far Indirect) to start a request. This routine validates the Function field, the Request-Block Length field, and the Unit field. All registers are saved and restored across a call to this routine.

**Interrupt-Routine Pointer:** The Interrupt-routine pointer is a doubleword pointer. It is called (using Call Far Indirect) to resume a multistaged request upon indication from the hardware. All multistaged requests are resumed through this routine if the operation is not complete. All registers are saved and restored across a call to this routine. If this function transfer table corresponds to a device that does not interrupt, the Interrupt-Routine Pointer field is initialized to hex 0:0.



**Time-Out Routine Pointer:** The Time-Out routine pointer is a doubleword pointer. It is called (using Call Far Indirect) to terminate a request that fails to receive a hardware interrupt within a specified length of time. This routine terminates the request and leaves the hardware controller in a known, initial state. All registers are saved and restored across a call to this routine. If this function transfer table corresponds to a device that does not interrupt, or to a device that interrupts but never times out, the Time-Out Routine Pointer field is initialized to hex 0:0.

**Function Count:** This is a word count of the number of functions that are supported by a device.

**Reserved:** This is a reserved word (allocated regardless of whether the value of the Function Count field is 0).

**Function 1 Pointer:** This is a doubleword pointer to the Function 1 routine.

**Function 2 Pointer:** This is a doubleword pointer to the Function 2 routine.




**Function *n* Pointer:** This is a doubleword pointer to the Function *n* routine.

For more information, see “Functional Parameters” on page 4-5.

---

## Device Block

ABIOS routines require a permanent work area, called a device block, for each device. Hardware port addresses, interrupt levels, and device-status information are stored in a device block.



The device block contains both public and private data. The public data in the device block is a readable area whose format is common across all device blocks. The operating system must not alter this area. Private data in the device block is used internally by BIOS. Its format and content are not necessarily identical in all implementations of BIOS. The operating system must not examine or alter private data, and IBM reserves the right to alter the contents of the private portion of the device block.

The following figure shows a device block and its relationship to the common data area.

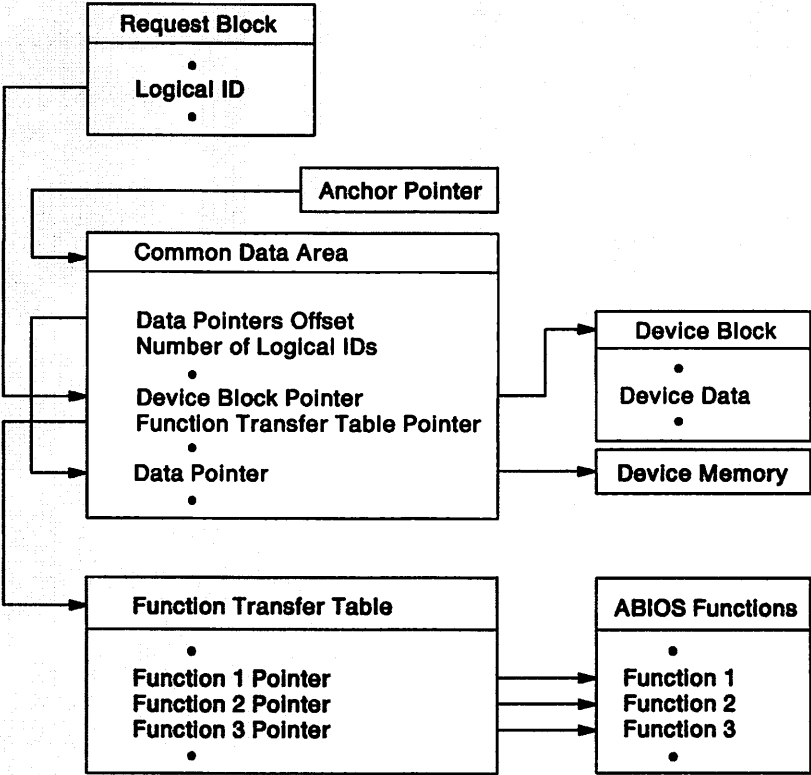


Figure 2-5. Flow of Device Block

Every BIOS device has an associated device block. The device-block format is shown in the following figure.

Size	Offset	Description	Access
Word	00H	Device-block length	Public read
Byte	02H	Revision	Public read
Byte	03H	Secondary device ID	Public read
Word	04H	Logical ID	Public read
Word	06H	Device ID	Public read
Word	08H	Count of logical-ID exclusive-port pairs	Public read
Word	0AH	Count of logical-ID common-port pairs	Public read
DWord	?	Logical-ID exclusive-port pairs 0	Public read
DWord	?	Logical-ID exclusive-port pairs 1	Public read
.	.	.	
.	.	.	
DWord	?	Logical-ID exclusive-port pairs <i>n</i>	Public read
DWord	?	Logical-ID common-port pairs 0	Public read
DWord	?	Logical-ID common-port pairs 1	Public read
.	.	.	
DWord	?	Logical-ID common-port pairs <i>n</i>	Public read
Word	?	Device-unique data-area length	Private
?	?	Device-unique data area	Private
Word	?	Count of units	Private
Word	?	Unit-unique data-area length	Private
?	?	Unit-unique data area	Private

? = placeholder for variable values  
*n* = the number of port pairs

**Figure 2-6. Device Block**

The device-block entries are:

**Device-Block Length:** This field is one word long, and it contains the number of bytes that are in the device block, including the Device-Block Length field. The maximum specifiable length is 64KB minus 1 byte. The required device-block size for a particular device is returned during BIOS Initialization.

**Revision:** This byte indicates the level of the supporting code for a device. The initial value of the base level is 0. The value of the Revision field is increased by 1 for each succeeding version of BIOS code for a particular device ID and secondary device ID (that is, when a new level of BIOS code is developed for existing hardware).

**Secondary Device ID:** This byte indicates the level of hardware that an BIOS implementation supports. The initial value of the base level is 0. The value of the Secondary Device ID field is increased by 1 when a new level of code is developed for a previously-defined device ID that supports new hardware. When the value of the Secondary Device ID field is increased, the Revision field is reset to 0.

**Logical ID:** This field indicates the logical name of the device that is associated with a device block. It is analogous to the software-interrupt number that BIOS uses to access different device types. Logical ID values are determined dynamically during BIOS initialization; the logical ID for a given device is determined by the index of its entry in the common data area.

To facilitate the patching of common internal BIOS functions, the operating system must reserve a number of logical IDs to enable BIOS to call these common internal BIOS functions. They are identified to the operating system during BIOS initialization. The logical-ID values are shown in the following figure.

Logical ID	Usage
00H	Reserved
01H	Reserved
02H to nH	BIOS internal calls
>n	System and adapter devices
n = the last logical ID reserved for BIOS internal calls	

Figure 2-7. Logical ID Values

**Device ID:** This field indicates the type of device that is addressed by a function request and the BIOS function level that is supported. The assigned values of this field are shown in the following figure.



Device ID	Device
00H	ABIOS Internal Calls
01H	Diskette
02H	Disk
03H	Video
04H	Keyboard
05H	Parallel Port
06H	Asynchronous Communication
07H	System Timer
08H	Real-Time Clock Timer
09H	System Services
0AH	Nonmaskable Interrupt
0BH	Pointing Device
0CH	Reserved
0DH	Reserved
0EH	Nonvolatile Random Access Memory (NVRAM)
0FH	Direct Memory Access (DMA)
10H	Programmable Option Select (POS)
11H to 15H	Reserved
16H	Keyboard Security
17H	SCSI Subsystem Interface
18H	SCSI Peripheral
19H to FFFFH	Reserved

*Figure 2-8. Device ID Values*

Device ID hex 00 is reserved for BIOS internal calls (an BIOS function calling another BIOS function). Each of the other device-ID values denotes a type of device and a level of BIOS support, as described in the "Interfaces" section.

**Count of Logical-ID Exclusive-Port Pairs:** This is the number of logical-ID exclusive-port pairs. A logical-ID exclusive port is a port that is used exclusively by a particular logical ID. Examples are the diskette ports, disk ports, asynchronous communication ports, parallel ports, and video ports. If the value of the Count of Logical-ID Exclusive-Port Pairs field is 0, no space is allocated for the Logical-ID Exclusive-Port Pairs fields.

**Count of Logical-ID Common-Port Pairs:** This is the number of logical-ID common-port pairs. Logical-ID common ports are ports that are shared across more than one logical-ID value. Examples are the DMA-controller ports, keyboard-controller ports, and NVRAM ports. Each logical ID that uses one of these ports contains an entry in the Logical-ID Common-Port Pairs fields of the device block. If the value of this field is 0, no space is allocated for the Logical-ID Common-Port Pairs fields.

**Logical-ID Exclusive-Port Pairs:** These are the logical-ID exclusive-port pairs. The first word of the doubleword is the starting I/O-port number of a range of I/O-port numbers. The second word is the ending I/O-port number of the range.

**Logical-ID Common-Port Pairs:** These are the logical-ID common-port pairs. The first word of the doubleword is the starting I/O-port number of a range of I/O-port numbers. The second word is the ending I/O-port number of the range.

**Note:** Every port that an BIOS logical ID reads from or writes to is contained in either the Logical-ID Exclusive-Port Pairs fields or the Logical-ID Common-Port Pairs fields.

**Device-Unique Data-Area Length:** This field contains the length, in bytes, of the device-unique data area for the specified device.

**Device-Unique Data Area:** This field contains data that is unique to a device. Parameters that describe the device and working data that span the device ID are kept in this area. This area contains private data for BIOS; its content and format might change. Examples of the data that is kept in this area are interrupt level, arbitration level, and device status.

**Count of Units:** This field contains the number of unit-unique data areas in the device block. If the value of this field is 0, the Count of Units field is the last field in the device block.

**Unit-Unique Data-Area Length:** This field contains the length, in bytes, of a single entry in the repeatable unit-unique data area, excluding the Unit-Unique Data-Area Length field. This field exists only when the value of the Count of Units field is greater than 0.

**Unit-Unique Data Area:** This field is a private area that is repeatable for each unit of the specified device ID. For example, if the device ID value is hex 01 (diskette), a particular diskette drive is considered a unit. Parameters that describe the unit and working data that span individual requests are kept in this area. This area contains private data for BIOS; its content and format might change. This field exists only when the value of the Count of Units field is greater than 0.

---

## Section 3. Initialization

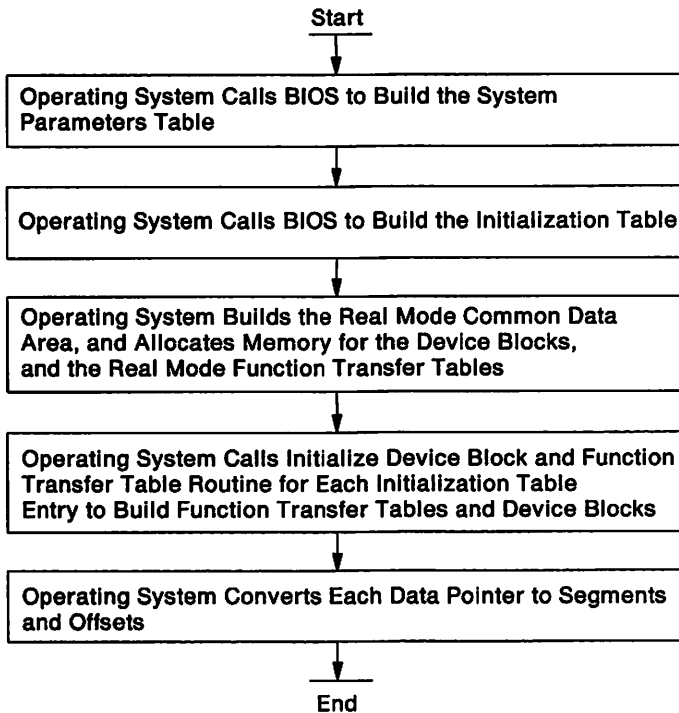
Introduction .....	3-3
Build System Parameters Table—Operating System .....	3-4
Build System Parameters Table—BIOS .....	3-4
Build Initialization Table—Operating System .....	3-6
Build Initialization Table—BIOS .....	3-6
Build Common Data Area—Operating System .....	3-8
Initialize Pointers—Operating System .....	3-9
Initialize Data Structures—ABIOS .....	3-10
Logical ID 2 Initialization .....	3-12
Build Protected-Mode Tables .....	3-13

## Notes:

---

## Introduction

ABIOS is initialized on demand. The operating system makes specific calls to BIOS and ABIOS to initialize ABIOS. The real-mode common data area must be initialized before any requests can be made to ABIOS. Initialization must be performed in the real mode of the microprocessor. It includes building the system parameters table, the initialization table, and the common data area. The following diagram shows the flow of the real-mode common-data-area initialization.



**Figure 3-1. Flow of Real-Mode Common-Data-Area Initialization**

---

## **Build System Parameters Table—Operating System**

The operating system allocates an area of hex 20 bytes and calls BIOS to build the system parameters table. This table describes the number of devices that are available in the system, the BIOS common entry points, and system-stack requirements.

### **Interrupt 15H—System Services**

#### **(AH) = 04H—Build System Parameters Table**

Invocation: Software interrupt, operating system calls BIOS

(ES:DI) - Pointer to caller's memory where system parameters table is to be built

(DS) - Segment with assumed offset of hex 0 to RAM extension area (points to a RAM extension with a length of 0 for no RAM extensions)

On Return:

(AH) = 0 - Operation successfully completed

CF = 1 - Exception error

All registers except (AX) and the flags are restored.

*Figure 3-2. Build System Parameters Table BIOS Function*

---

## **Build System Parameters Table—BIOS**

When it is called by the operating system, BIOS builds the system parameters table. The Number of Entries field is established from the system-board ROM, adapter ROMs, and RAM extensions. To accumulate the number of entries, configuration information is obtained from system-equipment data areas, from NVRAM, and possibly, by presence testing for devices whose operating code resides in the system-board ROM.

For devices that have code in an adapter ROM, an extension of the power-on self-test (POST) ROM scan determines the number of entries that the adapter ROM requires (see "Adapter-ROM Structure" on page 5-7).

For devices that have code in the RAM-extension area, the RAM-extension scan determines the number of initialization-table entries that the RAM extension requires (see "RAM-Extension Structure" on page 5-9).

When the system-parameters-table information has been obtained, the memory that is allocated for this table can be deallocated and reused by the operating system. The format of the system parameters table is shown below.

Size	Offset	Description
DWord	00H	Common Start routine pointer
DWord	04H	Common Interrupt routine pointer
DWord	08H	Common Time-out routine pointer
Word	0CH	Stack required
DWord	0EH	Reserved
DWord	12H	Reserved
DWord	16H	Reserved
DWord	1AH	Reserved
Word	1EH	Number of entries

*Figure 3-3. System Parameters Table*

The system-parameters-table entries are:

**Common Start Routine Pointer:** This is a doubleword address pointer to the Common Start routine entry point.

**Common Interrupt Routine Pointer:** This is a doubleword address pointer to the Common Interrupt routine entry point.

**Common Time-Out Routine Pointer:** This is a doubleword address pointer to the Common Time-Out routine entry point.

**Stack Required:** This field is a word that contains the amount of stack memory, in bytes, that is required for a particular ABIOS implementation.

**Number of Entries:** This field is a word that contains the number of entries that are required in the initialization table.

---

## Build Initialization Table—Operating System

The initialization table defines the initialization information for each device that the system supports. This information is used to initialize the device blocks and the function transfer tables.

The operating system allocates memory and calls BIOS to build the initialization table. The amount of memory, in bytes, that is required for the initialization table is the number of entries in the initialization table multiplied by hex 18. The Number of Entries field in the system parameters table is used for this calculation. When the initialization process is complete, the memory that was allocated for the initialization table can be deallocated and reused by the operating system.

### Interrupt 15H—System Services (AH)=05H—Build Initialization Table

Invocation: Software interrupt, operating system calls BIOS

(ES:DI) - Pointer to caller's memory where  
initialization table is to be built

(DS) - Segment with assumed offset of hex 0 to  
RAM extension area (points to a RAM extension  
with a length of 0 for no RAM extensions)

On Return:

(AH) = 0 - Operation successfully completed

CF = 1 - Exception error

All registers except (AX) and the flags are restored.

*Figure 3-4. Build Initialization Table BIOS Function*

---

## Build Initialization Table—BIOS

BIOS builds the initialization table. This table is established from the system-board ROM, adapter ROMs, and RAM extensions. For devices that have code in an adapter ROM, an extension of the power-on self-test (POST) ROM scan is used. For more information, see "Adapter-ROM Structure" on page 5-7.

For devices that have code in the RAM-extension area, the RAM-extension scan is used (see "RAM-Extension Structure" on page 5-9). All system-board ABIOS-device initialization-table entries precede any adapter-ROM or RAM-extension device entries. The



initialization-table structure, shown in the following figure, is repeated for each entry.

Size	Offset	Description
Word	00H	Device ID
Word	02H	Number of logical IDs
Word	04H	Device-block length
DWord	06H	Initialize Device Block and Function Transfer Table routine pointer
Word	0AH	Request-block length
Word	0CH	Function-transfer-table length
Word	0EH	Data-pointers length
Byte	10H	Secondary device ID
Byte	11H	Revision
Word	12H	Reserved
Word	14H	Reserved
Word	16H	Reserved

*Figure 3-5. Initialization Table*

The initialization-table entries are:

**Device ID:** For a list of the values of the Device ID field, see Figure 2-8 on page 2-13. There can be more than one entry in the initialization table with the same device ID.

**Number of Logical IDs:** This is a word that contains the maximum number of devices that require individual device blocks but are operated by the same code. The Number of Logical IDs field tells the operating system the maximum number of logical IDs that this initialization-table entry allows.

**Device-Block Length:** This is a word that contains the length, in bytes, of the storage allocation that is required for the device block for this device. A device-block length of 0 indicates that this initialization-table entry is for an BIOS patch or extension, and no device block needs to be built (see "Adding, Patching, Extending, and Replacing" on page 5-6). When the device-block length is 0, the operating system ensures that the device-block pointer in the common data area is initialized to hex 0:0.

**Initialize Device Block and Function Transfer Table Routine Pointer:** This is a doubleword address pointer (real mode segment:offset) to the routine to initialize the device blocks and function transfer tables for an entry in the initialization table. This routine is also provided by adapter ROMs or RAM extensions to add, patch, extend, or replace

services (see “Adding, Patching, Extending, and Replacing” on page 5-6).

**Request-Block Length:** This is a word that contains the length, in bytes, of the storage allocation that is required for the request block for this device. When a request is made to BIOS, any request-block size that is greater than the returned size is valid.

**Function-Transfer-Table Length:** This is a word that contains the length, in bytes, of the function transfer table. A function-transfer-table length of 0 indicates that this initialization-table entry is for an BIOS patch, and no function-transfer-table data area is to be allocated. When the function-transfer-table length is 0, the operating system ensures that the Function-Transfer-Table Pointer field in the common data area is initialized to 0:0.

**Data-Pointers Length:** This is a word that contains the length, in bytes, of the storage allocation that is required for the Data Pointer fields in the common data area.

**Secondary Device ID:** This is a byte that is used to determine the level of hardware that an BIOS implementation supports. See “Device Block” on page 2-9 for more information.

**Revision:** This byte is used to indicate the level of the supporting code for this device. See “Device Block” on page 2-9 for more information.

---

## Build Common Data Area—Operating System

After the system parameters table and the initialization table are built, the operating system has all the necessary information that is required to build the common data area and its associated data structures (see “Data Structures” on page 1-4). The size of the common data area, the size of each function transfer table, and the size of each device block can be determined from the initialization table.

The operating system builds the common data area at offset hex 00 in a segment and allocates memory for each device block and function transfer table. Memory is allocated in the common data area for the data pointers. The offset to the Data Pointer 0 field is initialized to point to the Data Pointer Length 0 field in the common data area. The Data Pointer Count field is initialized to 0. The Count of Logical IDs

field is filled in with the number of device-block and function-transfer-table pointer pairs. Each device-block pointer and each function-transfer-table pointer is initialized to point to the memory that has been allocated.

Logical-ID values for physical devices are assigned by their order in the initialization table. For example, if the value of the Number of Logical IDs field is 1 for each entry in the initialization table, the first entry corresponds to logical ID 2, the second entry corresponds to logical ID 3, and so on. If the value of the Number of Logical IDs field is greater than 1 for the first initialization-table entry, that entry corresponds to logical ID 2 through logical ID 2 plus the value of the Number of Logical IDs field minus 1. The second initialization-table entry corresponds to the next succeeding logical ID.

Multiple function-transfer-table pointers can point to the same function transfer table. This occurs when the value of the Number of Logical IDs field in an initialization-table entry is greater than 1. The operating system must ensure that the function-transfer-table pointers for the succeeding logical IDs, which correspond to a single initialization-table entry, point to the same function transfer table.

---

## **Initialize Pointers—Operating System**

The operating system calls the Initialize Device Block and Function Transfer Table routine once for each entry in the initialization table. The operating system passes the anchor pointer, the starting logical ID, and the number of logical IDs that are to be initialized.

The Initialize Device Block and Function Transfer Table routines are called in the order in which their pointers appear in the initialization table. This causes system-board ROM devices to be initialized before any adapter ROM or RAM extension. The Initialize Device Block and Function Transfer Table routines for adapter-ROM devices and RAM extensions can then identify the system-board services that might be needed. This is accomplished by scanning the common data area, using the device ID in the public portion of the device block to identify the system-board service that is needed. When the device ID has been found, the logical-ID number that is in the public portion of the device block is used for all subsequent requests to the system-board BIOS service.

The operating system needs to call the Initialize Device Block and Function Transfer Table routine only for the devices that are to be made operational. The operating system can determine whether to

initialize an ABIOS device on the basis of the values in the Device ID field and the Secondary Device ID field in each initialization-table entry. For devices that are initialized, the operating system must ensure that each additional initialization-table entry that contains the same device-ID and secondary-device-ID values must also be initialized to allow for patching. Each initialization-table entry that contains a device ID of 0 must be initialized to ensure that internal ABIOS calls are supported. Also, there are device IDs that might need to be initialized to support other device IDs. For example, DMA ABIOS must be initialized if fixed-disk ABIOS is initialized. These requirements are defined in the "Interfaces" section.

When the value of the Number of Logical IDs field is greater than 1, the operating system can initialize any number of logical IDs up to and including the value of the Number of Logical IDs field.

Invocation: Call FAR; operating system calls ABIOS on system-board ROM, on adapter ROM, or on RAM extension, depending on the device.

(CX) - Number of logical IDs to be initialized (up to the value of the Number of Logical IDs field in the initialization table)

(DX) - Starting logical ID

(DS) - Anchor pointer to the common data area

On Return:

(AL) - Exception condition

= 00H - Operation successfully completed

= 01H to FFH - Device-initialization failure

All registers except (AX) are restored.


*Figure 3-6. Initialize Device Block and Function Transfer Table Routine*

---

## **Initialize Data Structures—ABIOS**

When the Initialize Device Block and Function Transfer Table routine is called, ABIOS fills in the function transfer table at the location that is defined by the function-transfer-table pointer of the Starting Logical ID parameter (DX).

For adapter ROMs or RAM extensions, when the Initialize Device Block and Function Transfer Table routine is called, each segment value that is placed in the function transfer table must equal the segment value of its corresponding ROM header or RAM-extension header. This allows an operating system to read the Length field of the ROM header or the RAM-extension header to determine the




segment limit in a bimodal or protected-mode environment. When the protected-mode common data area is being built, if offset hex 00 of the ROM-header segment or RAM-extension-header segment contains the ROM or RAM signature, offset hex 02 contains the length, in multiples of 512 bytes (the limit is hex 7F). This value is used to calculate the segment limit.

After the function transfer table is filled in, the Initialize Device Block and Function Transfer Table routine fills in the device block for the Starting Logical ID parameter (DX) and each succeeding logical ID, up to the value in the Number of Logical IDs to Be Initialized parameter (CX).


On return from the Initialize Device Block and Function Transfer Table routine, if the value in the Exception Condition parameter (AL) is nonzero, indicating an error, deallocate the associated device blocks and function-transfer-table areas and replace the associated device-block pointers and function-transfer-table pointers with hex 0:0, making those entries null common-data-area entries.

### **Data Pointers**



The Initialize Device Block and Function Transfer Table routine stores all the necessary BIOS data pointers in the data-pointer portion of the common data area. As the data pointers are stored, the value of the Data Pointer Count field is increased. The offset to the stored data pointer in the common data area can be stored in the device block as a handle to the data pointer.

BIOS initializes data pointers as 32-bit physical addresses that are stored in Intel data format (low word, high word) in the Data Pointer Offset field. Preceding this 32-bit physical address is the Data-Pointer Length field, which indicates the segment limit for a protected-mode or bimodal implementation. In bimodal implementations, a data-pointer value of hex 0:0 in the real-mode version of the common data area indicates an address above 1MB.



The operating system must translate the 32-bit physical address of each data pointer to a 16-bit offset and a 16-bit segment before making any requests to BIOS.

## Logical ID 2 Initialization

Reserved data pointers are initialized by a call to the Initialize Device Block and Function Transfer Table routine for Logical ID 2. Logical ID 2 is reserved for BIOS internal calls (device ID hex 00).

The following is a list of the reserved data pointers.

Data-Pointer Number	Value (Physical)	Limit	Description
0	400H	0100H	BIOS data area
1	E0000H	FFFFH	First 64KB of system-board ROM
2	F0000H	FFFFH	Second 64KB of system-board ROM

*Figure 3-7. Reserved Data Pointers*

These data pointers allow a single common data pointer to be used by multiple BIOS devices instead of duplicating the same data pointer multiple times.

In addition, a call to the Initialize Device Block and Function Transfer Table routine for logical ID 2 places the Common Start routine pointer, the Common Interrupt routine pointer, and the Common Time-Out routine pointer at the Start, Interrupt, and Time-Out pointers in the function transfer table for logical ID 2. The initialization-table entry for the first device of 0 must have a function-transfer-table length of at least hex 10 (greater if BIOS internal functions exist) to allow for the three doubleword pointers, a word count of functions, and a reserved field.

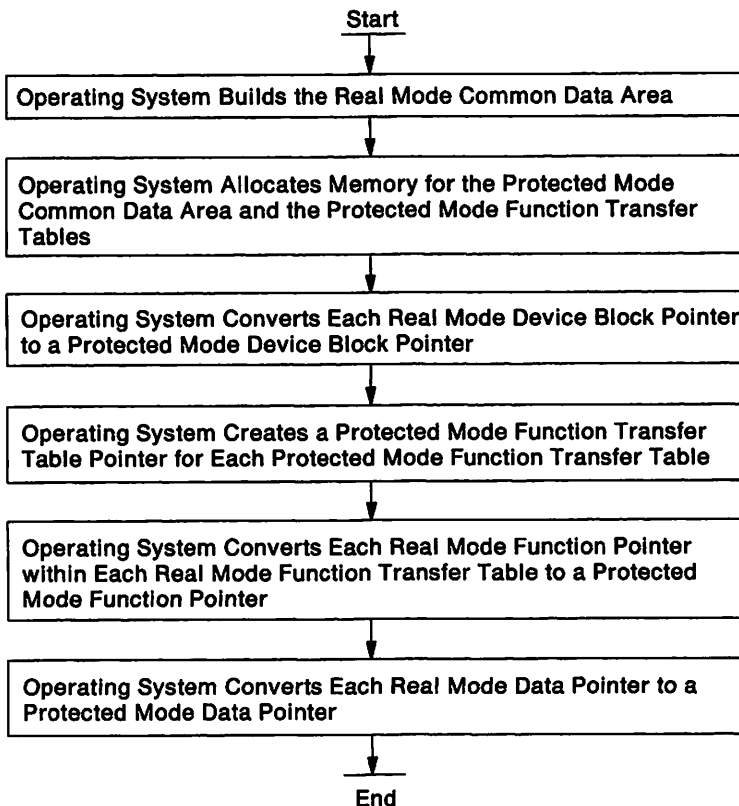
When the value of the Function Count field in the function transfer table for logical ID 2 is 0, the function transfer table has the following format:

Size	Offset	Description
DWord	00H	Common Start routine pointer
DWord	04H	Common Interrupt routine pointer
DWord	08H	Common Time-Out routine pointer
Word	0CH	Function count (set to 0)
Word	0EH	Reserved

*Figure 3-8. Function Transfer Table for Logical ID 2*

## Build Protected-Mode Tables

For protected-mode or bimodal implementations, it is necessary to build the protected-mode common data area and function transfer tables using the information that is built into the real-mode common data area and function transfer tables. The operating system must create selectors in the protected-mode common data area and function transfer tables whose effective addresses are identical to their corresponding segments in the real-mode common data area and function transfer tables. The following diagram illustrates the necessary steps to build the protected-mode common data area.



*Figure 3-9. Flow of Protected-Mode Common-Data-Area Initialization*

To build the descriptors that are associated with each selector, in addition to the physical address, the operating system needs to know the access rights and the segment limit of each segment.

The function-transfer-table pointers and the device-block pointers are writable data-segment descriptors whose expansion direction and limit are maintained by the operating system. The length of each of these tables is returned to the operating system through the Function-Transfer-Table Length field and the Device-Block Length field of the initialization table.

The selector of each data pointer must pertain to a writable data-segment descriptor whose expansion direction is up. The segment limit is determined by the Data Pointers Length field of each data-pointer entry in the common data area.

The pointers to BIOS functions in the function transfer table must be readable code-segment descriptors whose conforming bit is determined by the operating system. If offset hex 00 of the ROM-header segment or the RAM-extension-header segment contains the ROM or RAM signature, offset hex 02 contains the length, in multiples of 512 bytes (the limit is hex 7F). This value is to be used as the segment limit. If the ROM or RAM signature does not exist, the segment limit is hex FFFF.

If BIOS is called as a conforming code segment by multiple privilege levels, the operating system is responsible for ensuring that BIOS has I/O privilege at all times.

Each common-data-area entry in the protected-mode version must be a null common-data-area entry if its corresponding entry in the real-mode version is a null common-data-area entry. When the protected-mode version of each function transfer table is initialized, each entry in the protected-mode version that has a corresponding entry of hex 0:0 in the real-mode version must have a value of hex 0:0 to indicate that the function is not supported. The offset fields in the function transfer table must be the same for the corresponding entries in both tables. The device-block pointers for each logical-ID entry in both the real-mode and the protected-mode common data areas must point to the same device block.




---

## Section 4. Transfer Conventions

Request Block .....	4-3
Functional Parameters .....	4-5
Service-Specific Parameters .....	4-5
ABIOS Transfer Convention .....	4-13
Operating-System Transfer Convention .....	4-15

**Notes:**







ABIOS can be implemented in three environments: protected mode only, real mode only, and bimodal. BIOS requires a method of transferring control from the caller of BIOS to BIOS without sacrificing performance. The two methods that are provided for this transfer are the BIOS transfer convention and the operating-system transfer convention. Both of these conventions use the request block as the method by which an operating system communicates with and passes parameters to BIOS.

---

## **Request Block**

The request block is a parameter block that is used to communicate information bidirectionally between the caller and an BIOS service. Parameters are passed by the caller (IN) and returned by BIOS (OUT).



The following diagram shows the request block and its relationship to a common data area.

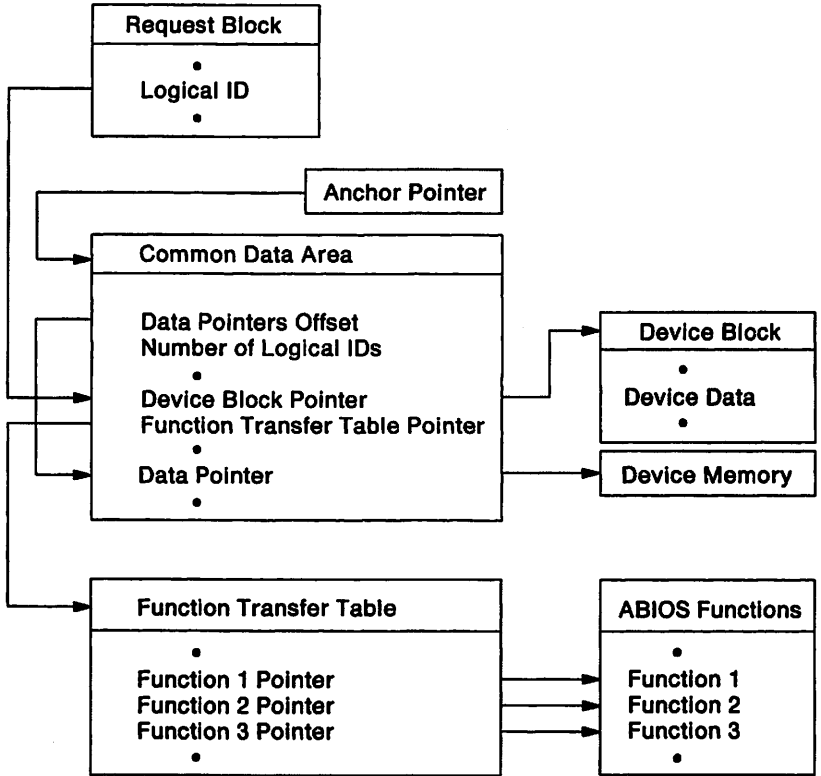


Figure 4-1. Flow of Request Block

Input parameters (IN) are not altered by BIOS during a request. Output parameters (OUT) and work areas do not need to be set to any predefined values before BIOS is called. This allows request blocks to be reused after requests are completed, but it requires that any Work Area fields that contain request-state information be initialized by the BIOS Start routine to the predefined values. Only input (IN) or input/output (IN/OUT) parameters that change between requests need to be initialized before the request block is reused.

All reserved input fields must be set to 0 by the caller of BIOS.

The parameters are divided into two categories: functional parameters and service-specific parameters.

## Functional Parameters

Functional parameters are common to all BIOS-service requests. They convey information to BIOS about which service should be invoked on which device. Each input parameter is initialized by the caller, and when it is initialized, it must remain unaltered until the requested operation is complete. The functional parameters are the Request-Block Length field through the Time-Out field, as shown in Figure 4-2.

## Service-Specific Parameters

Service-specific parameters are specific to BIOS requests. The details of the parameters that are passed by the caller and parameters that are returned by BIOS depend on the service that has been requested. The service-specific parameters are the Data Pointer 1 field through the Work Area field, as shown in Figure 4-2.

### Request-Block Structure

The structure of a request block that contains functional parameters and service-specific parameters is shown below.

#### Functional Parameters:

Word	00H	Request-block length (IN)
Word	02H	Logical ID (IN)
Word	04H	Unit (IN)
Word	06H	Function (IN)
Word	08H	Reserved
Word	0AH	Reserved
Word	0CH	Return code (IN/OUT)
Word	0EH	Time-out (OUT)

#### Service Specific Parameters:

Word	10H	Reserved
DWord	12H	Data pointer 1 (IN)
Word	16H	Reserved
Word	18H	Reserved
DWord	1AH	Data pointer 2 (IN)
?	1EH	Parameters (IN/OUT)
?	?	Work area

? = undefined initial value

Figure 4-2. Request Block

**Request-Block Length (IN):** The Request-Block Length field contains the length, in bytes, of the request block, including the Request-Block Length field itself. The maximum specifiable length is 64KB minus 1 byte. The Request-Block Length field contains a fixed value that is initialized by the caller for the specific logical ID. The size of the request block for a logical ID is returned by the Return Logical ID Parameters function (hex 01) when BIOS is initialized. However, the request block can be larger than the returned size.

**Logical ID (IN):** The Logical ID field indicates the particular device that is addressed by a function request. It is analogous to a software-interrupt number that is used by BIOS to access different device types.

**Unit (IN):** The Unit field is a parameter that addresses a particular unit of a device type within a logical ID. The range of valid values is limited by the number of units that are attached to a single controller. The maximum unit number is  $n-1$ , where  $n$  is the number of units that are attached to the controller. The minimum number of units is 1, which causes the value of the Unit field to be 0.

**Function (IN):** The Function field is a parameter that is used to request a particular category of operation. The assignment of functions is as follows.

**Function hex 00—Default Interrupt Handler:**

This function is called, with no service-specific parameters, for each logical ID by way of the Interrupt routine. The request block for the default interrupt handler has a fixed length of hex 10 bytes, and the Return Code field is updated on return with hex 0000 (Operation Successfully Completed) or hex 0005 (Not My Interrupt). For more information on the default interrupt handler, see “Default Interrupt Handler” on page 5-5.

**Function hex 01—Return Logical ID Parameters:**

This is a single-staged function that is common to all ABIOs device IDs. It returns information pertaining to the logical ID. Its request block has a fixed length of hex 20 bytes.

This function returns the following parameters.

**Service-Specific Input**

Size	Offset	Description
Word	1AH	Reserved
Word	1CH	Reserved
Word	1EH	Reserved

## Service-Specific Output

Size	Offset	Description
Word	0CH	Return code
Byte	10H	Hardware interrupt level <ul style="list-style-type: none"> <li>= FDH - Interrupt level not available</li> <li>= FEH - Special case for NMI</li> <li>= FFH - Noninterrupting logical ID</li> </ul>
Byte	11H	Arbitration level <ul style="list-style-type: none"> <li>= FDH - Arbitration level not available</li> <li>= FEH - Two arbitration levels are available (see offset hex 1C)</li> <li>= FFH - Not applicable</li> </ul>
Word	12H	Device ID
Word	14H	Count of units
Word	16H	Logical-ID flags <ul style="list-style-type: none"> <li>Bits 15 to 6 - Reserved (set to 0)</li> <li>Bit 5 - Address-limited indicator for DMA devices <ul style="list-style-type: none"> <li>= 0 - Address capability limited to 16MB</li> <li>= 1 - Address capability limited to 4GB</li> </ul> </li> <li>Bit 4 - Generic SCSI disk support availability <ul style="list-style-type: none"> <li>= 0 - Not available</li> <li>= 1 - Available</li> </ul> </li> <li>Bit 3 - Overlapped I/O across units <ul style="list-style-type: none"> <li>= 0 - Not supported</li> <li>= 1 - Supported</li> </ul> </li> <li>Bit 2 - 32-bit offset <ul style="list-style-type: none"> <li>= 0 - Not enabled</li> <li>= 1 - Enabled</li> </ul> </li> <li>Bits 1, 0 - Function read/write/additional-data-transfer data-pointer mode <ul style="list-style-type: none"> <li>= 00 - No read/write/additional-data-transfer functions are supported</li> <li>= 01 - Data pointer 1, logical</li> <li>      Data pointer 2, reserved</li> <li>= 10 - Data pointer 1, reserved</li> <li>      Data pointer 2, physical</li> <li>= 11 - Data pointer 1, logical</li> <li>      Data pointer 2, physical</li> </ul> </li> </ul>
Word	18H	Request-block length (for functions other than Default Interrupt Handler and Return Logical ID parameters; variable by logical ID)
Byte	1AH	Secondary device ID
Byte	1BH	Revision
Word	1CH	First and second arbitration levels (valid only when the Arbitration Level field is set to hex FE) <ul style="list-style-type: none"> <li>Bits 7 to 4 - Second arbitration level</li> <li>Bits 3 to 0 - First arbitration level</li> </ul>
Word	1EH	Reserved

The logical ID flags contain 2 bits that indicate the mode (physical or logical) of the data pointer for the Read function (hex 08), the Write function (hex 09), and the Additional Data Transfer function (hex 0A). If this parameter indicates that the pointer should be a logical pointer, data pointer 1 is a

logical pointer, and data pointer 2 is reserved. If this parameter indicates that the pointer should be a physical pointer, data pointer 2 is a physical pointer, and data pointer 1 is reserved. If this parameter indicates that both a logical pointer and a physical pointer are to be passed, data pointer 1 is a logical pointer, and data pointer 2 is a physical pointer. If this parameter indicates that neither a logical pointer nor a physical pointer is to be passed, either this logical ID does not support the Read, Write, and Additional Data Transfer functions, or these functions do not require address pointers. In this case, no space is reserved for data pointers in the request block.

**Function 02H—Reserved**

**Function 03H—Read Device Parameters:**

Device-specific parameters are returned.

**Function 04H—Set Device Parameters:**

Device-specific parameters are set.

**Function 05H—Reset/Initialize:**

The device is put into a known state.

**Function 06H—Enable:**

The device is enabled for interrupts (not at an interrupt controller).

**Function 07H—Disable:**

The device is disabled for interrupts (not at an interrupt controller).

**Function 08H—Read:**

Data is transferred from the device to memory. The data-pointer mode is determined by the Return Logical ID Parameters function (hex 01).

**Function 09H—Write:**

Data is transferred from memory to the device. The data-pointer mode is determined by the Return Logical ID Parameters function (hex 01).

**Function 0AH—Additional Data Transfer:**

The data-pointer mode is determined by the Return Logical ID Parameters function (hex 01).



## Functions 0BH to FFH—Additional functions as necessary:

The device-specific functions are described in the “Interfaces” section.

**Return Code (INIOUT):** This field contains the results of the current stage of the requested operation. For operations that are single staged or in the final stage of a discrete multistaged operation, the Return Code field indicates the results of the entire operation. The return-code values are shown in the following figure.

Return-Code Value	Definition
0000H	Operation Successfully Completed
0001H	Stage on Interrupt
0002H	Stage on Time
0005H	Not My Interrupt, Stage on Interrupt
0009H	Attention, Stage on Interrupt
0081H	Unexpected Interrupt Reset, Stage on Interrupt
8000H	Device in Use, Request Refused
8001H to 8FFFH	Service-Specific Unsuccessful Operation
9000H to 90FFH	Device Error
9100H to 91FFH	Retryable Device Error
9200H to 9FFFH	Device Error
A000H to A0FFH	Time-out Error
A100H to A1FFH	Retryable Time-Out Error
A200H to AFFFH	Time-Out Error
B000H to B0FFH	Device Error with Time-Out
B100H to B1FFH	Retryable Device Error with Time-Out
B200H to BFFFH	Device Error with Time-Out
C000H	Invalid Logical ID
C001H	Invalid Function
C002H	Reserved
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length
C005H to C01FH	Invalid Service-Specific Parameter
C020H to FFFEH	Service-Specific Unsuccessful Operation
FFFFH	Return Code Field Not Valid

Figure 4-3. Return Codes

The bits in the Return Code field are defined in the following figure.

Bit	Definition
15	Unsuccessful operation
14	Parameter error
13	Time-out error
12	Device error
11 to 9	Reserved
8	Retryable error
7	Unexpected interrupt reset
6 to 4	Reserved
3	Attention
2	Not my interrupt
1	Stage on time
0	Stage on interrupt

**Notes:**  
Bits 14 to 8 are defined as above only when bit 15 is set to 1.  
Bits 7 to 0 are defined as above only when bit 15 is set to 0.  
If all bits are set to 1, the Return Code field is not valid.

**Figure 4-4. Return Code Field Bit Definitions**

The caller of ABIOS must initialize the Return Code field to hex FFFF (Return Code Field Not Valid) before calling any ABIOS Start routine. If the operating system has an outstanding request block at interrupt time, it first checks the Return Code field. If the value of the Return Code field is hex FFFF (Return Code Field Not Valid), the operating system considers the Return Code field not set and does not attempt to resume this request. The ABIOS routine sets the Return Code field to its appropriate value when the interrupt is expected.

When ABIOS is processing a request that causes a hardware interrupt, interrupts are disabled between the time when a value is written to the Interrupt Enable port and the time when the value of the Return Code field is changed from hex FFFF (Return Code Field Not Valid) to a return-code value with the stage-on-interrupt bit (bit 0) set to 1. After the value of the Return Code field is changed, the interrupt flag is restored to the value that it contained before it was disabled.

When a hardware interrupt occurs, the caller responds only to requests that have a return-code value with the stage-on-interrupt bit (bit 0) set to 1. Outstanding requests with a return-code value of hex FFFF (Return Code Field Not Valid) are not called.

The caller should also maintain a flag that indicates whether a request has completed the Start routine to the point at which the Return Code field is read. This allows for a situation in which an

interrupt occurs after the Return Code field is set to a valid value (not hex FFFF) but before the caller reads the Return Code field. At this point, a Start routine and an Interrupt routine could be operating on the same request block, within different stack frames, making this flag necessary.

Return codes hex 0009 (Attention) and hex 0002 (Stage on Time) need to be tested only by services that require them. Return code hex 0009 indicates that data is available in a service-specific output parameter, although the function is not complete. Return code hex 0002 indicates that the operation is not complete and must be resumed when a specified length of time has elapsed. This length of time is contained in a service-specific output parameter, depending on the service. In addition, return-code values with bit 15 set to 1 are service specific. These values are documented in the "Interfaces" section.

Return code hex 8000 (Device in Use, Request Refused) is used for device serialization. If a logical ID/unit combination is a serially-reusable device, BIOS returns this return code when there is an outstanding request on the device.

**Time-Out (OUT):** The Time-Out field contains the expected duration of the requested stage. This is used to determine when an operation has timed out and needs to be reset by the Time-Out routine. The unit of time is 1 second, and the value occupies bits 15 to 3. Bits 2 to 0 of this field are reserved. A value of 0 in this field indicates that the operation has no time-out value. The Time-Out field is valid for return-code values with the stage-on-interrupt bit (bit 0) set to 1.

**Data Pointer 1, Data Pointer 2 (IN):** If data pointers are required, they are doubleword pointers to I/O-buffer areas for this request. In a bimodal environment, the effective address must be addressable in the current mode of the microprocessor. The address can be a 32-bit physical address for DMA or a segmented address for programmed I/O. The Return Logical ID Parameters function (hex 01) returns a parameter that indicates the mode (physical or logical) of the data pointer for the Read function (hex 08), the Write function (hex 09), and the Additional Data Transfer function (hex 0A). If this parameter indicates that the pointer should be a logical pointer, data pointer 1 is a logical pointer, and data pointer 2 is reserved. If this parameter indicates that the pointer should be a physical pointer, data pointer 2 is a physical pointer, and data pointer 1 is reserved. If this parameter indicates that both a logical pointer and a physical pointer are to be passed, data pointer 1 is a logical pointer, and data pointer 2 is a physical pointer. If this parameter indicates neither a logical pointer nor a physical pointer is to be passed, either this logical ID does not

support the Read, Write, and Additional Data Transfer functions, or these functions do not require address pointers. In this case, no space is reserved for data pointers in the request block.

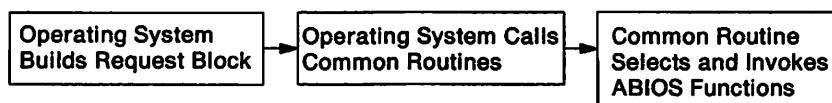
**Parameters (IN/OUT):** Parameters communicate operands and, in some cases, the results of BIOS functions. Parameter requirements vary by device and requested function. Detailed parameter requirements are documented in the "Interfaces" section.

**Work Area:** Work Area fields are optional data areas that are reserved for BIOS. No user data can be stored here. The content of these fields varies by the type of request and the particular device routine that is involved. These fields are not required to be initialized to any value. The caller must not alter their content across multistaged requests. Work Area fields are fields that are not defined as service-specific input or service-specific output parameters in the "Interfaces" section.

---

## ABIOS Transfer Convention

The ABIOS transfer convention makes ABIOS responsible for determining the effective address of a particular ABIOS function. ABIOS indexes into the common data area on the basis of the Logical ID field in the request block to access the necessary pointers, including the effective routine (Start, Interrupt, or Time-Out) pointer. The ABIOS transfer convention is the simplest calling sequence for the operating system. The flow of an ABIOS transfer request is shown below.



*Figure 4-5. Flow of ABIOS Transfer Convention*

For this transfer convention, only three routines are available to the caller for transferring control to ABIOS. The pointers to these three routines are returned in the system parameters table when ABIOS is initialized. They are also contained in the function transfer table for logical ID 2. These routines are:

### **Common Start Routine:**

This routine is called (using a Call Far Indirect) to start a request. The Logical ID field in the request block is validated. If this logical-ID value is greater than the value of the Count of Logical IDs field in the common data area, or if this logical-ID value pertains to a null common-data-area entry, the Return Code field is set to hex C000 (Invalid Logical ID).

### **Common Interrupt Routine:**

This routine is called (using a Call Far Indirect) to resume a multistaged request.

### **Common Time-Out Routine:**

This routine is called (using Call Far Indirect) to terminate a request that fails to receive a hardware interrupt within a specified length of time. The Time-Out routine terminates the request and leaves the hardware controller in a known initial state.

The parameter-passing convention for the ABIOS transfer convention is a set of two parameters, two reserved doublewords, and a return address on the stack. The first parameter is the common-data-area

anchor-pointer segment or selector with an assumed offset of hex 00. The second parameter is the doubleword pointer to the request block. The third parameter is a reserved doubleword placeholder for the function-transfer-table pointer. The fourth parameter is a reserved doubleword placeholder for the device-block pointer.

The BIOS common routines expect the order of the addresses to be from high to low (the order of pushing), as shown in the following figure.

Contents	Displacement from Stack Pointer
Return address of caller	00H
Placeholder for device-block pointer	04H
Placeholder for function-transfer-table pointer	08H
Request-block pointer	0CH
Common-data-area anchor pointer (segment or selector only)	10H

**Figure 4-6. BIOS Transfer Convention Stack Frame**

The following pseudocode instructions are suggested:

```
PUSH    Anchor-pointer segment or selector
PUSH    Request-block segment or selector
PUSH    Request-block offset
SUB     Stack pointer, 8
CALL    Common Start routine
```

### **Pseudocode—BIOS Transfer Convention**

The common routines use the logical ID from the request block and the anchor pointer to determine which device-block pointer and function-transfer-table pointer pair are to be used. These routines take this pair of pointers and place them in the stack-placeholder positions that have been allocated by the caller. Then the common routines transfer control to the Start, Interrupt, or Time-Out routine whose pointers are contained in the function transfer table for the requested value of the Logical ID field. The common-data-area segment or selector, the request-block pointer, the function-transfer-table pointer, and the device-block pointer are passed on the stack. For the BIOS transfer convention, the caller is responsible for removing the parameters from the stack on return.

The layout of the function transfer table is shown in Figure 2-4 on page 2-8.

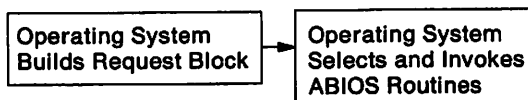
---

## Operating-System Transfer Convention

The operating-system transfer convention makes the operating system responsible for determining the effective address of a particular BIOS function. This method is most useful for handling interrupts from character and programmed-I/O devices that repeatedly call a single routine.

Two methods are available to accomplish operating-system transfers. In the first method, the operating system indexes into the common data area, on the basis of the logical ID, to access the necessary pointers, including the effective routine (Start, Interrupt, or Time-Out) pointer. The advantage of this method over the BIOS transfer convention is performance.

In the second method, the operating system stores pointers as necessary and accesses them without indexing into the common data area. An operating system might use this method if it is a real-mode-only or protected-mode-only operating system. The common data area is provided to access the necessary pointers as quickly as possible in a bimodal environment. In a single-mode environment, there is no advantage to accessing the pointers by indexing into the common data area. There is a small performance loss with this method. The flow of an operating-system-transfer-convention request is shown below.



*Figure 4-7. Flow of Operating-System Transfer Convention*

The parameter-passing convention for the operating-system transfer convention is a set of four parameters and a return address on the stack. The first parameter is the anchor-pointer segment or selector of the common data area, with an assumed offset of hex 00. The second parameter is a doubleword pointer to the request block. The third parameter is a doubleword pointer to the function transfer table. The fourth parameter is a doubleword pointer to the device block.

The Start, Interrupt, and Time-Out routines for each logical ID expect the order of the addresses to be from high to low (the order of pushing), as shown in the following figure.

<b>Contents</b>	<b>Displacement from Stack Pointer</b>
Return address of caller	00H
Device-block pointer	04H
Function-transfer-table pointer	08H
Request-block pointer	0CH
Common-data-area anchor pointer (segment or selector only)	10H

**Figure 4-8. Operating-System Transfer Convention Stack Frame**

The following pseudocode instructions are suggested:

```

PUSH   Anchor segment or selector
PUSH   Request-block segment or selector
PUSH   Request-block offset
PUSH   Function-transfer-table segment or selector
PUSH   Function-transfer-table offset
PUSH   Device-block segment or selector
PUSH   Device-block offset
CALL   Logical-ID Start routine

```

### **Pseudocode—Operating-System Transfer Convention**

For the operating-system transfer convention, the caller is responsible for removing the parameters from the stack on return.



---

## Section 5. Additional Information

Interrupt Processing .....	5-3
Interrupt Flow .....	5-3
Interrupt Sharing .....	5-3
Default Interrupt Handler .....	5-5
Adding, Patching, Extending, and Replacing .....	5-6
Adapter-ROM Structure .....	5-7
RAM-Extension Structure .....	5-9
Adding .....	5-11
Patching .....	5-12
Extending .....	5-13
Replacing .....	5-15
Considerations for RAM Extensions .....	5-16
Operating-System Implementation Considerations .....	5-18
ABIOS Rules .....	5-18
Considerations for Bimodal Implementations .....	5-20


**Notes:**



---


# Interrupt Processing

## Interrupt Flow



The operating system that communicates with ABIOS provides interrupt handlers that receive control through the hardware interrupt vector. The operating-system interrupt handler must retain the logical IDs of the devices that operate on a specified interrupt level. ABIOS provides routines that are called by the operating-system interrupt handlers.


Each device has a logical ID that is known to the operating system. A logical ID can have one or more active request blocks when an interrupt is processed by the operating-system interrupt handler. Each active request block of the logical ID is processed by calling ABIOS at its interrupt entry point. ABIOS sets the Return Code field to indicate whether the interrupt was associated with the request block.



The operating system can call ABIOS for interrupt processing with interrupts enabled or disabled. ABIOS restores the state of the interrupt flag after any period in which interrupts must be disabled. If no request blocks have the stage-on-interrupt bit (bit 0) of the Return Code field set to 1, and an interrupt occurs, the default interrupt handler is provided to remove the interrupt at the device.

## Interrupt Sharing

When more than one logical ID or logical ID/unit combination share an interrupt level, the process is repeated for each logical ID until all logical IDs are processed or the first logical ID with an interrupt is completely processed.



ABIOS expects the operating system to manage End of Interrupt (EOI) processing at the interrupt controller. The method that is used for EOI processing is determined by the operating system. ABIOS does not reset the interrupt controller. The operating system can select its strategy for resetting the interrupt controller after all outstanding request blocks for a particular logical ID are processed through the Interrupt routine and at least one request indicates that the interrupt was serviced. A serviced interrupt request returns from the Interrupt routine with any return-code value other than hex 0005 (Not My Interrupt, Stage on Interrupt).

## Rules for Interrupt Processing

**One Interrupt Level per Logical ID:** Every unit in a particular logical ID operates on the same interrupt level, and no logical ID operates on more than one interrupt level.

**One Microprocessor Mode per Call:** After being interrupted, BIOS is returned to the microprocessor mode (real or protected) in which it was running when it was interrupted. That is, after being preempted in the middle of a request stage, it will be returned to the microprocessor mode in which it was running when it was preempted.


**Microprocessor-Mode Changes Hidden from BIOS:** While BIOS function X is running in protected mode, it can be interrupted, and function Y can be invoked in real mode, and vice versa. X can equal Y. After being preempted in the middle of a request stage in one mode, BIOS can be called through the Start routine in the other mode.

**BIOS Preserves Microprocessor Interrupt Flag State:** BIOS does not change the state of the interrupt flag. BIOS might temporarily disable the interrupt flag, but it will restore it to its original state. BIOS never enables the interrupt flag if it is disabled on entry to BIOS.

**Operating System Maintains Request-Block Address Validity:** The pointer to a request block that is passed on a request is valid for the duration of that stage of the request.

**Data-Area Relocation:** The effective memory address of a logical-address pointer (a pointer in the request block in the format "segment:offset" or "selector:offset") can be changed or moved across stages of a request. In the real mode, the segment, the offset, or both can be changed. In the protected mode, the selector, the offset, or both can be changed, and the physical address in the descriptor can be changed.


**Operating System Performs EOI:** BIOS does not perform End of Interrupt (EOI) processing on its own behalf. In a level-sensitive interrupt environment, the device condition that causes the interrupt is reset by BIOS when it processes the request block at the Interrupt routine.



**Return Code Indicates Reset of Interrupt Condition:** The caller of BIOS can perform End of Interrupt (EOI) processing when BIOS returns with a successful return code during processing of the interrupt if all outstanding request blocks for that logical ID have been processed. If the Return Code field contains any value other than hex 0005 (Not My Interrupt, Stage on Interrupt) and all request blocks have been serviced on the logical ID, the caller can assume that the interrupt was serviced (including resetting of the interrupt condition at the device) and process the EOI.

**Resetting of Interrupt Condition:** Servicing an interrupt for an actual request or for the default interrupt handler resets the interrupting condition at the hardware if the Return Code field contains any value other than hex 0005 (Not My Interrupt, Stage on Interrupt).


**Exhaustive Calling:** The caller must call BIOS with each outstanding request per logical ID at interrupt time until the first logical ID with an interrupt is completely processed, which means that each request that has a return-code value with the stage-on-interrupt bit (bit 0) set to 1 for a logical ID has been called.



If multiple outstanding requests per logical ID are waiting for an interrupt, regardless of whether any single request indicates that the interrupt was serviced, each of the requests must be called. This is necessary because resetting the interrupting condition for the first request can reset the interrupt of the second request, causing an interrupt to be lost. Exceptions to this rule are specified in the "Interfaces" section. One exception is the Real-Time Clock Set Interrupt functions (hex 0B, hex 0C, and hex 0F). This cannot happen across logical IDs, because of the following rule concerning interrupts across logical IDs.

**Interrupts across Logical IDs:** Servicing an interrupt of a given logical ID does not reset the interrupt on another logical ID.

## **Default Interrupt Handler**



In a level-sensitive-interrupt environment, an unexpected hardware interrupt must be handled by resetting the interrupt at the device, as well as at the interrupt controller. BIOS provides this capability through the use of the default interrupt handler.

Each interrupting BIOS service provides a default interrupt handler that resets the interrupt at the device and sets the Return Code field to hex 0000 (Operation Successfully Completed) or hex 0005 (Not My Interrupt, Stage on Interrupt). A request block is passed to the default

interrupt handler with no service-specific parameters, and control is transferred to the default interrupt handler through the Interrupt routine. The default interrupt handler is called only if a given logical ID has no outstanding request blocks waiting on interrupt.

To determine whether a logical ID interrupts, call the Return Logical ID Parameters function (hex 01). If the Interrupt Level field pertains to a device that interrupts, it contains the hardware-interrupt level. If the Interrupt Level field contains a value of hex FF, it indicates a noninterrupting logical ID. The nonmaskable interrupt (NMI) device is a special case; it returns a value of hex FE for the interrupt level. If hex FE or hex FF is returned for the interrupt level, the logical ID does not provide a default interrupt handler.

---

## **Adding, Patching, Extending, and Replacing**

BIOS provides a mechanism for adding, patching, extending, and replacing the system-board ROM or adapter-ROM BIOS, using an adapter ROM as well as using RAM. Definitions for adding, patching, extending, and replacing BIOS are shown below, followed by the mechanisms for accomplishing each.

- Adding** This adds a previously-unsupported BIOS interface or adds the support for a new device within the constraints of the old interface, without replacing the old device. An example is adding a new hardware device with BIOS support. Adding involves a new or old interface, new BIOS, and new hardware.
- Patching** This reverts an existing BIOS function to a patched routine. Patching involves an existing interface, new BIOS, and existing hardware.
- Extending** This adds a previously-unsupported function to a particular BIOS interface that operates on the same device and uses the same device block. Extending involves a new interface, new BIOS, and existing hardware.
- Replacing** This involves supporting the existing interface and optionally extending the interface for new hardware of the same device ID. Replacing requires the initialization of a new device block. Replacing involves an existing or new interface, new BIOS, and new hardware.

The following figure shows these relationships.

	New ABIOS Interface	New ABIOS	New Hardware	New Device Block	New Function Transfer Table
Adding	Yes/No	Yes	Yes	Yes	Yes
Patching	No	Yes	No	No	No
Extending	Yes	Yes	No	No	Yes
Replacing	Yes/No	Yes	Yes	Yes	Yes

Figure 5-1. Adding, Patching, Extending, and Replacing BIOS

## Adapter-ROM Structure

ABIOS provides a facility to integrate adapters with on-board ROM code into the system. During BIOS initialization, the absolute addresses hex C0000 through hex DF800 are scanned in 2KB blocks to search for a valid adapter ROM.

Adapters that support ROMs can participate in the following convention.

Size	Offset	Description
Word	00H	Signature = hex AA55 (word value)
Byte	02H	Length, in 512-byte blocks
3 bytes	03H	BIOS initialization entry point
Word	06H	Signature = hex BB66 (word value)
Byte	08H	Number of initialization-table entries
—	09H	Build-initialization-table entry point

Figure 5-2. ROM-Module Header

The ROM-module-header entries are:

**Signature = Hex AA55 (Word Value):** This value in the ROM-module header indicates that this ROM address contains a BIOS ROM, an ABIOS ROM, or both.

**Length, in 512-Byte Blocks:** This field indicates the length (limit hex 7F) of the ROM that is associated with the ROM-module header.

**BIOS Initialization Entry Point:** This field is the ROM location that is called by the power-on self-test (POST).

**Signature = Hex BB66 (Word Value):** This value in the ROM-module header indicates that this ROM address contains an ABIOS ROM.

**Number of Initialization-Table Entries:** This field contains the number of initialization-table entries that this ABIOS ROM requires. The value of this field for each ABIOS ROM-module header must be at least 1. This field is used to determine the size of the initialization table.

**Build-Initialization-Table Entry Point:** This field is the location in the ROM of the adapter that Interrupt 15H, Build Initialization Table function ((AH)=05H, see Figure 3-4 on page 3-6) calls to build the initialization-table entry for the adapter.

The ABIOS structure is similar to the BIOS structure and does not preclude the support of existing adapters that use ROM operating under the BIOS structure. If an adapter ROM is an ABIOS-only adapter ROM, a dummy RETURN FAR instruction must be placed at the BIOS Initialization Entry Point field in the ROM-module header to allow for the BIOS ROM scan during POST.

When the operating system invokes Interrupt 15H, Build System Parameters Table function ((AH)=04H, see Figure 3-3 on page 3-5), a ROM scan is invoked to determine the number of entries in the initialization table. This number is obtained by accumulating the values in the Number of Initialization-Table Entries field of each ROM-module header and adding that number to the number of entries that are required for the system-board ROM.

When the operating system invokes Interrupt 15H, Build Initialization Table function ((AH)=05H, see Figure 3-4 on page 3-6), a ROM scan is invoked to search the ROM address space in 2KB increments until a valid ABIOS ROM is detected. The Build Initialization Table Entry function is called for each valid ROM to fill in the initialization table for devices that are operated by the code on the adapter. For more information, see Figure 5-4 on page 5-11.

After the initialization-table entry for the adapter ROM is added to the initialization table, the operating system treats the entry as if it were a system-board entry.

When the Initialize Device Block and Function Transfer Table routine is called for an adapter ROM, each segment value in the function transfer table must equal the segment value of the corresponding ROM-module header.



## RAM-Extension Structure

ABIOS provides a facility to integrate adapters with RAM-loadable code into the system. The operating system is responsible for loading the RAM extensions from permanent media to RAM before ABIOS initialization. After ABIOS initialization, RAM extensions can be relocated, but they must always be in memory. During ABIOS initialization, when Interrupt 15H, Build System Parameters Table function ((AH)=04H, see Figure 3-3 on page 3-5) and Build Initialization Table function ((AH)=05H, see Figure 3-4 on page 3-6) are called, a pointer to the RAM-extension area is passed as a parameter.

The layout of a RAM-extension header is shown below.

Size	Offset	Description
Word	00H	Signature = hex AA55 (word value)
Byte	02H	Length, in 512-byte blocks
Byte	03H	Model byte
Byte	04H	Submodel byte
Byte	05H	ROM revision level
Word	06H	Device ID
Byte	08H	Number of initialization-table entries
3 bytes	09H	Build-initialization-table entry point
Byte	0CH	Secondary device ID
Byte	0DH	Revision
Word	0EH	Reserved

Figure 5-3. RAM-Extension Header

The RAM-extension header entries are:

**Signature = Hex AA55 (Word Value):** This value in the RAM-extension header indicates that this RAM address contains an ABIOS RAM extension.

**Length, in 512-Byte Blocks:** This field indicates the length (limit hex 7F) of the RAM extension that is associated with the RAM-extension header.

**Model Byte, Submodel Byte, ROM Revision Level:** These fields describe the system-board ROM with which the RAM extension is associated.

**Device ID, Secondary Device ID, Revision:** These fields describe the ABIOS service with which the RAM extension is associated.

**Number of Initialization-Table Entries:** This field contains the number of initialization-table entries that this RAM extension requires. The value of this field for each RAM-extension header must be at least 1. This field is used to determine the size of the initialization table.

**Build-Initialization-Table Entry Point:** This field contains the location in the RAM extension that Interrupt 15H, Build Initialization Table function ((AH)=05H, see Figure 3-4 on page 3-6) calls to build the initialization-table entry for this RAM extension.

The RAM-extension area starts on a paragraph boundary and contains a chained list of individual RAM extensions that are linked by way of the Length field. The Reserved fields in the RAM-extension header must be set to 0.

The segment value of each RAM extension is calculated by converting the length of the preceding RAM extension to paragraphs and adding the result to the segment of the preceding RAM extension. If the header for RAM extension 0 is loaded at XXXX:0000 and its length is  $n$ , meaning that the extension is  $(n/2)$ KB in length, the header for RAM extension 1 is at location  $[XXXX + (20H \times n)]:0000$ . The last RAM extension in the RAM-extension area points to a RAM extension that has the Length field set to 0.

When the operating system invokes Interrupt 15H, Build System Parameters Table function ((AH)=04H, see Figure 3-3 on page 3-5), a RAM-extension scan occurs to determine the number of entries in the initialization table. This number is obtained by accumulating the values of the Number of Initialization-Table Entries field in the RAM-extension headers and adding that number to the entries that are required for the system-board and adapter ROMs.

When the operating system invokes Interrupt 15H, Build Initialization Table function ((AH)=05H, see Figure 3-4 on page 3-6), another RAM-extension scan occurs. The Build Initialization-Table Entry routine (see Figure 5-4 on page 5-11) is called for each RAM extension to fill the initialization-table entry for that RAM extension.

After the initialization-table entry for the RAM extension is added to the initialization table, the operating system treats the entry as if it were a system-board entry.

When the Initialize Device Block and Function Transfer Table routine is called, each segment value in the function transfer table must equal the segment of the corresponding RAM-extension header.

The following figure shows the interface to the Build Initialization Table routine that is used by adapter ROMs and RAM extensions.

Invocation: Call FAR; BIOS calls adapter ROM or RAM extension strictly for initialization.

(ES:DI) - Pointer to the next available entry in the initialization table

On Return:

(AL) - Exception condition

= 00H - Operation successfully completed

≠ 00H - No entries were added

= 80H - No units were found

(CX) - Number of entries that were added to the initialization table

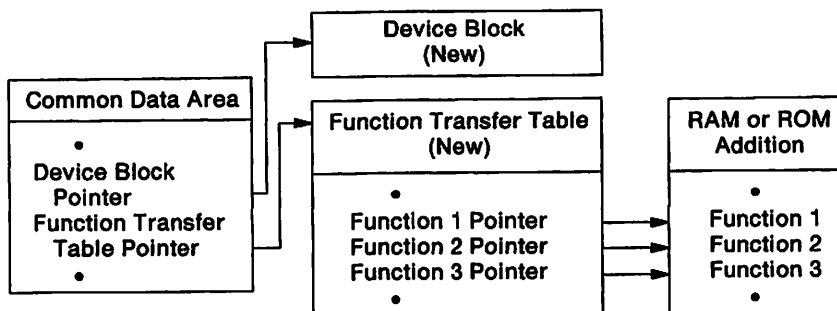
= 0 - (AL)≠0

All registers except (AX), (CX), and the flags are restored.

*Figure 5-4. Build Initialization-Table Entry Routine*

## Adding

To add a previously-unsupported BIOS interface, an adapter ROM or RAM extension provides the correct ROM-module or RAM-extension header, and the Build Initialization Table routine is used to build an entry in the initialization table. When the initialization table has been built, it makes no difference to the operating-system initialization process whether the initialization-table entry is associated with a system-board ROM, an adapter ROM, or a RAM extension. The following diagram shows the effect of adding an BIOS interface.



*Figure 5-5. Adding BIOS*

## Patching

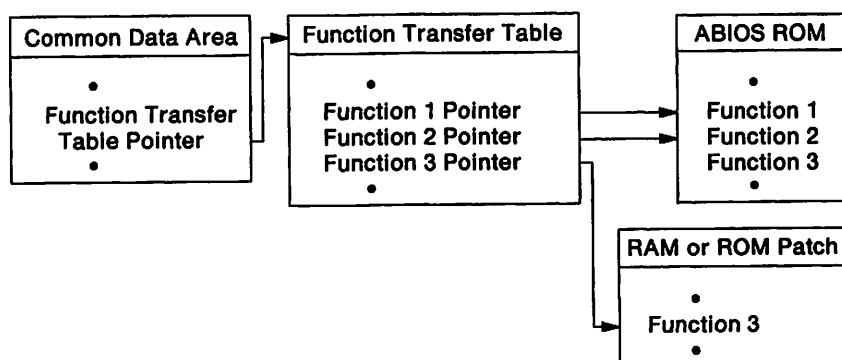
During adapter-ROM scan or RAM-extension scan, an adapter ROM or RAM extension is given control at the Build Initialization Table routine where the adapter ROM or RAM extension builds the new initialization-table entry. When an ABIOS service is patched, the new initialization-table entry that is built is the same as the old initialization-table entry, with the following exceptions:

- The Device-Block Length field is set to 0, indicating that the existing device block suffices for the adapter ROM or RAM extension. Therefore, the operating system should set the Device-Block Pointer field in the common data area that is associated with this initialization-table entry to hex 0:0.
- The Function-Transfer-Table Length field is set to 0, indicating that the existing function transfer table suffices for the adapter ROM or RAM extension. Therefore, the operating system should set the Function Transfer Table Pointer field in the common data area that is associated with this initialization-table entry to hex 0:0.
- The Number of Logical IDs field is set to 1, indicating that this entry requires one logical ID to be initialized for this initialization-table entry.
- The Revision field is set to the value of the Revision field in the old initialization-table entry plus 1.
- The Initialize Device Block and Function Transfer Table Routine Pointer field is initialized to point to the adapter ROM or RAM extension.

When control is transferred to the Initialize Device Block and Function Transfer Table routine, the common data area is scanned for the values of the Device ID field, the Secondary Device ID field, and the Revision field in the device block of the service that is to be patched. This is accomplished by reading the Device-Block Pointer field that is associated with each logical ID and examining the public portion of the device block that contains the Device ID field, the Secondary Device ID field, and the Revision field, until the logical ID (entry in the common data area) that is to be patched is found (see the ABIOS device block in Figure 2-6 on page 2-11). As the common data area is scanned, any null entries should be disregarded. The associated Function-Transfer-Table Pointer field is accessed, and the doubleword pointer of the patched routine is stored at the appropriate offset in the function transfer table.

The Device-Block Pointer field and the Function-Transfer-Table Pointer field that correspond to the Starting Logical ID parameter that is passed to the Initialize Device Block and Function Transfer Table routine are already set to hex 0:0, indicating that the operating system should disregard this entry as a null common-data-area entry.

The following diagram shows the effect of patching an ABIOS interface.



*Figure 5-6. Patching ABIOS*

## Extending

During adapter-ROM scan or RAM-extension scan, an adapter ROM or RAM extension is given control at the Build Initialization Table routine where the adapter ROM or RAM extension builds the new initialization-table entry. When an ABIOS service is extended, the new initialization-table entry that is built is the same as the old initialization-table entry, with the following exceptions:

- The Device-Block Length field is set to 0, indicating that the existing device block suffices for the adapter ROM or RAM extension. Therefore, the operating system should set the Device-Block Pointer field in the common data area that is associated with this initialization-table entry to hex 0:0.
- The Function-Transfer-Table Length field is set to the value of the old Function Transfer Table field plus the length of the extensions.
- The Number of Logical IDs field is set to 1, indicating that this entry requires one logical ID to be initialized for this initialization-table entry.

- The Revision field is set to the value of the Revision field in the old initialization-table entry plus 1.
- The Initialize Device Block and Function Transfer Table Routine Pointer field is initialized to point to the adapter ROM or RAM extension.

When control is transferred to the Initialize Device Block and Function Transfer Table routine, the common data area is scanned for the values of the Device ID field, the Secondary Device ID field, and the Revision field in the device block of the service that is to be extended. This is accomplished by reading the Device-Block Pointer field that is associated with each logical ID and examining the public portion of the device block that contains the Device ID field, the Secondary Device ID field, and the Revision field, until the logical ID (entry in the common data area) that is to be extended is found (see the BIOS device block in Figure 2-6 on page 2-11). As the common data area is scanned, any null entries should be disregarded. The old function pointers for the service that is to be extended are placed in the new function transfer table, followed by the doubleword pointers to the new functions in the adapter ROM or RAM extension. The Function Count field of the new function transfer table is updated to reflect the number of old functions plus the number of new functions. The Function-Transfer-Table Pointer field in the common data area, which previously pointed to the old function transfer table, is replaced with the pointer to the new function transfer table.

The Device-Block Pointer field that corresponds to the Starting Logical ID parameter that is passed to the Initialize Device Block and Function Transfer Table routine is already set to hex 0:0. The Initialize Device Block and Function Transfer Table routine must set the associated Function-Transfer-Table Pointer field to hex 0:0, indicating a null common-data-area entry.

The following diagram shows the effect of extending an ABIOS interface.

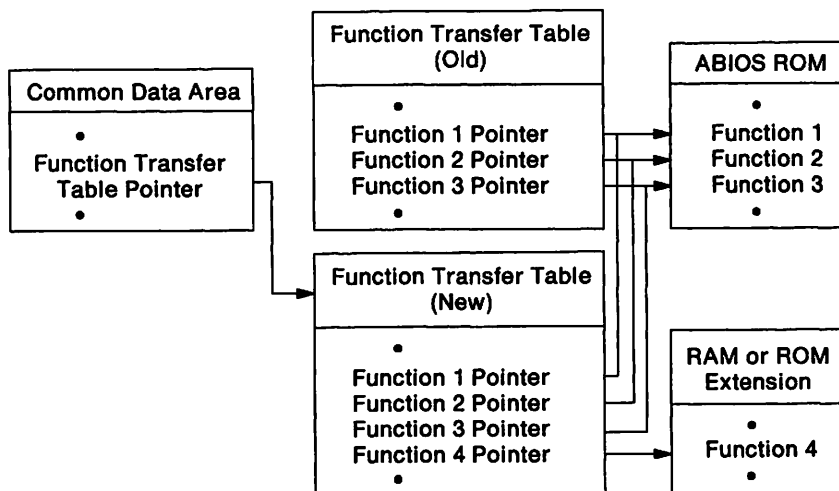


Figure 5-7. Extending ABIOS

## Replacing

To replace an ABIOS service, the adapter ROM or RAM extension is given control at the Build Initialization Table routine where the adapter ROM or RAM extension builds the new initialization-table entry. The initialization-table entry is built with a new value in each field except the Device ID field.

When the Initialize Device Block and Function Transfer Table routine is called, the common data area is scanned for the values of the Device ID field, the Secondary Device ID field, and the Revision field in the device block of the service that is to be replaced. This is accomplished by reading the Device-Block Pointer field that is associated with each logical ID and examining the public portion of the device block that contains the Device ID field, the Secondary Device ID field, and the Revision field, until the logical ID (entry in the common data area) that is to be replaced is found (see the ABIOS device block in Figure 2-6 on page 2-11). As the common data area is scanned, any null entries should be disregarded. The new function pointers that point to the adapter ROM or RAM extension are placed in the function transfer table that corresponds to the Starting Logical ID parameter. The Function-Transfer-Table Pointer field in the common data area, which previously pointed to the old function transfer table, is replaced with the pointer to the new function transfer

table. Then the device block is built for the Starting Logical ID parameter. When the device block has been built, the Device-Block Pointer field in the common data area that previously pointed to the old device block is replaced by the pointer to the new device block.

The Initialize Device Block and Function Transfer Table routine must reinitialize the Function-Transfer-Table Pointer field and the Device-Block Pointer field that correspond to the Starting Logical ID parameter to hex 0:0, indicating a null common-data-area entry.

The following diagram shows the effect of replacing an BIOS interface.

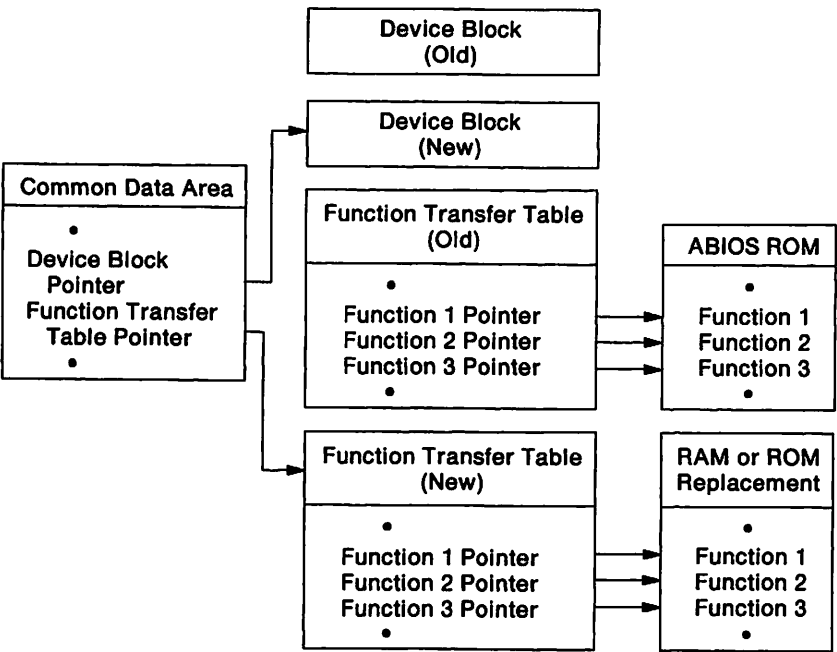


Figure 5-8. Replacing BIOS

## Considerations for RAM Extensions

The Model Byte field, the Submodel Byte field, and the ROM Revision Level field in the RAM-extension header are called system-board identifiers because they describe the system-board ROM that is associated with the RAM extension. The Device ID field, the Secondary Device ID field, and the Revision field are called service identifiers because they describe the specific BIOS service that is associated with the RAM extension.



Two tests determine whether the RAM extension is required for a particular system:

1. The first test determines whether the RAM extension should remain resident in memory. It must be performed when the RAM extension is loaded. This test determines whether the system-board identifiers in the RAM extension match the system-board identifiers that are returned by Interrupt 15H, Return System Configuration Parameters function ((AH)=C0H). If the system-board identifiers match, the RAM extension remains resident in memory for ABIOS initialization. If the RAM-extension header contains the system-board-identifier wild card (that is, the Model Byte, Submodel Byte, and ROM Revision Level fields are all set to 0), the RAM extension is loaded into memory for all systems.

To simplify the system-board identifier test, each RAM-extension file must contain RAM extensions with the same system-board identifiers. This allows the system-board-identifier test to be performed against only the first RAM-extension header while it ensures that the test is accurate for all RAM-extension headers in the file.

2. The second test determines whether the service identifiers in the RAM-extension header match a service that exists in the system-board ROM or in an adapter ROM. This test is performed by the Initialize Device Block and Function Transfer Table routine. If a matching service is not found during a scan of the common data area, the Initialize Device Block and Function Transfer Table routine sets the Exception Condition parameter to a nonzero value. When this parameter contains a nonzero value, the operating system makes the associated logical ID a null common-data-area entry.

If a single RAM extension contains patches for multiple service identifiers, the RAM-extension header must contain the service-identifier wild card (that is, the Device ID field is set to hex 00FF, and the Secondary Device ID and Revision fields are set to hex FF).

For each new version of an ABIOS service that patches, extends, or replaces an existing version, at least one service identifier in the new device block must be different from that in the old device block. The old device block is modified, or a new device block is built by the Initialize Device Block and Function Transfer Table routine, depending on the type of RAM extension (patching, extending, or replacing).

For patching and extending, a new device block is not built; therefore, the Revision field in the existing device block is updated, and the Device ID field and the Secondary Device ID field remain the same. For replacing, a new device block is built because hardware is added; therefore, the Device ID field in the new device block remains the same, but the value of the Secondary Device ID field is increased by 1, and the Revision field is set to 0. For adding, the existing device block is not tested; therefore, the new device block is built as necessary.

The IBM Operating System/2\* supports ABIOS updates (RAM extensions) as follows:

- A file called ABIOS.SYS contains a list of file specifications that are separated by blanks or new lines.
- ABIOS.SYS and the files that are associated with the file specifications in ABIOS.SYS are assumed to reside in the root directory of the IPL volume.
- If the RAM extension passes the system-identifier test, the files that are associated with the file specifications in ABIOS.SYS are loaded into memory and appended to one another in the order in which they appear in ABIOS.SYS. These files make up the ABIOS updates that are applied to ABIOS.
- The filename extension must be .BIO, and the sector size of the update files must be a multiple of 512 bytes.

---

## Operating-System Implementation Considerations

### ABIOS Rules

The following rules are presented for programmers who are writing operating systems and device drivers.

- Rule 1** The operating system must not alter the Device-Block Pointer field, the Function-Transfer-Table Pointer field, or any Data Pointer field for a given logical ID in the common data area during any stage of a request to that logical ID.

---

\* Operating System/2 is a trademark of the International Business Machines Corporation.

- Rule 2** After BIOS is interrupted during any stage of a request, it returns to that stage in the mode in which it was running at the time of the interrupt.
- Rule 3** BIOS device blocks are owned by BIOS, and only the public portions are accessible by the operating system. There is no guarantee of compatibility of device-block private-area contents across BIOS implementations.
- Rule 4** BIOS and the operating system share BIOS request blocks.
- Rule 5** BIOS must traverse the common data area to retrieve the necessary pointers. It does not store pointers in one request or one stage of a request to be used for another request or stage of a request.
- Rule 6** BIOS function X can be interrupted while it is running in protected mode, and function Y can be invoked in real mode, and vice versa. X can equal Y. After BIOS is preempted in the middle of a request stage in one mode, it can be called through the START routine in the other mode.
- Rule 7** BIOS does not change the state of the interrupt flag. BIOS can temporarily disable the interrupt flag, but it restores the flag to its original state.
- Rule 8** A request-block pointer that is passed on a request is valid for the duration of that stage of the request.
- Rule 9** The effective memory address of a physical-address pointer must not be moved for the duration of a single request. When a function requires the data pointer to be passed as a physical address in memory, an external process is assumed to be performing the read or write to memory; therefore, this address cannot change across stages.
- Rule 10** The effective memory address of a logical-address pointer (a pointer in the request block in the format "segment:offset" or "selector:offset") can be changed or moved across stages of a request. In real mode, the segment or offset can be changed. In protected mode, the selector or offset can be changed, and the physical address in the descriptor can also be changed.
- Rule 11** BIOS does not perform End of Interrupt processing. In a level-sensitive-interrupt environment, BIOS resets the device condition that caused the interrupt.

- Rule 12** The caller of BIOS can perform End of Interrupt processing when the Return Code field is set to any value other than hex 0005 (Not My Interrupt, Stage on Interrupt) and all request blocks on the logical ID are serviced. The caller can assume that the interrupt was serviced and process the End of Interrupt.
- Rule 13** The caller of BIOS must call each outstanding request for each logical ID at interrupt time until the first logical ID with an interrupting condition is completely processed. Exceptions are defined in the "Interfaces" section.
- Rule 14** The operating system allocates operating-system device numbers on the basis of increasing units within increasing logical IDs. For example, if the first logical ID has a printer device ID, unit 0 is LPT1:, and unit 1 is LPT2:. If unit 1 does not exist and the second logical ID has a printer device ID, unit 0 is LPT2:, and so on.
- Rule 15** In a protected-mode or bimodal implementation, BIOS must have I/O privilege when it is operating in protected mode.

## Considerations for Bimodal Implementations

BIOS is written to be independent of the mode of the microprocessor. Segmented address pointers have different meanings in the two modes, and memory above 1MB is generally not addressable in real mode; therefore, an operating system with a bimodal implementation must conform to the following requirements:

**Addressability of Tables:** The operating system must ensure that the request blocks, device blocks, function transfer tables, and common data area are always addressable by BIOS, in the mode in which it is called.

**Addressability of Data for Programmed I/O:** Non-DMA devices cannot readily use memory above 1MB in real mode. The operating system should allocate the I/O buffers for these devices below 1MB if BIOS is called in real mode.

**Mode Change and Reentrant Routines:** When BIOS is operating in one mode, it can be interrupted and invoked in the other mode. The Interrupt routine and the Time-Out routine are fully reentrant. The Start routines are reentrant with respect to the device blocks. That is, BIOS can support multiple requests to common code that is operating on different device blocks at the same time within different

stack frames. If the Start routine cannot begin a request, it sets the Return Code field to hex 8000 (Device in Use, Request Refused).

**Two Copies of Tables Recommended:** Because segmented memory pointers have ambiguous meaning in a bimodal environment, the operating system should keep a real-mode version and a protected-mode version of the common data area and function transfer tables. This avoids the overhead of converting all of the pointers in the tables after switching modes. The offset fields in the function transfer table must be the same for corresponding entries in both versions of the table. When there are two copies of the table, a value of hex 0:0 in the Data Pointer field in the real-mode common data area indicates that the address is above 1MB.

ABIOS is not affected by the existence of more than one table. The protected-mode table does not need to be initialized before ABIOS is called in real mode. However, the protected-mode table must be built before ABIOS is called in protected mode.

The following figure illustrates the BIOS common data area, function transfer tables, and device blocks in a bimodal environment.

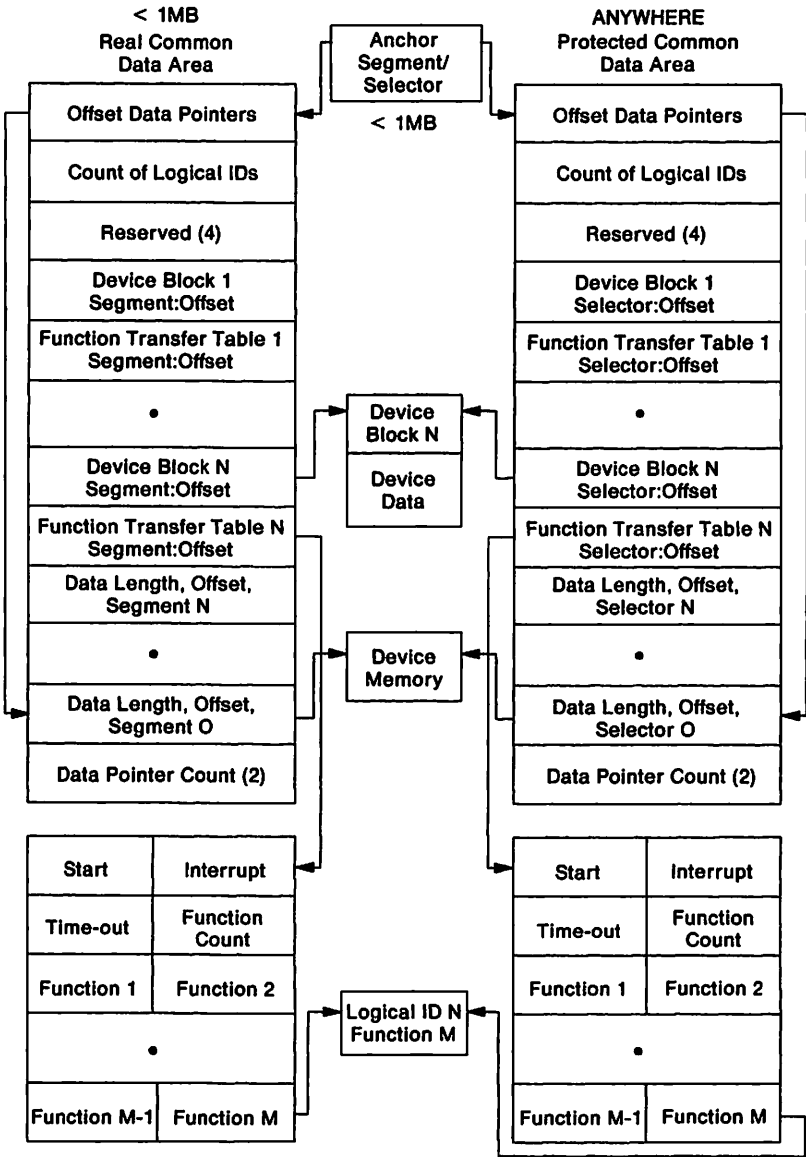


Figure 5-9. Bimodal Data Areas

The fields of the bimodal common data area (Figure 5-9) are:

**Anchor Segment/Selector:** This is called the anchor pointer. It is a word segment or selector with an assumed offset of hex 0, and it points to the common data area. The segment value that is passed in real mode does not need to equal the selector value that is passed in protected mode.

**Real Common Data Area:** This is the common data area that is used by BIOS in the real mode.

**Protected Common Data Area:** This is the common data area that is used by BIOS in the protected mode.

**Offset Data Pointers:** This is an offset that, in conjunction with the anchor pointer, points to the BIOS Data Pointer 0 Length field.

**Count of Logical IDs:** This is the number of device-block and function-transfer-table pointer pairs.

**Reserved (4):** This is a reserved doubleword.

**Device Block N Segment:Offset:** This is the doubleword pointer to the device block for logical ID *n*.

**Function Transfer Table N Segment:Offset:** This is the doubleword pointer to the function transfer table for logical ID *n*.

**Device Block N Selector:Offset:** This is the doubleword pointer to the device block for logical ID *n*.

**Function Transfer Table N Selector:Offset:** This is the doubleword pointer to the function transfer table for logical ID *n*.

**Data Pointer Count (2):** This is the number of Data Pointer fields.

**Data Length, Offset, Segment N:** This is the length, offset, and segment of data pointer *n*.

**Data Length, Offset, Selector N:** This is the length, offset, and selector of data pointer *n*.

**Device Block N:** This is the BIOS device block *n*.

**Start:** This is a doubleword pointer to the Start routine for this logical ID. It is available to the caller through the operating-system transfer convention.

**Interrupt:** This is a doubleword pointer to the Interrupt routine for this logical ID. It is available to the caller through the operating-system transfer convention.

**Time-Out:** This is a doubleword pointer to the Time-Out routine for this logical ID. It is available to the caller through the operating-system transfer convention.

**Function Count:** This is the number of functions that are supported for this logical ID.

**Function M:** This is a doubleword pointer to the *m*th Function routine for this logical ID.



## Section 6. Interfaces

This section describes the interfaces that are supported by BIOS. Each interface description includes the interface functions and return-code values. Programming considerations are also included where appropriate.

Parameters are passed to BIOS functions through request blocks. Input parameters are set by the caller, and output parameters are returned by the BIOS functions.

This section describes only the service-specific parameters. Functional parameters are described in "Request Block" in the "Transfer Conventions" section.

The following notes apply to each BIOS device interface in this section:

- The Default Interrupt Handler function (hex 00) and the Logical ID Parameters function (hex 01) are described in "Request Block" in the "Transfer Conventions" section.
- For the Read (hex 08), Write (hex 09), and Additional Data Transfer (hex 0A) functions, the data-pointer mode (physical or logical) should be determined through the Return Logical ID Parameters function (hex 01).
- All reserved input fields must be set to 0.
- All input fields are unaltered by BIOS across the stages of a request.
- All fields, other than input fields, do not need to be initialized to any predefined values before a request is initiated through the Start routine. The caller must not alter these fields across the stages of a request.
- The following return-code values are returned for parameter errors, although they are not indicated as possible return-code values in each function description:
  - Hex C000 – Invalid Logical ID (BIOS transfer convention only)
  - Hex C001 – Invalid Function Number
  - Hex C003 – Invalid Unit Number
  - Hex C004 – Invalid Request-Block Length.

- The return-code value hex 8000 (Device in Use, Request Refused) is used for device serialization. If a logical ID/unit combination is a serially-reusable device, ABIOS returns this value when there is an outstanding request on the device.
- The caller should generically handle the error ranges of the Return Code field as defined for the request block (see the "Transfer Conventions" section). This permits the definition of additional return codes in each of the ranges without affecting the caller's error handling.
- The Time to Wait before Resuming Request field is returned when the Return Code field is set to hex 0002 (Stage on Time).

## Device ID 01H—Diskette

### Functions

The following are the diskette functions. The Default Interrupt Handler function (hex 00) and the Return Logical ID Parameters function (hex 01) are described in "Request Block" in the "Transfer Conventions" section.

**Note:** All reserved input fields must be set to 0.

#### 00H—Default Interrupt Handler

#### 01H—Return Logical ID Parameters

#### 02H—Reserved

#### 03H—Read Device Parameters

- This function returns device-control information and the default parameters that are used in diskette operations.
- This function returns bit 6 of the Device Control Flags field to indicate whether the Gap Length for Format field is a required input for the Set Media Type for Format function (hex 0D). If bit 6 is set to 1, the Gap Length for Format parameter is determined on the basis of the Number of Tracks to Be Formatted field and the Number of Sectors per Track field that are passed in the Set Media Type for Format function (hex 0D). If bit 6 is set to 0, the user must provide the Gap Length for Format parameter for the media that is being formatted.
- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Word	18H	Reserved

### Service-Specific Output

Size	Offset	Description
Word	10H	Number of sectors per track for the maximum media density that is supported by the drive
Word	12H	Size of sector, in bytes
		= 00H - Reserved
		= 01H - Reserved
		= 02H - 512 bytes per sector
		= 03H to FFFFH - Reserved

Size	Offset	Description
Word	14H	Device control flags Bits 15 to 7 - Reserved Bit 6 - Support of Gap Length for Format parameter for the Set Media Type for Format function (hex 0D) = 0 - User must provide Gap Length for Format parameter = 1 - BIOS defines the Gap Length for Format parameter on the basis of the Number of Tracks to Be Formatted field and the Number of Sectors per Track field in the Set Media Type for Format function (hex 0D) Bits 5, 4 - Reserved Bit 3 - Recalibration status = 0 - Recalibration is not required = 1 - Recalibration is required Bit 2 - Concurrent operations support = 0 - Not supported = 1 - Supported Bit 1 - Format-unit support = 0 - Not supported = 1 - Supported Bit 0 - Change-signal availability = 0 - Not available = 1 - Available
Word	16H	Diskette drive type = 00H - Drive not present/invalid NVRAM = 01H - 5.25-inch, 40-track, 2-head, 360KB = 02H - 5.25-inch, 80-track, 2-head, 1.2MB = 03H - 3.5-inch, 80-track, 2-head, 720KB = 04H - 3.5 inch, 80-track, 2-head, 1.44MB = 05H - Reserved = 06H - 3.5-inch, 80-track, 2-head, 2.88MB 07H to FFFFH - Reserved
DWord	1CH	Delay before turning off motor (in microseconds)
DWord	20H	Motor-startup time (in microseconds)
Word	26H	Number of cylinders in the maximum media density that is supported by the drive
Byte	2AH	Number of heads
Byte	2BH	Recommended software retry count
Byte	2CH	Fill byte for format
Byte	2DH	Head settle time (in microseconds)
Byte	31H	Gap length for read/write/verify
Byte	32H	Gap length for format
Byte	33H	Data length

## 04H—Set Device Parameters

- This function can be used to change the default parameters for diskette operations.
- The possible values of the Return Code field are hex 0000, 8000, and C005.

## Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
Word	12H	Sector size, in bytes 00H - Reserved = 01H - Reserved = 02H - 512 bytes per sector = 03H to FFFFH - Reserved
Byte	31H	Gap length for read/write/verify
Byte	33H	Data length

## Service-Specific Output

Size	Offset	Description
None		

### 05H—Reset/Initialize

- This function resets the diskette system to an initial state.
- This function should be issued when switching from BIOS to ABIOS.
- If an error occurs, ABIOS will indicate that a controller reset is required upon entry to the next request.
- The caller is responsible for turning off the motor when the request is completed.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 8000, 9009, 9120, and 9180.

## Service-Specific Input

Size	Offset	Description
Word	10H	Reserved

## Service-Specific Output

Size	Offset	Description
None		

### 06H—Enable (Reserved)

## 07H—Disable/Reset Interrupt

- This function resets the interrupt at the device.
- The possible values of the Return Code field are hex 0000, 9120, and 9180.

## Service-Specific Input

Size	Offset	Description
Word	18H	Reserved

## Service-Specific Output

Size	Offset	Description
None		

## 08H—Read

- This function transfers data from the specified cylinder, head, and sector on the diskette to the specified memory location. The Return Logical ID Parameters function (hex 01) returns the data-pointer mode (whether it is physical or logical).
- If the 'diskette change' signal is inactive, BIOS proceeds with the operation.
- If the 'diskette change' signal is active and BIOS is able to reset the 'diskette change' signal to the inactive state, the Return Code field is set to hex 8006 (Media Changed). However, if the 'diskette change' signal is active and BIOS is not able to reset the 'diskette change' signal to the inactive state, the Return Code field is set to hex 800D (Media Not Present), and no data is transferred.
- If the Number of Sectors to Be Read field is set to 0, no action is performed, and the Return Code field is set to hex 0000 (Operation Successfully Completed).
- BIOS supports only a block size of 512 bytes per sector.
- The caller is responsible for turning off the motor when the request is completed.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 8000, 8006, 800D, 800E, 9009, 9102, 9103, 9104, 9110, 9120, 9140, 9180, and C00C.

## Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Data pointer 1
Word	16H	Reserved
Word	18H	Reserved
DWord	1AH	Data pointer 2
Word	1EH	Reserved
Word	24H	Number of sectors to be read
Word	26H	Cylinder number (0 based)
Byte	2AH	Head number (0 based)
Word	31H	Sector number

## Service-Specific Output

Size	Offset	Description
DWord	20H	Time to wait before resuming request, in microseconds
Word	24H	Number of sectors that were read

### 09H—Write

- This function transfers data from the specified memory location to the specified cylinder, head, and sector on the diskette.
- If the 'diskette change' signal is inactive, ABIOs proceeds with the operation.
- If the 'diskette change' signal is active and ABIOs is able to reset the 'diskette change' signal to the inactive state, the Return Code field is set to hex 8006 (Media Changed). However, if the 'diskette change' signal is active and ABIOs is not able to reset the 'diskette change' signal to the inactive state, the Return Code field is set to hex 800D (Media Not Present), and no data is transferred.
- If the Number of Sectors to Be Written field is set to 0, no action is performed, and the Return Code field is set to hex 0000 (Operation Successfully Completed).
- ABIOs supports only a block size of 512 bytes per sector.
- The caller is responsible for turning off the motor when the request is completed.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 8000, 8003, 8006, 800D, 800E, 9009, 9102, 9104, 9108, 9110, 9120, 9140, 9180, and C00C.

## Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Data pointer 1
Word	16H	Reserved
Word	18H	Reserved
DWord	1AH	Data pointer 2
Word	1EH	Reserved
Word	24H	Number of sectors to be written
Word	26H	Cylinder number (0 based)
Byte	2AH	Head number (0 based)
Word	31H	Sector number

## Service-Specific Output

Size	Offset	Description
DWord	20H	Time to wait before resuming request, in microseconds
Word	24H	Number of sectors that were written

### 0AH—Additional Data Transfer (Subfunction 00H—Format)

- This function writes the field ID from the given buffer for each sector to the specified track.
  - Each field ID entry in the buffer is composed of 4 bytes in this order: C, H, R, N, where C is the track number, H is the head number, R is the sector number, and N is the sector size. There must be one entry for every sector on the track.
  - Before this function is issued, the Set Media Type for Format function (hex 0D) must be issued once to ensure the proper format parameters.
  - The Set Media Type for Format function (hex 0D) must also be issued if the Return Code field is set to hex 8006 (Media Changed) or hex 800D (Media Not Present).
  - If the 'diskette change' signal is inactive, ABIOS proceeds with the operation.
  - If the 'diskette change' signal is active and ABIOS is able to reset the 'diskette change' signal to the inactive state, the Return Code field is set to hex 8006 (Media Changed). However, if the 'diskette change' signal is active and ABIOS is not able to reset the 'diskette change' signal to the inactive state, the Return Code field is set to hex 800D (Media Not Present), and the field ID is not written.
- | • ABIOS supports only a block size of 512 bytes per sector.
- | • The caller is responsible for turning off the motor when the request is completed.



- The possible values of the Return Code field are hex 0000, 0001, 0002, 8000, 8003, 8006, 800D, 800E, 9009, 9102, 9104, 9108, 9110, 9120, 9140, 9180, and C00C.

### Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Data pointer 1
Word	16H	Reserved
Word	18H	Reserved
DWord	1AH	Data pointer 2
Word	1EH	Reserved
Word	24H	Subfunction number
Word	26H	Cylinder number (0 based)
Byte	2AH	Head number (0 based)

### Service-Specific Output

Size	Offset	Description
DWord	20H	Time to wait before resuming request, in microseconds

### 0BH—Verify Sectors

- This function verifies the data on the diskette. The operation is similar to the Read function (hex 08), except that data is not transferred.
- If the 'diskette change' signal is inactive, ABIOs proceeds with the operation.
- If the 'diskette change' signal is active and ABIOs is able to reset the 'diskette change' signal to the inactive state, the Return Code field is set to hex 8006 (Media Changed). However, if the 'diskette change' signal is active and ABIOs is not able to reset the 'diskette change' signal to the inactive state, the Return Code field is set to hex 800D (Media Not Present).
- If the Number of Sectors to Be Verified field is set to 0, no action is performed, and the Return Code field is set to hex 0000 (Operation Successfully Completed).
- ABIOs supports only a sector size of 512 bytes per sector.
- The caller is responsible for turning off the motor when the request is completed.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 8000, 8006, 800D, 800E, 9009, 9102, 9104, 9108, 9110, 9120, 9140, 9180, and C00C.

## Service-Specific Input

Size	Offset	Description
Word	16H	Reserved
Word	1EH	Reserved
Word	24H	Number of sectors to be verified
Word	26H	Cylinder number (0 based)
Byte	2AH	Head number (0 based)
Word	31H	Sector number

## Service-Specific Output

Size	Offset	Description
DWord	20H	Time to wait before resuming request, in microseconds
Word	24H	Number of sectors that were verified

## 0CH—Read Media Parameters

- This function returns the media parameters that were used for the previous operation.
- Because multiple media types might be supported for a single drive type, the Read function (hex 08), Write function (hex 09), Verify Sectors function (hex 0B), or Format function (hex 0A) should be issued to ensure that the proper media parameter values are returned before the Read Media Parameters function (hex 0C) is issued.
- The caller is responsible for turning off the motor when the request is completed.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 8000, 8006, 800D, 9009, 9102, 9104, 9108, 9110, 9120, 9140, 9180, and C00C.

## Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
Word	10H	Number of sectors per track
Word	12H	Size of sector, in bytes = 00H - Reserved = 01H - Reserved = 02H - 512 bytes per sector = 03H to FFFFH - Reserved
Word	26H	Number of cylinders
Byte	2AH	Number of heads
Byte	31H	Gap length for read/write/verify
Byte	32H	Gap length for format
Byte	33H	Data length

### 0DH—Set Media Type for Format

- This function sets the media information for the format operation on the basis of the number of tracks to be formatted (in offset hex 26) and the number of sectors per track (in offset hex 10).
- The presence of media is checked.
  - If the diskette has been removed or the drive door is left open, ABIOs sets the Return Code field to hex 800D (Media Not Present), and the media parameters are not set.
  - If the diskette has been changed and a diskette is present in the drive, ABIOs sets the requested media parameters and resets the 'diskette change' signal to the inactive state.
- If the number of tracks to be formatted (offset hex 26) and the number of sectors per track (offset hex 10) are valid for the supported diskette drive types, ABIOs sets the correct parameters as requested. Otherwise, the Return Code field is set to hex C00C (Unsupported Media Type/Unestablished Media).
- The Read Device Parameters function (hex 03) returns bit 6 of the Device Control Flags field to indicate whether the Gap Length for Format field is a required input for the Set Media Type for Format function (hex 0D). If bit 6 is set to 1, the Gap Length for Format parameter is determined on the basis of the Number of Tracks to Be Formatted field and the Number of Sectors per Track field that are passed in the Set Media Type for Format function (hex 0D). If bit 6 is set to 0, the user must provide the Gap Length for Format parameter for the media that is being formatted.
- This function must be issued once to ensure the proper diskette format information before the Format function (hex 0A) is issued.
- ABIOs uses these parameters until they are changed by the Set Device Parameters function (hex 04) or until the drive door is opened.

- The caller is responsible for turning off the motor when the request is completed.
- The possible values of the Return Code field are hex 0000, 8000, 800D, 800F, C005, and C00C.

### Service-Specific Input

Size	Offset	Description
Word	10H	Number of sectors per track
Word	12H	Size of sector, in bytes = 00H - Reserved = 01H - Reserved = 02H - 512 bytes per sector = 03H to FFFFH - Reserved
Word	16H	Reserved
Byte	26H	Number of tracks to be formatted
Byte	2CH	Fill byte for format
Byte	32H	Gap length for format

### Service-Specific Output

Size	Offset	Description
DWord	20H	Time to wait before resuming request, in microseconds

### 0EH—Read 'Diskette Change' Signal Status

- This function returns the state of the 'diskette change' signal. It does not change the state of the 'diskette change' signal.
- The 'Diskette Change' Signal Status field is valid only when the specified drive supports the 'diskette change' signal. The Read Device Parameters function (hex 03) returns information about 'diskette change' signal availability.
- An active 'diskette change' signal indicates that one or more of the following conditions exist:
  - The diskette has been changed.
  - The diskette drive door is open.
  - The diskette-type information is invalid.

Data is not transferred when the 'diskette change' signal is active.

- The Read function (hex 08), Write function (hex 09), Verify Sectors function (hex 0B), and Format Function (hex 0A) reset the 'diskette change' signal to the inactive state before they begin execution.
- The caller is responsible for turning off the motor when the request is completed.

- The possible values of the Return Code field are hex 0000, 8000, and 800E.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

### Service-Specific Output

Size	Offset	Description
Byte	10H	'Diskette change' signal status = 00H - 'Diskette change' signal is inactive = 01H to 05H - Reserved = 06H - 'Diskette change' signal is active = 07H to FFH - Reserved

### 0FH—Turn Off Motor

- This function turns the diskette-drive motor off for the requested drive.
- The caller can turn the motor off when the Return Code field is set to hex 0000 (Operation Successfully Completed).
- This function is required for the Reset/Initialize function (hex 05), Read function (hex 08), Write function (hex 09), Additional Data Transfer function (hex 0A), Verify Sectors function (hex 0B), Read Media Parameters function (hex 0C), Set Media Type for Format function (hex 0D), and Read Change Signal Status function (hex 0E).
- The Read Device Parameters function (hex 03) returns the length of the delay before the motor is turned off.
- The possible values of the Return Code field are hex 0000 and 8000.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

### Service-Specific Output

Size	Offset	Description
None		

### 10H—Interrupt Status

- This function returns the diskette interrupt-pending status. It does not reset the interrupt condition.
- The possible values of the Return Code field are hex 0000 and 8000.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

### Service-Specific Output

Size	Offset	Description
Byte	10H	Interrupt-pending status = 00H - No interrupt is pending = 01H - Interrupt pending

### 11H—Get Media Type

- This function determines the media type that is present in a specified drive.
- If media sense is not supported, the Return Code field is set to hex 8011 (Drive Does Not Support Media Sense). In this case, the media type is considered to be undefined, and control is returned to the caller.
- If no media is found in the selected drive, the Return Code field is set to hex 800D (Media Not Present). In this case, the media type is considered to be undefined, and control is returned to the caller.
- If the drive type of the selected drive does not support the media type that is found in that drive, the Return Code field is set to hex 8010 (Media Type Not Supported by Drive). In this case, the media type that is returned corresponds to the media type that was found.
- If no errors have occurred, the Return Code field is set to hex 0000 (Operation Successfully Completed). In this case, the media type that is returned corresponds to the media type that was found.
- The possible values of the Return Code field are hex 0000, 800D, 8010, and 8011.

## Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
Word	10H	Media type found = 00H - Reserved = 03H - Diskette, 1MB (unformatted) = 04H - Diskette, 2MB (unformatted) = 06H - Diskette, 4MB (unformatted) All other values are reserved.

## Return Codes

Return codes are returned at offset hex 0C.

Value	Description
0000H	Operation Successfully Completed
0001H	Stage on Interrupt
0002H	Stage on Time
0005H	Not My Interrupt, Stage on Interrupt
8000H	Device Busy, Operation Refused
8003H	Write Attempted on a Write-Protected Diskette
8006H	Media Changed
800DH	Media Not Present
800EH	Change Signal Not Available
800FH	Invalid Value in NVRAM
8010H	Media Type Not Supported by Drive
8011H	Drive Does Not Support Media Sense
9009H	Controller Failure in Reset Operation
9102H	Address Mark Not Found
9104H	Requested Sector Not Found
9108H	DMA Overrun on Operation
9110H	Bad CRC on Diskette Read
9120H	Controller Failure
9140H	Seek Operation Failure
9180H	General Error
A120H	Controller Failure
B020H	Controller Failure
C000H	Invalid Logical ID (ABIOS transfer convention only)
C001H	Invalid Function
C003H	Invalid Unit Number
C004H	Invalid Request Block Length
C005H	Invalid Diskette Parameter
C00CH	Unsupported Media Type/Unestablished Media

Figure 6-1. Diskette Return Codes

## Programming Considerations

- Diskette ABIOs indicates in the Return Code field whether an unsuccessful operation needs to be retried. The Read Device Parameters function (hex 03) returns the recommended retry count.
- When the Return Code field is set to hex 0000 (Operation Successfully Completed) the caller can turn off the motor by using the Turn Off Motor function (hex 0F). This function is required for the Reset/Initialize function (hex 05), Read function (hex 08), Write function (hex 09), Additional Data Transfer function (hex 0A), Verify Sectors function (hex 0B), Read Media Parameters function (hex 0C), Set Media Type for Format function (hex 0D), and Read Change Signal Status function (hex 0E). When the request is completed, the caller is responsible for turning off the motor by using the Turn Off Motor function (hex 0F).
- Diskette ABIOs supports crossing of track boundaries, but only switching from head 0 to head 1 on the same cylinder. It does not support switching from head 1 of one cylinder to head 0 of the next cylinder.
- When diskette ABIOs and diskette BIOS requests are issued, the following rules must be followed:
  - Do not attempt an ABIOs call while there is an outstanding BIOS call.
  - Do not attempt a BIOS call while there is an outstanding ABIOs call.
  - The Reset/Initialize function (hex 05) must be the first ABIOs request that follows a BIOS request.
  - The Reset Diskette System BIOS function (Interrupt 13H, (AH)=00H) must be the first BIOS request that follows an ABIOs request. Also, before any diskette function is issued, bit 4 must be set to 0 in BIOS data area hex 40:90 for drive A and in BIOS data area hex 40:91 for drive B.
  - The Reset/Initialize function (hex 05) must be issued after ABIOs initialization has been completed.
- If an error occurs, ABIOs resets the diskette system.
- The Read function (hex 08), Write function (hex 09), Verify Sectors function (hex 0B), Additional Data Transfer function (hex 0A), and Set Media Type for Format function (hex 0D) reset the 'diskette change' signal to the inactive state before they begin execution.



## **Diskette Drive Parameters**

The following tables list the recommended parameters for diskette drives that are supported on Personal System/2 products.

<b>Byte Definition</b>	<b>320K Media</b>	<b>360K Media</b>
First specification byte	D0H	D0H
Second specification byte	02H	02H
Motor-off time	25H	25H
Bytes per sector	02H	02H
Sectors per track	08H	09H
Gap length	2AH	2AH
Data length	FFH	FFH
Gap length (format)	50H	50H
Fill byte (format)	F6H	F6H
Head settle time (in microseconds)	0FH	0FH
Motor start (in 1/4-seconds)	06H	06H
Maximum track numbers	27H	27H
Data-transfer rate	80H	80H
Multi-rate capability	00H	00H

**Figure 6-2. Media Parameter Table – 360KB Slimline Drive**

<b>Byte Definition</b>	<b>720K Media</b>
First specification byte	D0H
Second specification byte	02H
Motor-off time	25H
Bytes per sector	02H
Sectors per track	09H
Gap length	2AH
Data length	FFH
Gap length (format)	50H
Fill byte (format)	F6H
Head settle time (in microseconds)	0FH
Motor start (in 1/4-seconds)	04H
Maximum track numbers	4FH
Data-transfer rate	80H
Multi-rate capability	00H

**Figure 6-3. Media Parameter Table – 720KB Slimline Drive**

Byte Definition	320K Media	360K Media	1.2M Media
First specification byte	E0H	E0H	D0H
Second specification byte	02H	02H	02H
Motor-off time	25H	25H	25H
Bytes per sector	02H	02H	02H
Sectors per track	08H	09H	0FH
Gap length	2AH	2AH	1BH
Data length	FFH	FFH	FFH
Gap length (format)	50H	50H	54H
Fill byte (format)	F6H	F6H	F6H
Head settle time (in microseconds)	0FH	0FH	0FH
Motor start (in 1/6-seconds)	04H	04H	04H
Maximum track numbers	27H	27H	4FH
Data-transfer rate	40H	40H	00H
Multi-rate capability	02H	02H	02H

**Figure 6-4. Media Parameter Table — 1.2MB Slimline Drive**

Byte Definition	720K Media	1.44M Media
First specification byte	E0H	D0H
Second specification byte	02H	02H
Motor-off time	25H	25H
Bytes per sector	02H	02H
Sectors per track	09H	12H
Gap length	2AH	1BH
Data length	FFH	FFH
Gap length (format)	50H	65H
Fill byte (format)	F6H	F6H
Head settle time (in microseconds)	0FH	0FH
Motor start (in 1/6-seconds)	04H	04H
Maximum track numbers	4FH	4FH
Data-transfer rate	80H	00H
Multi-rate capability	02H	02H

**Figure 6-5. Media Parameter Table — 1.44MB Slimline Drive**

<b>Byte Definition</b>	<b>720K Media</b>	<b>1.44M Media</b>	<b>2.88M Media</b>
First specification byte	E0H	D0H	A0H
Second specification byte	02H	02H	02H
Motor-off time	25H	25H	25H
Bytes per sector	02H	02H	02H
Sectors per track	09H	12H	24H
Gap length	2AH	1BH	38H
Data length	FFH	FFH	FFH
Gap length (format)	50H	65H	53H
Fill byte (format)	F6H	F6H	F6H
Head settle time (in microseconds)	0FH	0FH	0FH
Motor start (in 1/6-seconds)	04H	04H	04H
Maximum track numbers	4FH	4FH	4FH
Data-transfer rate	80H	00H	C0H
Multi-rate capability	02H	02H	02H

**Figure 6-6. Media Parameter Table – 2.88MB Slimline Drive**

<b>Byte Definition</b>	<b>720K Media</b>	<b>1.44M Media</b>
First specification byte	D0H	A0H
Second specification byte	02H	02H
Motor-off time	25H	25H
Bytes per sector	02H	02H
Sectors per track	09H	12H
Gap length	2AH	1BH
Data length	FFH	FFH
Gap length (format)	50H	65H
Fill byte (format)	F6H	F6H
Head settle time (in microseconds)	0FH	0FH
Motor start (in 1/6-seconds)	04H	04H
Maximum track numbers	4FH	4FH
Data-transfer rate	80H	00H
Multi-rate capability	02H	02H

**Figure 6-7. Media Parameter Table – 1.44MB Half-High Drive**

## Notes:

## Device ID 02H—Fixed Disk

### Functions

The following are the fixed disk functions. The Default Interrupt Handler function (hex 00) and the Return Logical ID Parameters function (hex 01) are described in "Request Block" in the "Transfer Conventions" section.

**Note:** All reserved input fields must be set to 0.

### 00H—Default Interrupt Handler

### 01H—Return Logical ID Parameters

### 02H—Reserved

### 03H—Read Device Parameters

- This function returns disk-drive information for devices that are specified in the Unit field.
- The possible values of the Return Code field are hex 0000 and 8003.

### Service-Specific Input

Size	Offset	Description
Word	28H	Reserved

### Service-Specific Output

Size	Offset	Description
Word	10H	Sectors per track associated with unit in request block
Word	12H	Size of sectors in bytes
		= 00H to 01H - Reserved
		= 02H - 512-byte sectors
		= 03H to FFFFH - Reserved

Size	Offset	Description
Word	14H	Device control flags (see "Device Control Flags" on page 6-ID02-18)
		Bit 15 - SCB transfer support = 1 - SCB transfer function is supported
		Bit 14 - SCSI Device = 1 - Drive is a SCSI device
		Bit 13 - Reserved
		Bits 12, 11 - Format support (values in binary) = 00 - Format is not supported = 01 - Format Track is supported = 10 - Format Unit is supported = 11 - Format Track and Format Unit are supported
		Bit 10 - ST506 drive = 1 - Drive is ST506 device
		Bit 9 - Concurrent unit requests per logical ID = 0 - Not concurrent = 1 - Concurrent
		Bit 8 - Ejecting capability = 0 - Not ejectable = 1 - Ejectable
		Bit 7 - Media organization = 0 - Random = 1 - Sequential
		Bit 6 - Locking capability = 1 - Locking is supported
		Bit 5 - Read capability = 1 - Readable
		Bit 4 - Caching support = 1 - Caching is supported
		Bit 3 - Write frequency = 0 - Write once = 1 - Write many
		Bit 2 - Change-signal support = 1 - Signal is supported
		Bit 1 - Power is on or off = 1 - Power off
		Bit 0 - Parameters are valid or not valid = 1 - Parameters not valid
Byte	16H	Logical unit number (LUN); supported only if SCSI device
DWord	18H	Physical number of cylinders that are associated with the unit in the request block
Byte	1CH	Physical number of heads that are associated with the unit in the request block
Byte	1DH	Suggested number of software retries for retryable operations
DWord	20H	Physical number of relative block addresses that are associated with the unit in the request block
DWord	24H	Reserved
Word	28H	Reserved
Word	2CH	Maximum number of blocks to be transferred per one call

#### 04H—Set Device Parameters (Reserved)

#### 05H—Reset/Initialize

- This function resets the disk system to an initial state.
- All return-code values are possible.

## Service-Specific Input

Size	Offset	Description
Word	10H	Reserved

## Service-Specific Output

Size	Offset	Description
DWord	28H	Time to wait before resuming request, in microseconds

### 06H—Enable (Reserved)

### 07H—Disable (Reserved)

### 08H—Read

- The Read function transfers data from the specified relative block address to the specified memory location. The Number of Blocks to Be Read field contains the amount of the data that is to be transferred.
- If the Number of Blocks to Be Read field is set to 0, no action is performed.
- If the value in the Number of Blocks to Be Read field is greater than the maximum number of blocks, no action is performed, and the Return Code field is set to hex C005 (Invalid Count Value).
- The Number of Blocks Read field contains the amount of the data that was transferred.
- When a parameter error is returned, the Number of Blocks Read field is not updated. Also, when a hex 8000, 8001, 8002, 8003, or 800F error is returned, the Number of Blocks Read field is not updated.
- All return-code values are possible.

## Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Data pointer 1
Word	16H	Reserved
Word	18H	Reserved
DWord	1EH	Data pointer 2
Word	1EH	Reserved
DWord	20H	Relative block address
DWord	24H	Reserved
Word	2CH	Number of blocks to be read
Byte	2EH	Flags
		Bits 7 to 1 - Reserved (set to 0)
		Bit 0 - Caching
		= 0 - Caching is OK for this request
		= 1 - Do not cache on this request

## Service-Specific Output

Size	Offset	Description
DWord	28H	Time to wait before resuming request, in microseconds
Word	2CH	Number of blocks read
Word	2FH	Soft-error indicator
		= 00H - Soft error did not occur
		≠ 00H - Soft error occurred

## 09H—Write

- The Write function transfers data from the specified memory location to the specified relative block address. The Number of Blocks to Be Written field contains the amount of the data that is to be transferred.
- If the Number of Blocks to Be Written field is set to 0, no action is performed.
- If the value in the Number of Blocks to Be Written field is greater than the maximum number of blocks, no action is performed, and the Return Code field is set to hex C005 (Invalid Count Value).
- The Number of Blocks Written field contains the amount of the data that was transferred.
- When a parameter error is returned, the Number of Blocks Written field is not updated. Also, when a hex 8000, 8001, 8002, 8003, or 800F error is returned, the Number of Blocks Written field is not updated.
- All return-code values are possible.



## Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Data pointer 1
Word	16H	Reserved
Word	18H	Reserved
DWord	1AH	Data pointer 2
Word	1EH	Reserved
DWord	20H	Relative block address
DWord	24H	Reserved
Word	2CH	Number of blocks to be written
Byte	2EH	Flags
		Bits 7 to 1 - Reserved (set to 0)
		Bit 0 - Caching
		= 0 - Caching is OK for this request
		= 1 - Do not cache on this request

## Service-Specific Output

Size	Offset	Description
DWord	28H	Time to wait before resuming request, in microseconds
Word	2CH	Number of blocks written
Word	2FH	Soft-error indicator
		= 00H - Soft error did not occur
		≠ 00H - Soft error occurred

## 0AH—Write Verify

- The Write Verify function operates similarly to the Write function with the addition of a Read Verify function.
- If the Number of Blocks to Be Written/Verified field is set to 0, no action is performed.
- If the value in the Number of Blocks to Be Written/Verified field is greater than the maximum number of blocks, no action is performed, and the return code field is set to hex C005 (Invalid Count Value).
- The Number of Blocks Written field contains the amount of the data that was transferred.
- When a parameter error is returned, the Number of Blocks Written field is not updated. Also, when a hex 8000, 8001, 8002, 8003, or 800F error is returned, the Number of Blocks Written field is not updated.
- All return-code values are possible.

### Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Data pointer 1
Word	16H	Reserved
Word	18H	Reserved
DWord	1AH	Data pointer 2
Word	1EH	Reserved
DWord	20H	Relative block address
DWord	24H	Reserved
Word	2CH	Number of blocks to be written/verified
Byte	2EH	Flags
		Bits 7 to 1 - Reserved (set to 0)
		Bit 0 - Caching
		= 0 - Caching is OK for this request
		= 1 - Do not cache on this request
Word	31H	Reserved

### Service-Specific Output

Size	Offset	Description
DWord	28H	Time to wait before resuming request, in microseconds
Word	2CH	Number of blocks written
Word	2FH	Soft-error indicator
		= 00H - Soft error did not occur
		≠ 00H - Soft error occurred

### 0BH—Verify

- The Verify function reads from the specified relative block address without transferring any data to system memory. This function verifies the readability of the data.
- If the Number of Blocks to Be Verified field is set to 0, no action is performed.
- If the value in the Number of Blocks to Be Verified field is greater than the maximum number of blocks, no action is performed, and the Return Code field is set to hex C005 (Invalid Count Value).
- All return-code values are possible.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved
Word	18H	Reserved
Word	1EH	Reserved
DWord	20H	Relative block address
DWord	24H	Reserved
Word	2CH	Number of blocks to be verified

## Service-Specific Output

Size	Offset	Description
DWord	28H	Time to wait before resuming request, in microseconds
Word	2FH	Soft-error indicator = 00H - Soft error did not occur ≠ 00H - Soft error occurred

## 0CH—Interrupt Status

- This function returns the disk-interrupt-pending status. It does not reset the interrupt condition.
- After BIOS checks the Unit field for validity, the Unit field is not used in determining the interrupt status. The interrupt status reports that an interrupt is pending on the logical ID, and it might or might not be the Unit field of the request block.
- If there is a parameter error, the Interrupt Status field is undefined.
- The possible values of the Return Code field are hex 0000 and 8003.

## Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
Byte	10H	Interrupt status = 00H - Interrupt not pending = 01H - Interrupt pending = 02H to FFH - Reserved

## 10H—Set DMA Pacing Factor

- This function sets the DMA pacing for the adapter of the specified drive. All devices that are attached to this adapter are also affected.
- The pacing value is expressed as a percentage; valid values are from 25 to 100, inclusive.
- BIOS does not check the range of the pacing value. Using a value outside the specified range might cause an adapter error.
- No commands can be pending when the request is made.
- All return-code values are possible.

### Service-Specific Input

Size	Offset	Description
Byte	10H	Pacing value; valid values are from 25% to 100%
Word	16H	Reserved

### Service-Specific Output

Size	Offset	Description
DWord	28H	Time to wait before resuming request, in microseconds

## 11H—Return DMA Pacing Factor

- This function returns the current pacing value of the adapter for the specified drive.
- The pacing value is expressed as a percentage; valid values are from 25% to 100%, inclusive.
- The possible values of the Return Code field are hex 0000 and 8003.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

### Service-Specific Output

Size	Offset	Description
Byte	10H	Current pacing value

## 12H—Transfer SCB

- This function programs the adapter to process the subsystem control block (SCB) that is pointed to by the Physical Pointer to SCB field. (See the technical reference manual for the SCSI Adapter.)
- The SCB-transfer-support bit (bit 15 of the Device Control Flags field) indicates whether this function is supported.
- ABIOS does not check the validity of the SCB.
- If the adapter reports an error, ABIOS determines whether a termination status block (TSB) was returned. ABIOS evaluates the TSB and returns the appropriate error code and the pointer to the failing SCB header. Check bit 0 of the Status field to determine whether the Logical Pointer to Chain Header field (offset hex 1E on output) is valid.

The SCB chain header has the following format:

Size	Offset	Description
Word	00H	Reserved
DWord	02H	Logical pointer to next SCB header in chain, or chain-ending indicator (0)
Word	06H	Reserved
Word	08H	Reserved
DWord	0AH	Logical pointer to TSB that is associated with this SCB
Word	0EH	Reserved

- The TSB for each SCB in the chain is examined to determine which SCB caused the error; bit 0 in word 0 of the TSB is used to determine whether the error occurred for that SCB. Therefore, the caller must ensure that one of the following occurs:
  - A TSB is returned for each SCB (which can degrade system performance).
  - A TSB is returned only on an error. In this case, the controlling program must ensure that bit 0 of word 0 in the TSB is set to 1 before ABIOS is invoked.
- A logical pointer of 0 ends the SCB chain.
- The chain must have an ending.
- Chains that are passed to BIOS are not dynamically updated.
- If the Logical Pointer to SCB Chain Header field (offset hex 16 on input) is set to 0, ABIOS does not initiate the SCB transfer, and the Return Code field is set to hex 0000 (Operation Successfully Completed).
- The chain header must immediately precede the SCB.

- All return-code values are possible.

See “Transfer SCB Request Block” on page 6-ID02-19 for block diagrams.

### Service-Specific Input

Size	Offset	Description
DWord	10H	Physical pointer to SCB
Word	14H	Reserved
DWord	16H	Logical pointer to SCB chain header
Word	1CH	Reserved
Word	26H	Reserved
Word	2CH	Reserved
Byte	2EH	Flags
		Bits 7 to 1 - Reserved (set to 0)
		Bit 0 - SCB length
		= 0 - Normal-length SCB
		= 1 - Long SCB

### Service-Specific Output

Size	Offset	Description
DWord	1EH	Logical pointer to chain header of last SCB processed; check the Status field (offset hex 32) for validity
DWord	28H	Time to wait before resuming request, in microseconds
Word	2FH	Soft-error indicator
		= 00H - Soft error did not occur
		≠ 00H - Soft error occurred
Byte	32H	Status
		Bits 7 to 1 - Reserved
		Bit 0 = 1 - Pointer at offset hex 1E is valid

### 13H—Reserved

### 14H—Deallocate

- This function removes the association between the physical devices and the logical ID.
- All devices that are assigned to the logical ID are released.
- After a logical ID is deallocated:
  - The logical ID cannot be used.
  - The default interrupt handler for that logical ID returns hex 0005 (Not My Interrupt, Stage on Interrupt).
- This function is intended to be used in conjunction with the Allocate SCSI Peripheral Device function (hex 15 in “Device ID 18—SCSI Peripheral Type”). Access to the device is gained through a logical ID by deallocating the disk logical ID. There is an individual SCSI-peripheral-type logical ID for each unit.

- On return, the SCSI Disk Number field contains a value that can be combined with a unit number and used as input for the Allocate function. This allows the controlling program to request the same device (within its class) that it has just released.

For example, if the disk logical ID has two units, the Allocate SCSI Peripheral Device function must be called twice. The first call is for the first unit; use the value that is returned in the SCSI Disk Number field. The second call is for the second unit; add 1 to the value that is returned in the SCSI Disk Number field.

- The Return Logical ID Parameters function (hex 01), bit 4 of the Logical ID Flags field indicates whether SCSI ABIOS is supported. The Read Device Parameters function (hex 03), bit 14 of the Device Control Flags field indicates whether the device is a SCSI device.
- If any of the devices that are assigned to this logical ID are busy, the Return Code field is set to hex 8000 (Device Busy, Request Refused).
- The possible values of the Return Code field are hex 0000, 8000, and 8003.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

### Service-Specific Output

Size	Offset	Description
Word	12H	SCSI disk number - Can be used in the Allocate SCSI Peripheral Device function to identify which disk within this class is to be requested (offset hex 12 on input)

### | 18H—Set Physical Pointer to Device Block

- A physical pointer to the device block is stored in each SCSI fixed disk device block. This function should be called if the operating system moves the device block after ABIOS is initialized. This function updates the stored physical pointer.
- Each device block must be in physically-contiguous memory.
- The physical pointer is range checked to ensure that the device block will fit in memory.
- The possible values of the Return Code field are hex 0000 and C006.

## **| Service-Specific Input**

<b>Size</b>	<b>Offset</b>	<b>Description</b>
Word	18H	Reserved
DWord	1AH	Physical pointer to device block
Word	1EH	Reserved

## **| Service-Specific Output**

<b>Size</b>	<b>Offset</b>	<b>Description</b>
None		

## **| 19H—Return Pointers to Device Block**

- A physical pointer to the device block is stored in each SCSI fixed disk device block. This function returns the logical pointer to the device block (as passed in the ABIOS stack frame on this call) and the current physical pointer that is stored in the device block.
- The information that is returned in this function is to be used by operating systems that relocate the device blocks after ABIOS is initialized. The physical pointer might or might not be valid. If the device block is moved after ABIOS is initialized, the operating system must update the physical pointer to enable ABIOS to transfer command blocks to the hardware.
- The value of the Return Code field is hex 0000.

## **| Service-Specific Input**

<b>Size</b>	<b>Offset</b>	<b>Description</b>
Word	10H	Reserved

## **| Service-Specific Output**

<b>Size</b>	<b>Offset</b>	<b>Description</b>
DWord	12H	Logical pointer to device block
DWord	1AH	Physical pointer to device block
Word	20H	Device-block length



### **| 1AH—Return SCSI-Specific Parameters**

- This function returns information about the SCSI fixed disk.
- Using the returned information to program the hardware directly is not recommended.
- To protect the IML portion of the fixed disk, use the Read Device Parameters function (hex 03) to determine the maximum available relative block address.
- The possible values of the Return Code field are hex 0000 and 8003.

### **| Service-Specific Input**

<b>Size</b>	<b>Offset</b>	<b>Description</b>
Word	10H	Reserved

### **| Service-Specific Output**

<b>Size</b>	<b>Offset</b>	<b>Description</b>
Byte	12H	Physical unit number
Byte	13H	Logical unit number
Byte	14H	Logical device number
Byte	15H	Adapter index (0 based)
Word	16H	Base port address
Word	18H	Reserved
Word	1AH	Reserved

## Return Codes

Return codes are returned at offset hex 0C.

Value	Description
0000H	Operation Successfully Completed
0001H	Stage on Interrupt
0002H	Stage on Time
0005H	Not My Interrupt, Stage on Interrupt
8000H	Device Busy, Request Refused
8001H	Device Not Powered-On
8002H	Device Block Not Properly Initialized
8003H	Device Not Allocated
800FH	DMA Arbitration Level Out of Range
8100H	Retryable Device Busy, Request Refused
9001H	Bad Command
9002H	Address Mark Not Found
9003H	Write-Protect Error
9004H	Record Not Found
9005H	Reset Failed
9006H	Media Changed
9007H	Controller Parameter Activity Failed
9008H	DMA Failed
900AH	Defective Sector
900BH	Bad Track
900DH	Format Error
900EH	CAM Detected during Read or Verify
9010H	Uncorrectable ECC or CRC Error
9014H	Device Failed
9015H	Bus Fault
9020H	Bad Controller
9021H	Equipment Check
9040H	Bad Seek
9080H	Device Did Not Respond
90AAH	Drive Not Ready
90BBH	Undefined Error
90CCH	Write Fault
90E0H	Status Error
90FFH	Incomplete Sense Operation
9101H	Bad Command
9102H	Address Mark Not Found
9103H	Write-Protect Error
9104H	Record Not Found
9105H	Reset Failed
9106H	Media Changed
9107H	Controller Parameter Activity Failed
9108H	DMA Failed
9114H	Device Failed
9115H	Bus Fault
9120H	Bad Controller
9121H	Equipment Check

Figure 6-8 (Part 1 of 3). Fixed Disk Return Codes

<b>Value</b>	<b>Description</b>
9140H	Bad Seek
9180H	Device Did Not Respond
91AAH	Drive Not Ready
91BBH	Undefined Error
91CCH	Write Fault
91E0H	Status Error
91FFH	Incomplete Sense Operation
A000H	Time-Out Occurred, No Other Error
A001H	Bad Command
A002H	Address Mark Not Found
A004H	Record Not Found
A005H	Reset Failed
A007H	Parameter Activity Failed
A00AH	Defective Sector
A00BH	Bad Track
A00DH	Invalid Sector on Format
A00EH	CAM Detected during Read or Verify
A010H	Uncorrectable ECC or CRC Error
A011H	ECC Corrected Data Error
A020H	Bad Controller
A021H	Equipment Check
A040H	Bad Seek
A080H	Device Did Not Respond
A0AAH	Drive Not Ready
A0BBH	Undefined Error
A0CCH	Write Fault
A0FFH	Incomplete Sense Operation
A100H	Time-Out Occurred, No Other Error
A105H	Reset Failed
A107H	Controller Parameter Activity Failed
A120H	Bad Controller
A121H	Equipment Check
A140H	Bad Seek
A180H	Device Did Not Respond
A1AAH	Drive Not Ready
A1BBH	Undefined Error
A1CCH	Write Fault
A1FFH	Incomplete Sense Operation
B001H	Bad Command
B020H	Bad Controller
B021H	Equipment Check
B080H	Device Did Not Respond
B0BBH	Undefined Error
B0FFH	Sense Failed
B101H	Bad Command
B120H	Bad Controller
B121H	Equipment Check
B180H	Device Did Not Respond

**Figure 6-8 (Part 2 of 3). Fixed Disk Return Codes**

<b>Value</b>	<b>Description</b>
B1BBH	Undefined Error
B1FFH	Sense Failed
C000H	Invalid Logical ID (ABIOS transfer convention only)
C001H	Invalid Function
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length
C005H	Invalid Count Value
C006H	Range Exceeded
C007H	Invalid Disk Parameter

**Figure 6-8 (Part 3 of 3). Fixed Disk Return Codes**

### **Programming Considerations**

- In ABIOS, the disk interface requires the use of the DMA ABIOS interface; therefore, if the disk routines are initialized and used, the DMA ABIOS routines must be initialized.
- The Read Device Parameters (hex 03) returns the number of retries for any one operation when an error occurs.
- When an error occurs, ABIOS resets the disk system, if necessary.
- For the Read, Write, and Write/Verify functions, the output parameter at offset hex 2C represents the number of blocks that were transferred, as determined by the hardware. This value is supplied when the request ends in an error before the data transfer is completed. If no error is reported, this value equals the number of blocks that were requested to be transferred. This value is valid only when the request is completed (successfully or unsuccessfully).
- When error-recovery procedures are successful and reported, fixed disk routines attempt to determine the nature of the recovery procedures that were performed. Fixed disk routines set the Soft Error field in the transfer SCB request block with the recovered error code.
- Relative block addresses begin ordering with the first disk block that is assigned a value of 0. For hardware devices that do not support relative block addresses, the equivalent is cylinder 0, head 0, and sector 1. In the following formulas, "sectors per track," "sector ID," "heads," and "cylinders" refer to physical (1 based) entities. "Cylinder" and "head" refer to ID values as they are actually sent to the controller (0 based). Fixed disk ABIOS returns physical values for the number of sectors per track, the number of heads, and the number of cylinders in the Read Device

Parameters function (hex 03). These values should be used for relative-block-address calculations. ABIOS uses the following formulas to break down the relative block address (RBA):

$$\text{Sector ID} = (\text{RBA MOD Sectors per track}) + 1$$

$$\text{Head} = (\text{RBA/Sectors per track}) \times \text{MOD heads}$$

$$\text{Cylinder} = (\text{RBA/Sectors per track})/\text{Heads}$$

The RBA is calculated as follows:

$$\text{RBA} = (\text{Sectors per track} \times \text{Heads} \times \text{Cylinder}) + (\text{Sectors per track} \times \text{Head}) + (\text{Sector ID} - 1)$$

The number of RBAs is calculated as follows (this is the value that is returned by the Read Device Parameters function):

$$\text{RBAs} = \text{Cylinders} \times \text{Heads} \times \text{Sectors per track}$$

The maximum allowable RBA is calculated as follows:

$$\text{Maximum RBA} = (\text{Cylinders} \times \text{Heads} \times \text{Sectors per track}) - 1$$

- When fixed disk ABIOS and fixed disk BIOS requests are issued, the following rules must be followed:
  - Do not attempt an ABIOS call while there is an outstanding BIOS call.
  - Do not attempt a BIOS call while there is an outstanding ABIOS call.
  - The Reset/Initialize function (hex 05) must be issued after ABIOS initialization has been completed.

## Device Control Flags

The following flags are returned at offset hex 14 of the Read Device Parameters function (hex 03).

<b>SCB transfer support</b>	This bit indicates whether ABIOS supports the Transfer SCB function (hex 12).
<b>SCSI device</b>	This bit indicates whether the attached device is a SCSI device. It can also be used to test for the validity of the LUN field. This interface supports SCSI peripheral type-0 devices with nonremovable media and 512-byte block states.
<b>Power off</b>	This bit reflects the power state of the device at initialization. When this bit is set to 1, the device is powered-off. When this bit is set to 0, the device is powered-on.
<b>Parameter validity</b>	This bit indicates the validity of the returned values in the Sector, Head, Block Size, Cylinder, and Maximum RBA fields. When this bit is set to 1, the parameters are not valid. When this bit is set to 0, the parameters are valid. In most cases, when the parameters are not valid, it is because the disk drive was powered-off during ABIOS initialization.

## Device Block

Subsystem control blocks (SCB) and termination status blocks (TSB) require a physical pointer for the adapter. Therefore, the physical pointer to the device block (as calculated during ABIOS initialization) is kept in the device block and used when the SCB or TSB is required. This requirement places a restriction on relocating the disk device block. It cannot be relocated after ABIOS is initialized.

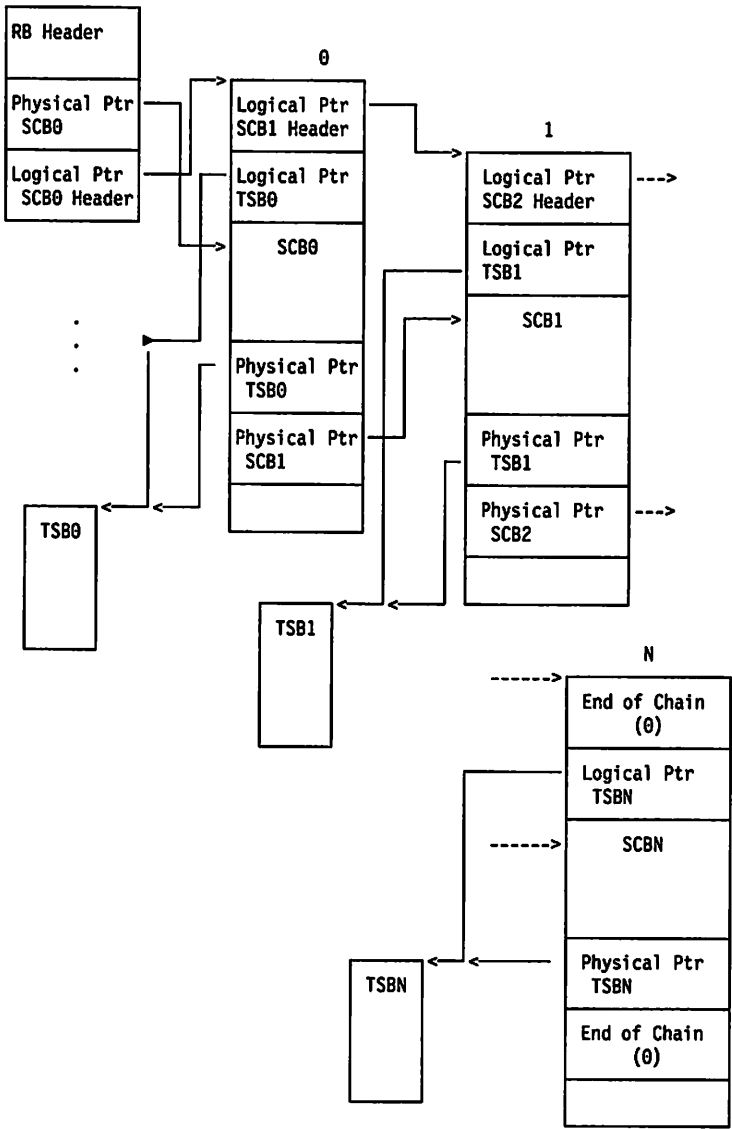
## Transfer SCB Request Block

Size	Offset	Description
DWord	10H	Physical pointer to SCB
Word	14H	Reserved
DWord	16H	Logical pointer to SCB chain header
Word	1CH	Reserved
DWord	1EH	Logical pointer to last SCB that was processed
Word	26H	Reserved
DWord	28H	Time to wait before resuming request (stage on time)
Word	2CH	Reserved
Byte	2EH	Flags
Word	2FH	Soft error
Byte	32H	Status (byte)

## Chain Header

Size	Offset	Description
Word	00H	Reserved
DWord	02H	Logical pointer to next SCB chain header
Word	06H	Reserved
Word	08H	Reserved
DWord	0AH	Logical pointer to TSB
Word	0EH	Reserved
	10H	Start of SCB

Chain Example





## Device ID 03H—Video

### Functions

The following are the video functions. The Default Interrupt Handler function (hex 00) and the Return Logical ID Parameters function (hex 01) are described in "Request Block" in the "Transfer Conventions" section.

**Note:** All reserved input fields must be set to 0.

### 00H—Default Interrupt Handler

### 01H—Return Logical ID Parameters

### 02H—Reserved

### 03H—Read Device Parameters

- This function returns parameters that indicate the current video state.
- The Character Block Specifier field returns the active character-generator blocks. The Character Block Select A field specifies the block that is used to generate alphanumeric characters when bit 3 of the character-attribute byte is set to 1. The Character Block Select B field specifies the block that is used to generate alphanumeric characters when bit 3 of the character-attribute byte is set to 0. When the value in the Character Block Select A field is equal to the value in the Character Block Select B field, character selection is disabled, and bit 3 of the character-attribute byte determines the foreground-intensity state (1 = On, 0 = Off).
- The Save/Restore Header Size field, Hardware State Size field, Device Block State Size field, and DAC State Size field are used in calculating the size of the save buffer for the Save Environment function (hex 0C). Refer to the Save Environment function (hex 0C) on page 6-ID03-5 for more information.
- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size Word	Offset 28H	Description Reserved
--------------	---------------	-------------------------

## Service-Specific Output

Size	Offset	Description
Byte	1CH	Number of scan lines on the screen = 00H - 200 scan lines = 01H - 350 scan lines = 02H - 400 scan lines = 03H - 480 scan lines = 04H to 0FFH - Reserved
Word	1EH	Video mode setting (see Figure 6-10 on page 6-ID03-14)
Word	20H	Type of display attached Bits 15 to 1 - Reserved Bit 0 - Color or monochrome = 0 - Color display = 1 - Monochrome display
Word	22H	Character height (bytes per character)
Word	24H	Character-block specifier Bits 15 to 12 - Reserved Bits 11 to 8 - Character block select A Bits 7 to 4 - Reserved Bits 3 to 0 - Character block select B
Word	2AH	Size of data buffer that is required for the Return ROM Fonts Information function (hex 0B)
Word	2EH	Size of the save/restore buffer header, in bytes
Word	30H	Size of the save/restore hardware state, in bytes
Word	32H	Size of the save/restore device-block state, in bytes
Word	34H	Size of the save/restore digital-to-analog converter (DAC) state, in bytes

## 04H—Set Device Parameters (Reserved)

### 05H—Reset/Initialize

- This function initializes the video controller to the requested mode (see Figure 6-10 on page 6-ID03-14).
- The Character Blocks to Be Loaded field indicates which character blocks will be loaded with the default ROM character font for the specified mode and number of scan lines. This parameter is required only when an alphanumeric mode (hex 0, 1, 2, 3, 7, or 14) is being set.
- The Number of Scan Lines field and the Character Block Specifier field are specified only when an alphanumeric mode (hex 0, 1, 2, 3, 7, or 14) is being set.
- In the Character Block Specifier field, the Character Block Select A field specifies the block that is used to generate alphanumeric characters when bit 3 of the character-attribute byte is set to 1. The Character Block Select B field specifies the block that is used to generate alphanumeric characters when bit 3 of the character-attribute byte is set to 0. When the value in the Character Block Select A field is equal to the value in the

Character Block Select B field, character selection is disabled, and bit 3 of the character-attribute byte determines the foreground-intensity state (1 = On, 0 = Off).

- The summing bit of the Device Control Flags field is required only when a color display is attached. Summing is performed automatically for monochrome displays.
- When a monochrome display is used in a color mode, the colors are displayed as shades of gray. Of 64 gray shades, 16 are available in all modes except mode hex 13. In mode hex 13, all 64 gray shades are available.
- Modes hex 0, 2, and 4 are identical to modes hex 1, 3, and 5, respectively.
- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Word	1AH	Device control flags Bits 15 to 3 - Reserved Bit 2 - Summing = 0 - Summing disabled = 1 - Summing enabled Bit 1 - Initialize digital-to-analog converter (DAC) to default = 0 - Do not initialize DAC to default = 1 - Initialize DAC to default Bit 0 - Regenerative buffer flag = 0 - Do not clear buffer = 1 - Clear buffer
Byte	1CH	Number of scan lines = 00H - 200 scan lines (modes 0, 1, 2, 3, and 14) = 01H - 350 scan lines (modes 0, 1, 2, 3, 7, and 14) = 02H - 400 scan lines (modes 0, 1, 2, 3, 7, and 14) = 03H to FFH - Reserved
Word	1EH	Video mode to be set (see Figure 6-10 on page 6-ID03-14)
Word	24H	Character block specifier Bits 15 to 12 - Reserved Bits 11 to 8 - Character block select A Bits 7 to 4 - Reserved Bits 3 to 0 - Character block select B
Word	26H	Character blocks to be loaded with default ROM font Bit <i>n</i> - Block <i>n</i> flag = 0 - Do not update font = 1 - Update font
Word	28H	Reserved

## Service-Specific Output

Size	Offset	Description
None		

**06H—Enable (Reserved)**

**07H—Disable (Reserved)**

**08H—Read (Reserved)**

**09H—Write (Reserved)**

**0AH—Additional Data Transfer Function (Reserved)**

**0BH—Return ROM Fonts Information**

- This function returns the following information about each of the ROM fonts:
  - The pointer to the ROM font
  - The character size (row and column)
  - Whether it is a total font or a partial font
  - If it is a partial font, which font it relates to.
- There are 12 bytes of information for each ROM font. They are stored sequentially in the specified data area.
- Each ROM-font entry has the following format:

Word - Reserved

DWord - Pointer to ROM font

Word - Reserved

Byte - Character size (number of columns)

Byte - Character size (number of rows)

Byte - Total-/partial-font indicator

= 00H - Total font

= 01H - Partial font

= 02H to FFH - Reserved

Byte - Related font

If this is a partial font, this byte contains a number to indicate which font this font goes with.

The font number is based on the place a particular font occupies in the ROM-font entries.

- Before this function is used, the Read Device Parameters function (hex 03) must be issued to determine the size of the buffer that is required to save the ROM-fonts information.
- The value of the Return Code field is hex 0000.

Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Pointer to buffer to store ROM-fonts information
Word	16H	Reserved

Service-Specific Output

Size	Offset	Description
None		

0CH—Save Environment

- This function stores the caller’s requested video states in the specified buffer.
- The video environment consists of the following states:
  - Hardware state
  - Device block state
  - Digital-to-analog converter (DAC) state.
- To calculate the size of the save buffer that is required, the Read Device Parameters function (hex 03) must be issued. It gives the individual sizes of the possible states to be saved and the size of the save/restore header. Then:

Save-buffer size = (A + B + C + D)

where:

- A = Size of the save/restore header
- B = Environment (bit 0) x (size of hardware state)
- C = Environment (bit 1) x (size of device block state)
- D = Environment (bit 2) x (size of DAC state)

- The value of the Return Code field is hex 0000.

Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Pointer to environment save area
Word	16H	Reserved
Word	2CH	Video-environment states to be saved <ul style="list-style-type: none"><li>Bits 15 to 3 - Reserved (set to 0)</li><li>Bit 2 - DAC state<ul style="list-style-type: none"><li>= 1 - Save state</li></ul></li><li>Bit 1 - Device block state<ul style="list-style-type: none"><li>= 1 - Save state</li></ul></li><li>Bit 0 - Hardware state<ul style="list-style-type: none"><li>= 1 - Save state</li></ul></li></ul>

### Service-Specific Output

Size	Offset	Description
None		

### 0DH—Restore Environment

- This function restores the video environment from the specified buffer location. Refer to the Save Environment function (hex 0C) for more information about the contents and structure of the video environment.
- Restoring a state that was not previously saved can cause unpredictable results.
- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Pointer to environment restore area
Word	16H	Reserved
Word	1AH	Device control flag <ul style="list-style-type: none"><li>Bits 15 to 1 - Reserved</li><li>Bit 0 - Regenerative-buffer flag<ul style="list-style-type: none"><li>= 0 - Do not clear buffer</li><li>= 1 - Clear buffer</li></ul></li></ul>
Word	2CH	Video-environment states to be restored <ul style="list-style-type: none"><li>Bits 15 to 3 - Reserved (set to 0)</li><li>Bit 2 - DAC state<ul style="list-style-type: none"><li>= 1 - Restore state</li></ul></li><li>Bit 1 - Device block state<ul style="list-style-type: none"><li>= 1 - Restore state</li></ul></li><li>Bit 0 - Hardware state<ul style="list-style-type: none"><li>= 1 - Restore state</li></ul></li></ul>

### Service-Specific Output

Size	Offset	Description
None		

### 0EH—Select Character-Generator Block

- This function selects up to two character-generator blocks.
- In the Character Block Specifier field, the Character Block Select A field specifies the block that is used to generate alphanumeric characters when bit 3 of the character-attribute byte is set to 1. The Character Block Select B field specifies the block that is used to generate alphanumeric characters when bit 3 of the character-attribute byte is set to 0. When the value in the Character Block Select A field is equal to the value in the

Character Block Select B field, character selection is disabled, and bit 3 of the character-attribute byte determines the foreground-intensity state (1 = On, 0 = Off).

- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved
Word	24H	Character-block specifier
		Bits 15 to 12 - Reserved
		Bits 11 to 8 - Character block select A
		Bits 7 to 4 - Reserved
		Bits 3 to 0 - Character block select B

### Service-Specific Output

Size	Offset	Description
None		

### 0FH—Alphanumeric Load

- This function loads the requested character generator, or part of it, to the specified character blocks.
- This function does not update the hardware registers. Refer to the Enhanced Alphanumeric Load function (hex 10) if hardware updating is required.
- When any of the ROM character generators is being loaded (the Character Generator Type field is set to 1, 2, or 3), the full set of characters (hex 100) is loaded. Therefore, the only parameters that are required to invoke this function are the Character Generator Type field and the Character Block Specifier field.
- When a user font is being loaded (the Character Generator Type field is set to 0), all parameters are required.
- When a user font is being loaded, if the Count of Characters field is set to 0, no character is loaded, and the Return Code field is set to hex 0000 (Operation Successfully Completed).
- When a user font is being loaded, the sum of the values in the Count of Characters field and the Character Offset field must not exceed the maximum valid number of characters in a set (hex 100). If the sum does exceed the maximum valid number of characters, the Return Code field is set to hex C005 (Invalid Video Parameter).
- The possible values of the Return Code field are hex 0000 and C005.

## Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Pointer to user font
Word	16H	Reserved
Word	18H	Count of characters = 001H to 100H - Valid count of characters
Byte	1DH	Character-generator type: = 00H - User's alphanumeric font = 01H - 8x8 alphanumeric ROM font = 02H - 8x14 alphanumeric ROM font = 03H - 8x16 alphanumeric ROM font = 04H to FFH - Reserved
Word	22H	Character height (bytes per character)
Word	24H	Character block to be loaded = 0000H to 0007H - Valid values of character blocks to be loaded = 0008H to FFFFH - Reserved
Word	28H	Character offset into the table

## Service-Specific Output

Size	Offset	Description
None		

### 10H—Enhanced Alphanumeric Load

- This function loads the requested character generator, or part of it, to the specified character block and updates the hardware registers.
- When any of the ROM character generators is being loaded (the Character Generator Type field is set to 1, 2, or 3), the full set of characters (hex 100) is loaded. Therefore, the only parameters that are required to invoke this function are the Character Generator Type field and the Character Blocks to Be Loaded field.
- When a user font is being loaded (the Character Generator Type field is set to 0), all parameters are required.
- When a user font is being loaded, if the Count of Characters field is set to 0, no character is loaded, and the Return Code field is set to hex 0000 (Operation Successfully Completed).
- When a user font is being loaded, the sum of the values of the Count of Characters field and the Character Offset field must not exceed the maximum valid number of characters in a set (hex 100). If the sum does exceed the maximum valid number of characters, the Return Code field is set to hex C005 (Invalid Video Parameter).



- The possible values of the Return Code field are hex 0000 and C005.

### Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Pointer to user font
Word	16H	Reserved
Word	18H	Count of characters = 001H to 100H - Valid count of characters
Byte	1DH	Character-generator type = 00H - User's alphanumeric font = 01H - 8x8 alphanumeric ROM font = 02H - 8x14 alphanumeric ROM font = 03H - 8x16 alphanumeric ROM font = 04H to FFH - Reserved
Word	22H	Character height (bytes per character)
Word	24H	Character block to be loaded = 0000H to 0007H - Valid values of character blocks to be loaded = 0008H to FFFFH - Reserved
Word	28H	Character offset into the table

### Service-Specific Output

Size	Offset	Description
None		

### 11H—Read Palette Register

- This function reads a palette register.
- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved
Word	32H	Palette register to be read = 0000H to 000FH - Valid values of palette register to be read = 0010H to FFFFH - Reserved

### Service-Specific Output

Size	Offset	Description
Word	34H	Palette value that was read

### 12H—Write Palette Register

- This function writes a value to a palette register.
- Execution of this function is not supported in mode hex 13. The hardware requires that the values in these registers not be changed after they are set by the Reset/Initialize function (hex 05). Changing the values in these registers can cause unpredictable results.
- The value of the Return Code field is hex 0000.

#### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved
Word	32H	Palette register to be written = 0000H to 000FH - Valid values of palette register to be written = 0010H to FFFFH - Reserved
Word	34H	Palette value to be loaded = 0000H to 003FH - Valid palette values = 0040H to FFFFH - Reserved

#### Service-Specific Output

Size	Offset	Description
None		

### 13H—Read Color Register

- This function reads the red, green, and blue values of a color register from the video digital-to-analog converter.
- The value of the Return Code field is hex 0000.

#### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved
Word	2AH	Color register to be read = 0000H to 00FFH - Valid values of color register to be read = 0100H to FFFFH - Reserved

#### Service-Specific Output

Size	Offset	Description
Word	2CH	Red value that was read
Word	2EH	Green value that was read
Word	30H	Blue value that was read

### 14H—Write Color Register

- This function loads a digital-to-analog converter color register with the specified red, green, and blue values.
- In the Device Control Flags field, the summing bit is disregarded when a monochrome display is attached. Summing always occurs with a monochrome display that is operating in a color mode.
- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved
Word	1AH	Device control flags <ul style="list-style-type: none"><li>Bits 15 to 3 - Reserved</li><li>Bit 2 - Summing<ul style="list-style-type: none"><li>= 0 - Summing disabled</li><li>= 1 - Summing enabled</li></ul></li><li>Bits 1, 0 - Reserved</li></ul>
Word	2AH	Color register to be written <ul style="list-style-type: none"><li>= 0000H to 00FFH - Valid values of color registers to be written</li><li>= 0100H to FFFFH - Reserved</li></ul>
Word	2CH	Red value to be written <ul style="list-style-type: none"><li>= 0000H to 003FH - Valid red values to be written</li><li>= 0040H to FFFFH - Reserved</li></ul>
Word	2EH	Green value to be written <ul style="list-style-type: none"><li>= 0000H to 003FH - Valid green values to be written</li><li>= 0040H to FFFFH - Reserved</li></ul>
Word	30H	Blue value to be written <ul style="list-style-type: none"><li>= 0000H to 003FH - Valid blue values to be written</li><li>= 0040H to FFFFH - Reserved</li></ul>

### Service-Specific Output

Size	Offset	Description
None		

### 15H—Read Block of Color Registers

- This function reads a block of digital-to-analog converter color registers into the specified save area, beginning at the specified color register.
- The format of the data that is returned is “red value, green value, blue value, red value, green value, blue value, . . . , red value, green value, blue value.”
- The range for the red, green, and blue values is from hex 00 to hex 3F.

- If the Number of Color Registers to Be Read field is set to 0, no action is performed, and the Return Code field is set to hex 0000 (Operation Successfully Completed).
- If the sum of the values of the First Color Register to Be Read field and the Number of Color Registers to Be Read field is greater than the maximum valid number of color registers, no action is performed, and the Return Code field is set to hex C005 (Invalid Video Parameter).
- The possible values of the Return Code field are hex 0000 and C005.

### Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Pointer to read save area
Word	16H	Reserved
Word	18H	Number of color registers to be read
Word	2AH	First color register to be read = 0000H to 00FFH - Valid values of first color register to be read = 0100H to FFFFH - Reserved

### Service-Specific Output

Size	Offset	Description
None		

### 16H—Write Block of Color Registers

- This function loads a block of digital-to-analog converter color registers with the requested values, beginning with the requested color register.
- The format of the data that is returned is "red value, green value, blue value, red value, green value, blue value, . . . , red value, green value, blue value."
- If the Number of Color Registers to Be Written field is set to 0, no action is performed, and the Return Code field is set to hex 0000 (Operation Successfully Completed).
- If the sum of the values of the First Color Register to Be Written field and the Number of Color Registers field is greater than the maximum valid number of color registers, no action is performed, and the Return Code field is set to hex C005 (Invalid Video Parameter).
- In the Device Control Flags field, the summing bit is disregarded when a monochrome display is attached. Summing always

occurs with a monochrome display that is operating in a color mode.

- The possible values of the Return Code field are hex 0000 and C005.

**Service-Specific Input**

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Pointer to write save area
Word	16H	Reserved
Word	18H	Number of color registers to be written
Word	1AH	Device control flags
		Bits 15 to 3 - Reserved
		Bit 2 - Summing
		= 0 - Summing disabled
		= 1 - Summing enabled
		Bits 1, 0 - Reserved
Word	2AH	First color register to be written
		= 0000H to 00FFH - Valid value of first color register to be written
		= 0100H to FFFFH - Reserved

**Service-Specific Output**

Size	Offset	Description
None		

## Return Codes

Return codes are returned at offset hex 0C.

Value	Description
0000H	Operation Successfully Completed
C000H	Invalid Logical ID (ABIOS transfer convention only)
C001H	Invalid Function
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length
C005H	Invalid Video Parameter

Figure 6-9. Video Return Codes

## Video Modes

The following table shows the supported video modes.

Mode (Hex)	Type	Maximum Colors	A/N Format	Buffer Start	Box Size	Maximum Pages	Display Size
00	A/N	16	40x25	B8000H	8x8	8	320x200
00	A/N	16	40x25	B8000H	8x14	8	320x350
00	A/N	16	40x25	B8000H	8x16	8	320x400
00	A/N	16	40x25	B8000H	9x16	8	360x400
01	A/N	16	40x25	B8000H	8x8	8	320x200
01	A/N	16	40x25	B8000H	8x14	8	320x350
01	A/N	16	40x25	B8000H	8x16	8	320x400
01	A/N	16	40x25	B8000H	9x16	8	360x400
02	A/N	16	80x25	B8000H	8x8	4	640x200
02	A/N	16	80x25	B8000H	8x8	8	640x200
02	A/N	16	80x25	B8000H	8x14	8	640x350
02	A/N	16	80x25	B8000H	8x16	8	640x400
02	A/N	16	80x25	B8000H	9x16	8	720x400
03	A/N	16	80x25	B8000H	8x8	4	640x200
03	A/N	16	80x25	B8000H	8x8	8	640x200
03	A/N	16	80x25	B8000H	8x14	8	640x350
03	A/N	16	80x25	B8000H	8x16	8	640x400
03	A/N	16	80x25	B8000H	9x16	8	720x400
04	APA	4	40x25	B8000H	8x8	1	320x200
05	APA	4	40x25	B8000H	8x8	1	320x200
06	APA	2	80x25	B8000H	8x8	1	640x200
07	A/N	Monochrome	80x25	B0000H	9x14	1	720x350
07	A/N	Monochrome	80x25	B0000H	9x14	8	720x350
07	A/N	Monochrome	80x25	B0000H	9x16	8	720x400
07	A/N	Monochrome	80x25	B0000H	8x8	4	640x200
08	APA	16	20x25	B0000H	8x8	1	160x200
09	APA	16	40x25	B0000H	8x8	1	320x200

Figure 6-10 (Part 1 of 2). Video Modes

Mode (Hex)	Type	Maximum Colors	A/N Format	Buffer Start	Box Size	Maximum Pages	Display Size
0A	APA	4	80x25	B0000H	8x8	1	640x200
0B	Reserved						
0C	Reserved						
0D	APA	16	40x25	A0000H	8x8	8	320x200
0E	APA	16	80x25	A0000H	8x8	4	640x200
0F	APA	Monochrome	80x25	A0000H	8x14	2	640x350
10	APA	16	80x25	A0000H	8x14	2	640x350
11	APA	2	80x30	A0000H	8x16	1	640x480
12	APA	16	80x30	A0000H	8x16	1	640x480
13	APA	256	40x25	A0000H	8x8	1	320x200
14	A/N	16	132x25	B8000H	8x8	4	1056x200
14	A/N	16	132x25	B8000H	8x14	4	1056x350
14	A/N	16	132x25	B8000H	8x16	4	1056x400

APA = All points addressable (graphics)  
A/N = Alphanumeric (text)

*Figure 6-10 (Part 2 of 2). Video Modes*

**Notes:**





## Device ID 04H—Keyboard

### Functions

The following are the keyboard functions. The Default Interrupt Handler function (hex 00) and the Return Logical ID Parameters function (hex 01) are described in "Request Block" in the "Transfer Conventions" section.

**Note:** All reserved input fields must be set to 0.

### 00H—Default Interrupt Handler

### 01H—Return Logical ID Parameters

### 02H—Reserved

### 03H—Read Device Parameters

- This function returns the keyboard identification values.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 0005, 8000, 8003, 9000, 9002, 9003, 9004, 9100, 9102, 9103, 9104, B001, and B101.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

### Service-Specific Output

Size	Offset	Description
DWord	10H	Time to wait before resuming request, in microseconds
Byte	14H	Keyboard ID byte 1
Byte	15H	Keyboard ID byte 2

### 04H—Set Device Parameters (Reserved)

### 05H—Reset/Initialize Keyboard

- This function resets the keyboard and turns off the Num Lock, Caps Lock, and Scroll Lock indicator lights.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 0005, 8000, 8003, 9000, 9001, 9002, 9100, 9101, 9102, B001, and B101.

## Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
DWord	10H	Time to wait before resuming request, in microseconds

### 06H—Enable

- This function enables the keyboard so that keyboard data can be passed to the system.
- The possible values of the Return Code field are hex 0000, 0002, 8000, 8003, 9000, and 9100.

## Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
DWord	10H	Time to wait before resuming request, in microseconds

### 07H—Disable

- This function disables the keyboard by inhibiting the flow of keyboard data to the system.
- The possible values of the Return Code field are equal to hex 0000, 0002, 8000, 8003, 9000, and 9100.

## Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
DWord	10H	Time to wait before resuming request, in microseconds

## 08H—Continuous Read

- This function returns keyboard scan codes. It must be called soon after BIOS initialization to allow for the processing of keystroke interrupts.
- After this function has been started, it is a continuous multistaged request.
- At interrupt time, if a scan code is available, the Keyboard Interrupt routine returns hex 0009 (Attention, Stage on Interrupt) in the Return Code field. This return code indicates that there is a valid scan code in the Keyboard Raw Scan Code field.
- The possible values of the Return Code field are hex 0001, 0005, 0009, and 8000.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

### Service-Specific Output

Size	Offset	Description
Byte	14H	Keyboard raw scan code

## 09H—Write (Reserved)

## 0AH—Additional Data Transfer (Reserved)

## 0BH—Read Keyboard Indicators

- This function returns the current state of the keyboard Num Lock, Caps Lock, and Scroll Lock indicator lights.
- The possible values of the Return Code field are hex 0000 and 8000.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
Byte	14H	Keyboard-indicator data
		Bits 7 to 3 - Reserved
		Bit 2 - Caps Lock
		= 0 - Off
		= 1 - On
		Bit 1 - Num Lock
		= 0 - Off
		= 1 - On
		Bit 0 - Scroll Lock
		= 0 - Off
		= 1 - On

## 0CH—Write Keyboard Indicators

- This function programs the state of the keyboard Num Lock, Caps Lock, and Scroll Lock indicator lights.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 0005, 8000, 8003, 9000, 9002, 9100, 9102, B001, and B101.

## Service-Specific Input

Size	Offset	Description
Byte	14H	Keyboard indicator data
		Bits 7 to 3 - Reserved (must be set to 0)
		Bit 2 - Caps Lock
		= 0 - Off
		= 1 - On
		Bit 1 - Num Lock
		= 0 - Off
		= 1 - On
		Bit 0 - Scroll Lock
		= 0 - Off
		= 1 - On
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
DWord	10H	Time to wait before resuming request, in microseconds

## 0DH—Set Typematic Rate and Delay

- This function changes the current setting of the typematic rate and delay for the keyboard.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 0005, 8000, 8003, 9000, 9002, 9100, 9102, B001, and B101.

### Service-Specific Input

Size	Offset	Description
Byte	14H	Rate value
		Bits 7 to 5 - Reserved (must be set to 0)
		Bits 4 to 0 - Rate value, in characters per second (values are in binary)
		= 00000 - 30.0 = 10000 - 7.5
		= 00001 - 26.7 = 10001 - 6.7
		= 00010 - 24.0 = 10010 - 6.0
		= 00011 - 21.8 = 10011 - 5.5
		= 00100 - 20.0 = 10100 - 5.0
		= 00101 - 18.5 = 10101 - 4.6
		= 00110 - 17.1 = 10110 - 4.3
		= 00111 - 16.0 = 10111 - 4.0
		= 01000 - 15.0 = 11000 - 3.7
		= 01001 - 13.3 = 11001 - 3.3
		= 01010 - 12.0 = 11010 - 3.0
		= 01011 - 10.9 = 11011 - 2.7
		= 01100 - 10.0 = 11100 - 2.5
		= 01101 - 9.2 = 11101 - 2.3
		= 01110 - 8.6 = 11110 - 2.1
		= 01111 - 8.0 = 11111 - 2.0
Byte	15H	Delay value
		Bits 7 to 2 - Reserved (must be set to 0)
		Bits 1, 0 - Delay value, in milliseconds (values are in binary)
		= 00 - 250
		= 01 - 500
		= 10 - 750
		= 11 - 1000
Word	16H	Reserved

### Service-Specific Output

Size	Offset	Description
DWord	10H	Time to wait before resuming request, in microseconds

### 0EH—Read Keyboard Mode

- This function returns the current keyboard scan-code mode.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 0005, 8000, 8003, 9000, 9002, 9003, 9004, 9006, 9100, 9102, 9103, 9104, 9106, B001, and B101.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
DWord	10H	Time to wait before resuming request, in microseconds
Byte	14H	Current keyboard scan-code mode = 00H - Reserved = 01H - Scan code set 1 = 02H - Scan code set 2 = 03H - Scan code set 3 = 04H to FFH - Reserved

## 0FH—Set Keyboard Mode

- This function changes the current keyboard scan-code mode.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 0005, 8000, 8003, 9000, 9002, 9100, 9102, B001, and B101.

## Service-Specific Input

Size	Offset	Description
Byte	14H	Keyboard scan-code mode to be set = 00H - Reserved = 01H - Scan code set 1 = 02H - Scan code set 2 = 03H - Scan code set 3 = 04H to FFH - Reserved
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
DWord	10H	Time to wait before resuming request, in microseconds

## 10H—Write Keyboard-Controller Data String

- This function sends the requested data string to the keyboard controller.
- If the Data String Count field is set to 0, no action is performed, and the Return Code field is set to hex 0000 (Operation Successfully Completed).
- The possible values of the Return Code field are hex 0000, 0001, 0002, 0005, 8000, 8003, 9000, 9002, 9100, 9102, B001, and B101.

## Service-Specific Input

Size	Offset	Description
Word	14H	Reserved
DWord	16H	Pointer to data area
Byte	1CH	Data string count
Word	28H	Reserved

### Service-Specific Output

Size	Offset	Description
DWord	10H	Time to wait before resuming request, in microseconds

### 11H—Write Keyboard Data String

- This function sends the requested data string to the keyboard.
- If the Data String Count field is set to 0, no action is performed, and the Return Code field is set to hex 0000 (Operation Successfully Completed).
- The possible values of the Return Code field are hex 0000, 0001, 0002, 0005, 8000, 8003, 9000, 9002, 9100, 9102, B001, and B101.

### Service-Specific Input

Size	Offset	Description
Word	14H	Reserved
DWord	16H	Logical pointer to data area
Byte	1CH	Data string count
Word	28H	Reserved

### Service-Specific Output

Size	Offset	Description
DWord	10H	Time to wait before resuming request, in microseconds

## Return Codes

Return codes are returned at offset hex 0C.

Value	Description
0000H	Operation Successfully Completed
0001H	Incomplete – Stage on Interrupt
0002H	Incomplete – Stage on Time (service specific)
0005H	Incomplete – Not My Interrupt, Stage on Interrupt
0009H	Attention, Stage on Interrupt
8000H	Device Busy – Request Refused
8003H	Security Enabled, Keyboard Inhibited – Request Refused
9000H	Keyboard Controller Perpetually Busy
9001H	Keyboard Failed Reset
9002H	Resend Error
9003H	Keyboard Parity Error
9004H	General Hardware Time-Out
9006H	Undefined Mode Returned by Keyboard
9100H	Keyboard Controller Perpetually Busy
9101H	Keyboard Failed Reset
9102H	Resend Error
9103H	Keyboard Parity Error
9104H	General Hardware Time-Out
B001H	Keyboard Error
B101H	Keyboard Error
C000H	Invalid Logical ID (BIOS transfer convention only)
C001H	Invalid Function
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length
FFFFH	Return Code Is Not Valid

*Figure 6-11. Keyboard Return Codes*

## Programming Considerations

- Keyboard BIOS does not attempt any retries. The calling program is responsible for performing retries.
- The Write Keyboard Data String function (hex 11) sends bytes to the keyboard and expects an acknowledgment (ACK) after each byte is sent.
- The Write Keyboard Controller Data String function (hex 10) sends bytes to the keyboard controller; it does not expect a response.
- The Read Keyboard Indicators function (hex 0B) reflects the state of the indicators after either a successful Reset/Initialize Keyboard function (hex 05) or a successful Write Indicators function (hex 0C). If the Write Keyboard Data String function (hex 11) is used to change the indicators, the value that is returned by



the Read Indicators function (hex 0B) might not reflect the true state of the keyboard.

- The Keyboard Time-Out routine does not attempt to reset the keyboard; rather, it sets the Return Code field to hex B001 or hex B101 (Keyboard Error). The calling program is responsible for executing the Keyboard Reset/Initialize function (hex 05).
- If keyboard BIOS expects an acknowledgment from the keyboard after a command is issued to the keyboard, it does not pass the acknowledgment to the controlling program.

**Notes:**



---

## Device ID 05H—Parallel Port

### Functions

The following are the parallel port functions. The Default Interrupt Handler function (hex 00) and the Return Logical ID Parameters function (hex 01) are described in "Request Block" in the "Transfer Conventions" section.

**Note:** All reserved input fields must be set to 0.

#### 00H—Default Interrupt Handler

#### 01H—Return Logical ID Parameters

#### 02H—Reserved

#### 03H—Read Device Parameters

- This function returns the device-specific information.
- The Printer Initialization Time to Wait before Resuming Request field contains the wait-time value that is returned by the Reset/Initialize function (hex 05).
- The Printer Interrupt Time-Out field contains the time-out limit for printing. If the print request is performed using programmed I/O, this field represents the time-out limit to print each character. If the print request is performed using direct memory access (DMA), this field represents the time-out limit to print the entire block of characters.
- The value of the Return Code field is hex 0000.

#### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
Byte	16H	Support flags Bit 7 - DMA-support flag = 0 - DMA is not supported = 1 - DMA is supported Bits 6 to 0 - Reserved
Byte	17H	Reserved
DWord	20H	Printer initialization time to wait before resuming request, in microseconds
Byte	29H	Interrupt level
Word	2AH	Printer interrupt time-out Bits 15 to 3 - Time-out, in seconds Bits 2 to 0 - Reserved

### 04H—Set Device Parameters

- This function sets the device-specific information according to the input parameters.
- The Printer Initialization Time to Wait before Resuming Request field sets the wait-time value that is returned by the Reset/Initialize function (hex 05).
- The Printer Initialization Time to Wait before Resuming Request field must contain a nonzero value. If the field is set to 0, BIOS sets the Return Code field to hex C006 (Invalid Time to Wait).
- The Printer Interrupt Time-Out field contains the time-out limit for printing. If the print request is performed using programmed I/O, this field represents the time-out limit to print each character. If the print request is performed using direct memory access (DMA), this field represents the time-out limit to print the entire block of characters.
- The Printer Interrupt Time-Out field must contain a nonzero value. If the field is set to 0, BIOS sets the Return Code field to hex C005 (Invalid Time-Out).
- BIOS uses these parameters until this function is called to change them.
- The possible values of the Return Code field are hex 0000, 8000, C005, and C006.

## Service-Specific Input

Size	Offset	Description
Word	16H	Reserved
DWord	20H	Printer initialization time to wait before resuming request, in microseconds
Word	2AH	Printer interrupt time-out Bits 15 to 3 - Time-out, in seconds Bits 2 to 0 - Reserved

## Service-Specific Output

Size	Offset	Description
None		

### 05H—Reset/Initialize

- This function initializes the printer.
- After performing this function, the printer indicates a busy status while it performs a self-test.
- The Printer Status field is valid only when this function is completed. The status that is returned in the request block is not valid during intermediate stages.
- The possible values of the Return Code field are hex 0000, 0002, and 8000.

## Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
DWord	20H	Time to wait before resuming request, in microseconds
Byte	28H	Printer status Bit 7 = 1 - Busy Bit 6 = 1 - Acknowledge Bit 5 = 1 - End of paper Bit 4 = 1 - Selected Bit 3 = 1 - I/O error Bit 2 = 1 - Interrupt (non-DMA mode only) Bits 1, 0 - Reserved

### 06H—Enable (Reserved)

### 07H—Disabled (Reserved)

### 08H—Read (Reserved)

## 09H—Print Block

- This function sends a block of characters to the parallel port.
- The Data Pointer 1 field is a logical address to the data that is to be printed. A nonzero value in this field indicates that the pointer is present as input to the Print Block function. When programmed I/O is used, the Data Pointer 1 field must contain a nonzero value. See "Programming Considerations" on page 6-ID05-8 for more details.
- The Data Pointer 2 field is a physical address to the data that is to be printed. A nonzero value in this field indicates that the pointer is present as input to the Print Block function. When direct memory access (DMA) is used, the Data Pointer 2 field must contain a nonzero value. See "Programming Considerations" on page 6-ID05-8 for more details.
- The arbitration-deallocation flag (bit 7) of the Request Flags field enables the device to retain its arbitration level after a DMA transfer is completed. When this flag is set to 0 and a DMA transfer is requested, BIOS dedicates an arbitration level to the device, and any subsequent DMA requests to this device will use this already-allocated arbitration level. A call to the Cancel Print Block function (hex 0B) can be used to deallocate the dedicated arbitration level. When the arbitration-deallocation flag is set to 1 and a DMA transfer is requested, BIOS automatically deallocates the arbitration level when the transfer is completed.
- When a Print Block function is in progress, the caller must use the Cancel Print Block function (hex 0B) to cancel it before issuing another Print Block function to the same unit.
- When a print error occurs, when the printer is offline, or when the printer is busy, BIOS terminates the print-block request. The Number of Characters Printed field indicates which portion of the print block has been printed. To print the unprinted portion of the print block, issue another print-block request when the terminating condition has been corrected.
- The Printer Status field is valid only when the function is successfully completed (return code hex 0000) or when the function is terminated because of an error condition. The status that is returned in the Request Block is not valid in intermediate stages.
- The possible values of the Return Code field are hex 0000, 0001, 0005, 8000, 8001, 8008, 8081, 8082, 8083, 8084, 8085, 8086, 8087, 9000, and C007.

## Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Data pointer 1
Word	16H	Request flags
		Bit 7 - Arbitration deallocation flag
		= 0 - Do not deallocate
		= 1 - Deallocate
		Bit 6 - Arbitration allocation flag
		= 0 - Do not assume that the
		arbitration level is allocated
		by the operating system
		= 1 - Assume that the arbitration
		level is allocated by the
		operating system
		Bits 5 to 0 - Reserved
Byte	17H	Reserved
Word	18H	Reserved
DWord	1AH	Data pointer 2
Word	1EH	Reserved
Word	24H	Number of characters to be printed

## Service-Specific Output

Size	Offset	Description
Word	26H	Number of characters printed
Byte	28H	Printer status
		Bit 7 = 1 - Busy
		Bit 6 = 1 - Acknowledge
		Bit 5 = 1 - End of paper
		Bit 4 = 1 - Selected
		Bit 3 = 1 - I/O error
		Bit 2 = 1 - Interrupt (non-DMA mode only)
		Bits 1, 0 - Reserved

## 0AH—Additional Data Transfer (Reserved)

## 0BH—Cancel Print Block

- This function cancels an outstanding Print Block function (hex 09) request and deallocates any arbitration level that is allocated to the device.
- The arbitration-deallocation flag (bit 7) of the Request Flags field enables the device to retain its arbitration level after the Cancel Print Block function is completed. When this flag is set to 1, BIOS deallocates any arbitration level that is currently allocated to the device. When this flag is set to 0, BIOS does not deallocate any arbitration level that is currently allocated to the device.
- The Number of Characters Printed field is valid only when this function is used to cancel an outstanding Print Block function (hex

09) request and when the function is called with the same request block that was used to begin the Print Block function request.

- The Printer Status field is valid only when this function is completed. The status that is returned in the request block is not valid during intermediate stages.
- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Byte	16H	Request flags
		Bit 7 - Arbitration-deallocation flag
		= 0 - Do not deallocate
		= 1 - Deallocate
		Bit 6 - Arbitration allocation flag
		= 0 - Do not assume that the arbitration level is deallocated by the operating system
		= 1 - Assume that the arbitration level is deallocated by the operating system
		Bits 5 to 0 - Reserved
Byte	17H	Reserved

### Service-Specific Output

Size	Offset	Description
Word	26H	Number of characters printed
Byte	26H	Printer status
		Bit 7 = 1 - Busy
		Bit 6 = 1 - Acknowledge
		Bit 5 = 1 - End of paper
		Bit 4 = 1 - Selected
		Bit 3 = 1 - I/O Error
		Bit 2 = 1 - Interrupt (non-DMA mode only)
		Bits 1, 0 - Reserved

### 0CH—Return Printer Status

- This function returns the printer status.
- The Printer Status field is valid only when the function is successfully completed.
- The possible values of the Return Code field are hex 0000 and 8000.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved



**Service-Specific Output**

Size	Offset	Description
Byte	28H	Printer status
		Bit 7 = 1 - Busy
		Bit 6 = 1 - Acknowledge
		Bit 5 = 1 - End of paper
		Bit 4 = 1 - Selected
		Bit 3 = 1 - I/O error
		Bit 2 = 1 - Interrupt (non-DMA mode only)
		Bits 1, 0 - Reserved

## Return Codes

Return codes are returned at offset hex 0C.

Value	Description
0000H	Operation Successfully Completed
0001H	Stage on Interrupt
0002H	Stage on Time
0005H	Not My Interrupt, Stage on Interrupt
8000H	Device in Use
8001H	Device Busy
8008H	DMA Not Supported
8081H	Arbitration Level Not Available
8082H	Arbitration Level Not Allocated
8083H	Arbitration Level Disabled
8084H	Transfer in Progress
8085H	No Transfer in Progress
8086H	No DMA Channel Available
8087H	Arbitration Level Not Disabled
9000H	Printer Error
C000H	Invalid Logical ID (ABIOS transfer convention only)
C001H	Invalid Function Number
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length
C005H	Invalid Time-Out
C006H	Invalid Time to Wait
C007H	Invalid Data-Pointer Values

*Figure 6-12. Parallel Port Return Codes*

## Programming Considerations

- The Return Logical ID Parameters function (hex 01) indicates that data pointer 1 and data pointer 2 are expected by the Print Block function (hex 09).
- The Print Block function (hex 09) does not recognize 0 as a valid data pointer. If both data pointer 1 and data pointer 2 are set to 0, the request is denied, and ABIOS sets the Return Code field to hex C007 (Invalid Data-Pointer Values). The following table shows how the Print Block function behaves under all possible combinations of hardware support and data-pointer fields in the request block.

<b>DMA Support Hardware</b>	<b>Nonzero Data Pointer 1</b>	<b>Nonzero Data Pointer 2</b>	<b>Performed Action</b>
N	N	N	Error (C007H)
N	N	Y	Error (8008H)
N	Y	N	PIO
N	Y	Y	PIO
Y	N	N	Error (C007H)
Y	N	Y	DMA*
Y	Y	N	PIO
Y	Y	Y	DMA**

\* Indicates that if there are no DMA channels or the arbitration level is not available, the request is denied with an error condition. There is no default to programmed I/O (PIO).

\*\* Indicates that if there are no DMA channels or the arbitration level is not available, the request is performed using programmed I/O (PIO).

**Figure 6-13. Print Block Functions**

- If the Printer Status field is busy when a Print Block function (hex 09) is requested, BIOS checks the printer status for a specified period of time, and the function is terminated if the device is still busy after that time. BIOS sets the Return Code field to hex 8001 (Device Busy). The Number of Characters Printed field indicates the number of characters that were printed.
- If the Print Block function (hex 09) is in programmed I/O mode and the printer is put offline in the middle of a print block, BIOS checks the printer status for a specified period of time, and the function is terminated if the device is still busy after that time. BIOS sets the Return Code field to hex 8001 (Device Busy). The caller can issue a new Print Block function to print the remaining characters when the printer is put back online.
- If the Print Block function is in DMA mode and there are any transitions in the printer lines (such as "offline," "out of paper," or "error"), BIOS terminates the request. An appropriate return code (hex 8000, 8001, or 9000) and the number of characters that were printed are returned in the request block. The caller can issue a new Print Block function to print the remaining characters when the condition has been corrected.
- When the Reset/Initialize function (hex 05) is called, some printers perform a self-test that causes the printer to be busy until the self-test is completed. The busy bit (bit 7) of the Printer Status field indicates this busy condition.
- Parallel port BIOS supports the parallel port in transmit mode only.

## Notes:

## Device ID 06H—Asynchronous Communication

### Functions

The following are the asynchronous communication functions for the serial port. The Default Interrupt Handler function (hex 00) and the Return Logical ID Parameters function (hex 01) are described in "Request Block" in the "Transfer Conventions" section.

**Note:** All reserved input fields must be set to 0.

#### 00H—Default Interrupt Handler

#### 01H—Return Logical ID Parameters

#### 02H—Reserved

#### 03H—Read Device Parameters

- This function returns the serial-port information. It has no effect on any other outstanding request, and it does not interact with the hardware.
- The parameters are maintained in the device block as a shadow of the hardware. The shadow is updated when requests are made through BIOS. If the serial ports are programmed directly or through the BIOS Interrupt 14H functions, the asynchronous parameters in the device block will be incorrect. In this case, the Read Device Parameters function returns the previously-stored values.
- To synchronize these parameters, call the Reset/Initialize function (hex 05).
- During BIOS initialization, the serial port is initialized to the following default parameters:
  - Baud rate of 1200
  - No parity
  - One stop bit
  - Seven bits per character
  - No break.
- The three Compare Character Match fields enable programming of receive-match characters in receive operations.
- If a character that is received from the controller matches the value in one of the three enabled Compare Character Match

fields, an action is performed as defined by the three Compare Character Function fields.

- The data in a Compare Character Match field is an 8-bit match character. If the word length is less than 8 bits, the match character must be right justified, and any unused bits must be set to 0.
- If null stripping is enabled, the Compare Character Match 3 field is unavailable.
- The three Compare Character Function fields correspond to the three Compare Character Match fields. They enable programming of the controller to start the transmitter, stop the transmitter, delete a match character from an incoming data stream, or to interrupt.
- Multiple Compare Character Match functions per Compare Character function are supported.
- If bit 3 (start transmitter) and bit 2 (stop transmitter) of the Compare Character Function field are both set to 1, bit 2 takes precedence.
- If bit 0 (interrupt) is set to 1, the Interrupt occurs as soon as a match with a received data character is detected.
- To disable a Compare Character Match field, set the corresponding Compare Character Function field to 0 before calling the Set Compare Character Match and Function function (hex 1B).
- The Modem Control field returns the active modem-control functions. The Enhanced Function Control field returns additional modem-control functions.
- The baud rate is calculated by the following formula:  
$$\text{Baud clock/Scaler/Baud rate} = \text{Divisor count}$$
- The baud clock for the type-3 controller has two internal frequencies: 22.1184 MHz with a scaler of 32, and 1.8432 MHz with a scaler of 16. The baud clock for the type-1 and type-2 controllers has an internal frequency of 1.8432 MHz with a scaler of 16. The frequency is divided internally by a scaler to increase the sample time per character bit.

- The Binary Baud Rate field contains the values that are used to calculate the baud rate when the Communication Baud Rate field is set to hex 0FF. The Binary Baud Rate field consists of a 24-bit value plus an 8-bit fraction that is described in the request block. The divisor count is a 16-bit value that is loaded into the divisor latch of the serial device to generate the final baud-rate clock. Any fractional amount is rounded to the nearest divisor count to get the closest baud rate. This integer is put into the following formula to get the values that are returned as output in the request block:

$$\text{Baud clock/Scaler/Divisor count} = \text{Output baud rate}$$

- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Word	18H	Reserved

### Service-Specific Output

Size	Offset	Description
Byte	10H	Logical ID interrupt level
Word	18H	Enhanced function status
		Bit 15 - Binary baud rate support
		= 0 - Binary baud functions are not supported
		= 1 - Binary baud functions are supported
		Bits 14 to 9 - Reserved
		Bit 8 - 'High data rate' signal status
		= 0 - 'High data rate' signal is inactive
		= 1 - 'High data rate' signal is active
		Bit 7 - Transmit byte pacing
		= 0 - Not enabled
		= 1 - Enabled
		Bit 6 - High-frequency rate
		= 0 - High-frequency rate not enabled (1.8432 MHz)
		= 1 - High-frequency rate enabled (22.1184 MHz)
		Bit 5 - Slow transmit rate (not available when transmit byte pacing is used)
		= 0 - Transmit at baud rate
		= 1 - Transmit at 1/16 baud rate
		Bit 4 - Slow receive rate (not available when transmit byte pacing is used)
		= 0 - Receive at baud rate
		= 1 - Receive at 1/16 baud rate

Size	Offset	Description
		Bit 3 - Control receiver via data-set-ready (DSR) = 0 - No receiver control via DSR = 1 - If DSR = 0, the control receiver is turned off; if DSR = 1, the control receiver is turned on; if DSR drops, the character currently being received is discarded.
		Bit 2 - Control transmitter via data-carrier-detect (DCD) = 0 - No transmitter control via DCD = 1 - If DCD = 0, the control transmitter is turned off; if DCD = 1, the control transmitter is turned on; the transmitter must have initially been turned on by a Start Sending command.
		Bit 1 - Control transmitter via DSR = 0 - No transmitter control via DSR = 1 - If DSR = 0, the control transmitter is turned off; if DSR = 1, the control transmitter is turned on; the transmitter must have initially been turned on by a Start Sending command.
		Bit 0 - Control transmitter via clear-to-send (CTS) = 0 - No transmitter control via CTS = 1 - If CTS = 0, the control transmitter is turned off; if CTS = 1, the control transmitter is turned on; the transmitter must have initially been turned on by a Start Sending command.
Byte	1AH	Compare character match 1
Byte	1BH	Compare character function 1 Bits 7 to 4 - Reserved Bit 3 - Start transmitter on a character match = 0 - Transmitter not started on a character match = 1 - Transmitter started on a character match (the transmitter will not start if bit 2 is set to 1) Bit 2 - Stop transmitter on a character match = 0 - Transmitter not stopped on a character match = 1 - Transmitter stopped on a character match (if the transmitter is stopped with a character match, a call to the Start Transmission function (hex 18) can be used to restart the transmitter) Bit 1 - Delete character on a character match = 0 - Character not deleted on a character match = 1 - Character deleted on a character match Bit 0 - Interrupt on character match = 0 - Interrupt not enabled on a character match = 1 - Interrupt enabled on a character match (the interrupt occurs as soon as there is a match with a received data character)
Byte	1CH	Compare character match 2
Byte	1DH	Compare character function 2 (see the description of the Compare Character Function 1 field)
Byte	1EH	Compare character match 3
Byte	1FH	Compare character function 3 (see the description of the Compare Character Function 1 field)



Size	Offset	Description
Byte	28H	<b>Modem control</b> Bits 7 to 2 - Reserved Bit 1 - 'Request to send' (RTS) signal status = 0 - RTS signal is disabled = 1 - RTS signal is enabled Bit 0 - 'Data terminal ready' (DTR) signal status = 0 - DTR signal is disabled = 1 - DTR signal is enabled
Byte	29H	<b>Asynchronous interrupt status word</b> Bit 7 - DMA receive mode = 0 - Disabled = 1 - Enabled Bit 6 - DMA transmit mode = 0 - Disabled = 1 - Enabled Bit 5 - Modem status interrupt = 0 - Disabled = 1 - Enabled Bit 4 - Receive-line status interrupt = 0 - Disabled = 1 - Enabled Bit 3 - Transmit interrupt = 0 - Disabled = 1 - Enabled Bit 2 - Receive interrupt = 0 - Disabled = 1 - Enabled Bit 1 - Receiver character count = 0 - Disabled = 1 - Enabled Bit 0 - 'High data rate' signal selector interrupt = 0 - Disabled = 1 - Enabled
Byte	2AH	<b>Receive trigger-level status</b> = 00H - 1 byte = 01H - 4 bytes = 02H - 8 bytes = 03H - 14 bytes = 04H to FFH - Reserved
Byte	2BH	<b>FIFO-mode status</b> Bit 7 - Dynamic arbitration allocation = 0 - Arbitration levels are fixed = 1 - Arbitration levels are programmable Bits 6 to 4 - Reserved Bit 3 - FIFO support available in ABIOS = 0 - No ABIOS support for FIFO = 1 - ABIOS supports FIFO Bit 2 - DMA transfer available = 0 - No DMA transfer capability = 1 - Device capable of data transfer via DMA Bit 1 - Serial device has FIFO ability = 0 - FIFO not available in serial device = 1 - FIFO available in serial device Bit 0 - Transmit and receive FIFO mode status = 0 - Serial device operates in character mode = 1 - Serial device operates in FIFO mode

Size	Offset	Description
Byte	40H	8-bit fraction of binary baud rate (if offset hex 44 is set to hex FF)
Byte	41H	Least-significant byte of binary baud rate (if offset hex 44 is set to hex FF)
Byte	42H	Next-significant byte of binary baud rate (if offset hex 44 is set to hex FF)
Byte	43H	Most-significant byte of binary baud rate (if offset hex 44 is set to hex FF)
Byte	44H	Communication baud rate = 00H - 110 = 01H - 150 = 02H - 300 = 03H - 600 = 04H - 1200 = 05H - 2400 = 06H - 4800 = 07H - 9600 = 08H - 19200 = 09H - 31418.2 = 0AH - 38400 = 0BH - 57600 = 0CH - 76800 = 0DH - 115200 = 0EH - 138240 = 0FH - 172800 = 10H - 230400 = 11H - 345600 = 12H to FEH - Reserved = FFH - Calculate baud rate from variable at offset hex 40
Byte	45H	Type of parity = 00H - None = 01H - Odd = 02H - Even = 03H - Stick parity odd = 04H - Stick parity even = 05H to FFH - Reserved
Byte	46H	Stop bit = 00H - The stop-bit length is 1 bit = 01H - If the data-bit length is 6, 7, or 8 bits, the stop-bit length is 2 bits; If the data-bit length is 5 bits, the stop-bit length is 1½ bits. = 02H to FFH - Reserved
Byte	47H	Data-bit length = 00H - 5 bits = 01H - 6 bits = 02H - 7 bits = 03H - 8 bits = 04H to 0FFH - Reserved
Byte	48H	Break status = 00H - Disabled = 01H - Enabled = 02H to FFH - Reserved
Byte	49H	Reserved
Byte	4AH	Reserved

## 04H—Reserved

## 05H—Reset/Initialize

- This function initializes the serial port according to the input parameters.
- All communication interrupts (receive, transmit, and modem status) are disabled. The caller is responsible for clearing all outstanding request blocks and the interrupt controller, where appropriate. From the BIOS standpoint, all outstanding request blocks are canceled.
- Any received data that is pending at the serial port is cleared.
- The Reset/Initialize function is used to synchronize the device-block parameters with the current Hardware port values in preparation for a Read Device Parameters function (hex 03) call.
- The Modem Control field enables the setting of modem-control functions. The Enhanced Function Control field enables the setting of additional modem-control functions.
- The baud rate is calculated by the following formula:

$$\text{Baud clock/Scaler/Baud rate} = \text{Divisor count}$$

- The baud clock for the type-3 controller has two internal frequencies: 22.1184 MHz with a scaler of 32, and 1.8432 MHz with a scaler of 16. The baud clock for the type-1 and type-2 controllers has an internal frequency of 1.8432 MHz with a scaler of 16. The frequency is divided internally by a scaler to increase the sample time per character bit.
- The Binary Baud Rate field contains the values that are used to calculate the baud rate when the Communication Baud Rate field is set to hex 0FF. The Binary Baud Rate field consists of a 24-bit value plus an 8-bit fraction that is described in the request block. The divisor count is a 16-bit value that is loaded into the divisor latch of the serial device to generate the final baud-rate clock. Any fractional amount is rounded to the nearest divisor count to get the closest baud rate. This integer is put into the following formula to get the values that are returned as output in the request block:

$$\text{Baud clock/Scaler/Divisor count} = \text{Output baud rate}$$

- The caller is responsible for determining baud-rate tolerances. To determine the maximum or minimum baud rate of the serial device, enter hex 0FFFFFFF (for the maximum) or hex 0 (for the minimum) into the input parameters. On output, the Binary Baud Rate fields contain the limit for the parameter that was entered.

- For type-3 controllers, baud rates above 19200 bits per second can be selected only from the Communication Baud Rate field (at offset hex 44).
- The FIFO trigger level is programmed only if the FIFO mode is enabled.
- The value of the Return Code field is hex 0000.

## Service-Specific Input

Size Word	Offset 18H	Description
		Enhanced function control
		Bits 15 to 6 - Reserved
		Bit 5 - Slow transmit rate (not available when transmit byte pacing is used)
		= 0 - Transmit at baud rate
		= 1 - Transmit at 1/16 baud rate
		Bit 4 - Slow receive rate (not available when transmit byte pacing is used)
		= 0 - Receive at baud rate
		= 1 - Receive at 1/16 baud rate
		Bit 3 - Control receiver via DSR
		= 0 - No receiver control via DSR
		= 1 - If DSR = 0, the control receiver is turned off; if DSR = 1, the control receiver is turned on; if DSR drops, the character currently being received is discarded.
		Bit 2 - Control transmitter via DCD
		= 0 - No transmitter control via DCD
		= 1 - If DCD = 0, the control transmitter is turned off; if DCD = 1, the control transmitter is turned on; the transmitter must have initially been turned on by a Start Sending command.
		Bit 1 - Control transmitter via DSR
		= 0 - No transmitter control via DSR
		= 1 - If DSR = 0, the control transmitter is turned off; if DSR = 1, the control transmitter is turned on; the transmitter must have initially been turned on by a Start Sending command.
		Bit 0 - Control transmitter via CTS
		= 0 - No transmitter control via CTS
		= 1 - If CTS = 0, the control transmitter is turned off; if CTS = 1, the control transmitter is turned on; the transmitter must have initially been turned on by a Start Sending command.
Byte	28H	Modem control
		Bits 7 to 2 - Reserved
		Bit 1 - RTS signal status
		= 0 - RTS signal is disabled
		= 1 - RTS signal is enabled
		Bit 0 - DTR signal status
		= 0 - DTR signal is disabled
		= 1 - DTR signal is enabled

<b>Size</b>	<b>Offset</b>	<b>Description</b>
<b>Byte</b>	<b>29H</b>	<b>FIFO-mode control</b> = 00H - Disable = 01H - Enable and reset FIFO = 02H - Enable without resetting FIFO = 03H to FFH - Reserved
<b>Byte</b>	<b>2AH</b>	<b>Receive trigger level</b> = 00H - 1 byte = 01H - 4 bytes = 02H - 8 bytes = 03H - 14 bytes = 04H to FFH - Reserved
<b>Byte</b>	<b>40H</b>	<b>8-bit fraction of binary baud rate</b> (if offset hex 44 is set to hex FF)
<b>Byte</b>	<b>41H</b>	<b>Least-significant byte of binary baud rate</b> (if offset hex 44 is set to hex FF)
<b>Byte</b>	<b>42H</b>	<b>Next-significant byte of binary baud rate</b> (if offset hex 44 is set to hex FF)
<b>Byte</b>	<b>43H</b>	<b>Most-significant byte of binary baud rate</b> (if offset hex 44 is set to hex FF)
<b>Byte</b>	<b>44H</b>	<b>Communication baud rate</b> = 00H - 110 = 01H - 150 = 02H - 300 = 03H - 600 = 04H - 1200 = 05H - 2400 = 06H - 4800 = 07H - 9600 = 08H - 19200 = 09H - 31418.2 = 0AH - 38400 = 0BH - 57600 = 0CH - 76800 = 0DH - 115200 = 0EH - 138240 = 0FH - 172800 = 10H - 230400 = 11H - 345600 = 12H to FEH - Reserved = FFH - Calculate baud rate from variable at offset hex 40
<b>Byte</b>	<b>45H</b>	<b>Type of parity</b> = 00H - None = 01H - Odd = 02H - Even = 03H - Stick parity odd = 04H - Stick parity even = 05H to FFH - Reserved
<b>Byte</b>	<b>46H</b>	<b>Stop bit</b> = 00H - The stop-bit length is 1 bit = 01H - If the data-bit length is 6, 7, or 8 bits, the stop-bit length is 2 bits; if the data-bit length is 5 bits, the stop-bit length is 1½ bits. = 02H to FFH - Reserved

Size	Offset	Description
Byte	47H	Data-bit length = 00H - 5 bits = 01H - 6 bits = 02H - 7 bits = 03H - 8 bits = 04H to 0FFH - Reserved
Byte	48H	Break status = 00H - Disable = 01H - Enable = 02H to FFH - Reserved

## Service-Specific Output

Size	Offset	Description
Byte	40H	8-bit fraction of binary baud rate (if offset hex 44 is set to hex FF)
Byte	41H	Least-significant byte of binary baud rate (if offset hex 44 is set to hex FF)
Byte	42H	Next-significant byte of binary baud rate (if offset hex 44 is set to hex FF)
Byte	43H	Most-significant byte of binary baud rate (if offset hex 44 is set to hex FF)
Byte	49H	Line status Bit 7 - Error in receiver FIFO = 0 - No error in FIFO = 1 - Error in receiver FIFO Bit 6 - Transmitter empty = 0 - Inactive = 1 - Active Bit 5 - Transmitter holding register empty = 0 - Inactive = 1 - Active Bit 4 - Break interrupt = 0 - Inactive = 1 - Active Bit 3 - Framing error = 0 - Inactive = 1 - Active Bit 2 - Parity error = 0 - Inactive = 1 - Active Bit 1 - Overrun error = 0 - Inactive = 1 - Active Bit 0 - Data ready = 0 - Inactive = 1 - Active

Size	Offset	Description
Byte	4AH	Modem status
		Bit 7 - Data-carrier detect
		= 0 - Inactive
		= 1 - Active
		Bit 6 - Ring indicator
		= 0 - Inactive
		= 1 - Active
		Bit 5 - Data set ready
		= 0 - Inactive
		= 1 - Active
		Bit 4 - Clear to send
		= 0 - Inactive
		= 1 - Active
		Bit 3 - Delta data-carrier detect
		= 0 - Inactive
		= 1 - Active
		Bit 2 - Trailing-edge ring indicator
		= 0 - Inactive
		= 1 - Active
		Bit 1 - Delta data set ready
		= 0 - Inactive
		= 1 - Active
		Bit 0 - Delta clear to send
		= 0 - Inactive
		= 1 - Active

**06H—Enable (Reserved)**

**07H—Disable (Reserved)**

**08H—Read (Reserved)**

**09H—Write (Reserved)**

**0AH—Additional Data Transfer (Reserved)**

**0BH—Set Modem Control**

- This function sets the modem control according to the input parameters. This function does not affect the interrupt state of any other stage-on-interrupt requests.
- The Modem Control field enables the setting of modem-control functions. The Enhanced Function Control field enables the setting of additional modem-control functions.
- The value of the Return Code field is hex 0000.

## Service-Specific Input

Size	Offset	Description
Word	18H	<p>Enhanced function control</p> <p>Bits 15 to 4 - Reserved</p> <p>Bit 3 - Control receiver via DSR</p> <p>= 0 - No receiver control via DSR</p> <p>= 1 - If DSR=0, the control receiver is turned off; if DSR=1, the control receiver is turned on; If DSR drops, the character currently being received is discarded.</p> <p>Bit 2 - Control transmitter via DCD</p> <p>= 0 - No transmitter control via DCD</p> <p>= 1 - If DCD=0, the control transmitter is turned off; if DCD=1, the control transmitter is turned on; the transmitter must have initially been turned on by a Start Sending command.</p> <p>Bit 1 - Control transmitter via DSR</p> <p>= 0 - No transmitter control via DSR</p> <p>= 1 - If DSR=0, the control transmitter is turned off; if DSR=1, the control transmitter is turned on; the transmitter must have initially been turned on by a Start Sending command.</p> <p>Bit 0 - Control transmitter via CTS</p> <p>= 0 - No transmitter control via CTS</p> <p>= 1 - If CTS=0, the control transmitter is turned off; if CTS=1, the control transmitter is turned on; the transmitter must have initially been turned on by a Start Sending command.</p>
Byte	28H	<p>Modem control</p> <p>Bits 7 to 2 - Reserved</p> <p>Bit 1 - RTS signal status</p> <p>= 0 - RTS signal is disabled</p> <p>= 1 - RTS signal is enabled</p> <p>Bit 0 - DTR signal status</p> <p>= 0 - DTR signal is disabled</p> <p>= 1 - DTR signal is enabled</p>

## Service-Specific Output

Size	Offset	Description
None		

### 0CH—Set Line Control

- This function sets the line control according to the input parameters. It does not affect the interrupt state of any other stage-on-interrupt requests.
- The value of the Return Code field is hex 0000.



**Service-Specific Input**

Size	Offset	Description
Word	18H	Reserved
Byte	45H	Type of parity = 00H - None = 01H - Odd = 02H - Even = 03H - Stick parity odd = 04H - Stick parity even = 05H to FFH - Reserved
Byte	46H	Stop bit = 00H - The stop-bit length is 1 bit = 01H - If the data-bit length is 6, 7, or 8 bits, the stop-bit length is 2 bits; If the data-bit length is 5 bits, the stop-bit length is 1½ bits. = 02H to FFH - Reserved
Byte	47H	Data-bit length = 00H - 5 bits = 01H - 6 bits = 02H - 7 bits = 03H - 8 bits = 04H to 0FFH - Reserved
Byte	48H	Break status = 00H - Disabled = 01H - Enabled = 02H to FFH - Reserved

**Service-Specific Output**

Size	Offset	Description
None		

**0DH—Set Baud Rate**

- This function sets the baud rate according to the input parameter. It does not affect the interrupt state of any other stage-on-interrupt requests.
- The baud rate is calculated by the following formula:  
$$\text{Baud clock/Scaler/Baud rate} = \text{Divisor count}$$
- The baud clock for the type-3 controller has two internal frequencies: 22.1184 MHz with a scaler of 32, and 1.8432 MHz with a scaler of 16. The baud clock for the type-1 and type-2 controllers has an internal frequency of 1.8432 MHz with a scaler of 16. The frequency is divided internally by a scaler to increase the sample time per character bit.

- The Binary Baud Rate field contains the values that are used to calculate the baud rate when the Communication Baud Rate field is set to hex 0FF. The Binary Baud Rate field consists of a 24-bit value plus an 8-bit fraction that is described in the request block. The divisor count is a 16-bit value that is loaded into the divisor latch of the serial device to generate the final baud-rate clock. Any fractional amount is rounded to the nearest divisor count to get the closest baud rate. This integer is put into the following formula to get the values that are returned as output in the request block:

$$\text{Baud clock/Scaler/Divisor count} = \text{Output baud rate}$$

- The caller is responsible for determining baud-rate tolerances. To determine the maximum or minimum baud rate of the serial device, enter hex 0FFFFFFF (for the maximum) or hex 0 (for the minimum) into the input parameters. On output, the Binary Baud Rate fields contain the limit for the parameter that was entered.
- For type-3 controllers, baud rates above 19200 bits per second can be selected only from the Communication Baud Rate field (at offset hex 44).
- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Word	18H	Enhanced function control
		Bits 15 to 6 - Reserved
		Bit 5 - Slow transmit rate (not available when transmit byte pacing is used)
		= 0 - Transmit at baud rate
		= 1 - Transmit at 1/16 baud rate
		Bit 4 - Slow receive rate (not available when transmit byte pacing is used)
		= 0 - Receive at baud rate
		= 1 - Receive at 1/16 baud rate
		Bits 3 to 0 - Reserved
Byte	40H	8-bit fraction of binary baud rate (if offset hex 44 is set to hex FF)
Byte	41H	Least-significant byte of binary baud rate (if offset hex 44 is set to hex FF)
Byte	42H	Next-significant byte of binary baud rate (if offset hex 44 is set to hex FF)
Byte	43H	Most-significant byte of binary baud rate (if offset hex 44 is set to hex FF)

Size	Offset	Description
Byte	44H	Communication baud rate
		= 00H - 110
		= 01H - 150
		= 02H - 300
		= 03H - 600
		= 04H - 1200
		= 05H - 2400
		= 06H - 4800
		= 07H - 9600
		= 08H - 19200
		= 09H - 31418.2
		= 0AH - 38400
		= 0BH - 57600
		= 0CH - 76800
		= 0DH - 115200
		= 0EH - 138240
		= 0FH - 172800
		= 10H - 230400
		= 11H - 345600
		= 12H to FEH - Reserved
		= FFH - Calculate baud rate from variable at offset hex 40

### Service-Specific Output

Size	Offset	Description
Byte	40H	8-bit fraction of binary baud rate (if offset hex 44 is set to hex FF)
Byte	41H	Least-significant byte of binary baud rate (if offset hex 44 is set to hex FF)
Byte	42H	Next-significant byte of binary baud rate (if offset hex 44 is set to hex FF)
Byte	43H	Most-significant byte of binary baud rate (if offset hex 44 is set to hex FF)
Byte	44H	Communication baud rate (if hex FF is selected, the value in offset hex 40 is returned; otherwise, no values are changed)

### 0EH—Transmit Interrupt

- This function provides several modes for transferring data to the transmitter in a serial port.
- Data pointers that describe a buffer area are passed in a request block to ABIOs. The data in the buffer is loaded into the transmitter in an asynchronous device. Each time the transmitter is emptied, it is reloaded. For a system that supports DMA mode, several options are available for reloading the transmitter. Output parameters are returned to inform the program of transmit-operation progress.
- There are two modes for transmitting data: programmed input/output (PIO) mode and direct memory access (DMA) mode.

- A variety of transmit options are available to support PIO mode and DMA mode. These options are controlled through bit settings in the Enhanced Mode field (offset hex 18) and the Transmit Control field (offset hex 44). When a DMA-transmit operation is requested, options can be selected to assist the program in completing the data transfer. If DMA mode is selected but is not available, an option is available to default to PIO mode or to return an error. Depending on the mode that is selected, logical pointers, physical pointers, or both are required. An option is also available to deallocate the DMA arbitration level and channel when the data transfer is complete.
- The caller must initialize the Operation Status field to 0 before calling the Start routine. None of the bits in the Operation Status field takes precedence over any other bits in this field.
- If the value of the Number of Bits Per Character field (offset hex 47) is less than 8, 8 bits are sent to the serial port, but only the number of bits that is specified in the Number of Bits Per Character field are transmitted.
- The Additional Operation Status field is defined only in systems that support DMA mode. Therefore, do not use the Additional Operation Status field if bit 2 of the FIFO-Mode Status field (offset hex 2B on output) is set to 0.
- The possible values of the Return Code field are hex 0000, 0001, 0005, 0081, 8000, 8001, 8081, 8082, 8083, 8084, 8085, 8086, 8087, and 9000.

### **Programmed Input/Output (PIO) Transmit**

- Programmed input/output is a technique in which serial ports transfer data through interrupts.
- Each time the transmitter is emptied, an interrupt is generated, causing a branching to the Transmit Interrupt function (hex 0E). The Transmit Interrupt function fills the transmitter, using the processing-unit I/O instructions. This process is repeated until all the data in the buffer has been transmitted.
- In early serial ports, the transmitter holds only one character at a time. In more advanced serial ports, the transmitter holds a 16-character, first-in, first-out (FIFO) buffer, allowing more characters to be sent from the data buffer to the transmitter each time an interrupt occurs.
- When bit 0 of the Enhanced Mode field is set to 0, PIO mode is enabled.

- **ABIOS enables the Transmit interrupt but does not transmit any data until a Transmit interrupt occurs.**
- **The Transmit Tail Pointer field points to the first byte of data that is to be transmitted. The Transmit Head Pointer field points to one byte logically beyond the last byte that is to be transmitted.**
- **The values of the Transmit Head Pointer field and the Transmit Tail Pointer field are relative to the beginning of the transmit buffer, where a value of 0 indicates the first physical byte of the buffer and the value of the Transmit-Buffer Length field minus 1 indicates the last physical byte of the buffer. The values of the Transmit Head Pointer field and the Transmit Tail Pointer field must never be out of that range.**
- **The maximum number of characters that the transmit buffer can indicate to be sent at any time is the value of the Transmit-Buffer Length field minus 1.**
- **When a Transmit interrupt occurs, it increases the value in the Transmit Tail Pointer field by the number of bytes that were transmitted. Bit 1 of the Operation Status field (offset hex 4B on output) is set to 1. Because of the possibility that the buffer will be checked asynchronously by the Transmit interrupt, the data is written to the transmit buffer before the value of the Transmit Tail Pointer field is increased.**
- **As interrupts occur, the value of the Transmit Tail Pointer field approaches the value of the Transmit Head Pointer field. If the value of the Transmit Tail Pointer field reaches the end of the transmit buffer, the Transmit Tail Pointer field is set to 0.**
- **A transmit-buffer-empty condition occurs when the value of the Transmit Tail Pointer field equals the value of the Transmit Head Pointer field. During processing of a Transmit interrupt, if a transmit-buffer-empty condition occurs after data is sent to the serial port, ABIOS stops sending data to the serial port and informs the caller of the condition. Bit 6 of the Operation Status field is set to 1, indicating that the transmit buffer is empty but the Transmit interrupt is still enabled. On any subsequent Transmit interrupt, if the transmit-buffer-empty condition exists, ABIOS disables the Transmit interrupt, the request block is considered to be complete, and bit 7 of the Operation Status field is set to 1. Bits 6 and 7 of the Operation Status field are mutually and exclusively set.**
- **A transmit-buffer-full condition occurs when the value of the Transmit Head Pointer field equals the value of the Transmit Tail Pointer field minus 1. A transmit-buffer-full condition occurs also when the value of the Transmit Tail Pointer field is 0 and the**

value of the Transmit Head Pointer field equals the value of the Transmit-Buffer Length field minus 1.

- The values of the Transmit-Buffer Segment field, the Transmit-Buffer Offset field, and the Transmit-Buffer Length field can be altered during calls to the Transmit interrupt. Because BIOS removes the data from the transmit buffer before changing the value of the Transmit Tail Pointer field, the caller can put additional data into the buffer and logically increase the value of the Transmit Head Pointer field during processing of a Transmit interrupt. The caller must not allow the value of the Transmit Head Pointer field to equal the value of the Transmit Tail Pointer field, because this indicates a transmit-buffer-empty condition.
- The following fields must be initialized and passed for a PIO-transmit operation:
  - Transmit-Buffer Offset field
  - Transmit-Buffer Segment field
  - Enhanced Mode field
  - Transmit-Buffer Length field
  - Transmit-Buffer Head field
  - Transmit-Buffer Tail field
  - Transmit Byte Pacing Rate field (if bit 2 of the Enhanced Mode field is set to 1).
- Transmit byte pacing is available only in PIO mode.
- The transmit-byte-pacing bit (bit 2 of the Enhanced Mode field) enables pacing of the transmitter. If this bit is set to 1, the serial port is programmed to wait a specified length of time before sending the next character. The delay is specified in the Transmit Byte Pacing Rate field. The value in this field represents the number of 16-bit packets that are contained within the delay between characters. Therefore, the delay time (in seconds) between characters is calculated as follows:

$$\frac{\text{Transmit byte pacing rate} \times 16}{\text{Baud rate}}$$

Each time the transmit-byte-pacing-rate counter decreases to 0, an interrupt occurs to enable BIOS to reload the counter.

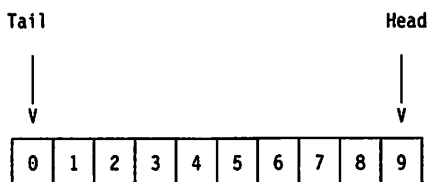
- If a transmit operation is enabled by bit 2 of the Enhanced Mode field after the Receive Interrupt function (hex 0F) is enabled by bit 3 of the Enhanced Mode field (in function hex 0F), the interrupt-on-receiver-character-count=0 option is disabled. The

functions that are associated with bit 2 of the Enhanced Mode field (in function hex 0E) and bit 3 of the Enhanced Mode field (in function hex 0F) are mutually exclusive at the serial port.

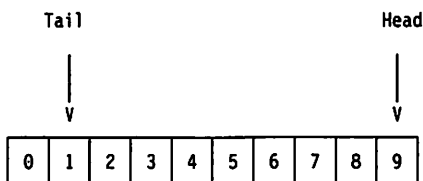
- In PIO mode, if the Transmit-Buffer Length field (offset hex 2C) is set to 0 when the Start routine is called, no action is performed, and the Return Code field is set to hex 0000 (Operation Successfully Completed).

The following is an example of a PIO-mode transmit operation:

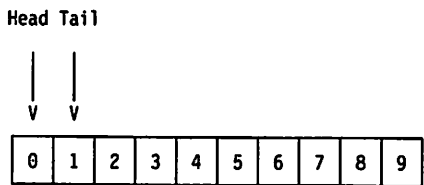
1. The caller initializes the transmit request block and calls the Start routine. The Transmit Head Pointer field and Transmit Tail Pointer field must be set within the range from 0 to the value of the Transmit-Buffer Length field, inclusive. In this example, the buffer length is set to 10, and the transmit buffer is full when the value of the Transmit Head Pointer field is 9. The maximum number of characters that can be sent without the caller changing the value of the Transmit Head Pointer field is 9 (bytes 0 to 8).



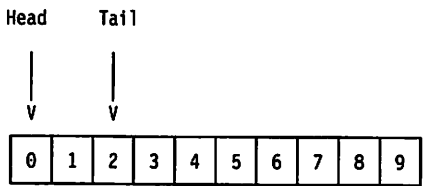
2. The Start routine enables the Transmit interrupt but does not send any data to the serial port.
3. Because the Transmitter Holding register is empty, an interrupt is generated.
4. The Transmit interrupt is called. A character is sent to the serial port, and the value of the Transmit Tail Pointer field is increased by 1. In this example, byte 0 is sent, and the Transmit Tail Pointer field is set to 1.



5. Eight bytes are left to be sent (bytes 1 to 8). To increase the number of bytes in the buffer from 8 to 9, the caller can reset the value of the Transmit Head Pointer field to 0 (the beginning of the buffer).

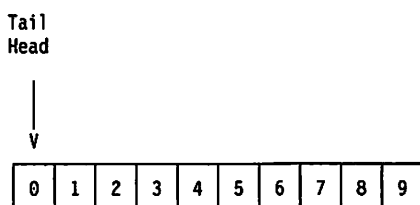


6. Again, when the Transmitter Holding register is empty, an interrupt is generated to the processor. The Interrupt routine is called, and a character is sent to the serial port. The value of the Transmit Tail Pointer field is increased by 1. In this example, byte 1 is sent, and the Transmit Tail Pointer field is set to 2.





7. Assuming that the caller does not change the value of the Transmit Head Pointer field, the process repeats as interrupts occur. Byte 2 is sent, and the Transmit Tail Pointer field is set to 3, and so forth, until byte 9 is sent and the Transmit Tail Pointer field is reset to 0. At this point, the value of the Transmit Tail Pointer field equals the value of the Transmit Head Pointer field, and the caller is informed of the transmit-buffer-empty condition (by the Operation Status field). If the transmit-buffer-empty condition still exists when the next Transmit interrupt is generated, BIOS disables the Transmit interrupt and sets the Return Code field to hex 0000 (Operation Successfully Completed).



### Direct Memory Access (DMA) Transmit

- In a system that supports DMA mode, the transmitter can be loaded through a DMA device. Each time the transmitter is emptied, the DMA device (when properly initialized) loads the transmitter by sending data directly from the buffer memory. This technique requires no processing-unit action and allows data to be sent to serial ports at higher baud rates than those allowed by programmed input/output (PIO) mode.
- To determine whether DMA mode is supported, call the Read Device Parameters function (hex 03). Bit 2 of the FIFO-Mode Status field indicates whether DMA mode is available.
- When bit 0 of the Enhanced Mode field is set to 1, DMA mode is enabled. BIOS attempts to initialize the DMA device with the values that are in the Transmit Physical Address Buffer Pointer field and the Transmit-Buffer Transfer Length field.

- The following fields must be initialized and passed for a DMA-transmit operation:
  - Transmit-Buffer Offset field (if bit 0 of the Transmit Control field is set to 1)
  - Transmit-Buffer Segment field (if bit 0 of the Transmit Control field is set to 1)
  - Enhanced Mode field
  - Transmit-Buffer Length field (if bit 0 of the Transmit Control field is set to 1)
  - Transmit Head Pointer field (if bit 0 of the Transmit Control field is set to 1)
  - Transmit Tail Pointer field (if bit 0 of the Transmit Control field is set to 1)
  - Transmit Control field (if bit 0 of the Enhanced Mode field is set to 1)
  - Transmit Byte Pacing Rate field (if bit 2 of the Enhanced Mode field is set to 1)
  - Transmit Physical Address Buffer Pointer field
  - Transmit-Buffer Transfer Length field.
- An arbitration level and a DMA channel must be allocated to the DMA device to accomplish a data transfer. The system operator assigns arbitration levels during setup. Asynchronous communication BIOS automatically allocates DMA channels before any DMA transmit operations are started. If the arbitration level and DMA channel are available, the DMA device is initialized with the parameters that are passed in the request block, and the serial port immediately begins transmitting data.
- If bit 0 of the Transmit Control field is set to 1 and either the arbitration level or the necessary DMA channel is not available, the serial port is initialized to operate in PIO mode, and any DMA-transmit operation will default to PIO mode. If this occurs, BIOS sets bit 0 of the Enhanced Mode field to 0 to indicate that the request has defaulted to PIO mode, and operation continues as if the request had been made in PIO mode. If bit 0 of the Transmit Control field is set to 0 and the arbitration level or DMA channel is not available, an error is returned, and no action is taken.
- If bit 15 of the Transmit Control field is set to 1, the arbitration level is deallocated when the data transfer is complete. If bit 15 of the Transmit Control field is set to 0, the arbitration level remains allocated to the serial port, and any subsequent transmit requests to that serial port will use that arbitration level. The Cancel function (hex 12) can be used to deallocate the arbitration level.

- If bit 14 of the Enhanced Mode field is set to 1, ABIOS assumes that another process (such as an external resource manager) will allocate and deallocate DMA arbitration-level resources, and it starts the DMA-transmit operation without allocating the arbitration level. When this bit is set to 1, ABIOS does not deallocate the arbitration level when the transmit operation is complete, regardless of whether bit 15 of the Transmit Control field is set to 1. If bit 14 of the Enhanced Mode field is set to 0, ABIOS allocates the arbitration level before starting the DMA-transmit operation.
- In DMA mode, if the Transmit-Buffer Length field (offset hex 2C) is set to 0 when the Start routine is called, a DMA channel is allocated, but no data transfers occur, and the Return Code field is set to hex 0000 (Operation Successfully Completed).

### Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
Word	12H	Transmit-buffer offset
Word	14H	Transmit-buffer segment
Word	18H	Enhanced mode
		Bit 15 - Reserved
		Bit 14 - Assume transmit arbitration level
		= 0 - ABIOS does not assume that the arbitration level is assigned
		= 1 - ABIOS assumes that the arbitration level is assigned
		Bits 13 to 3 - Reserved
		Bit 2 - Transmit byte pacing
		= 0 - Disable
		= 1 - Enable
		Bit 1 - Reserved
		Bit 0 - DMA-transmit mode
		= 0 - Disable
		= 1 - Enable
DWord	1AH	Reserved
Word	2CH	Transmit-buffer length, in bytes
Word	2EH	Reserved
Word	30H	Transmit head pointer
Word	32H	Reserved
Word	34H	Transmit tail pointer
Word	36H	Reserved

Size	Offset	Description
Word	44H	<b>Transmit control</b> Bit 15 - Transmit arbitration level deallocation = 0 - Do not deallocate arbitration level when transfer is complete = 1 - Deallocate arbitration level when transfer is complete Bits 14 to 3 - Reserved Bit 2 - Transmit search arbitration level direction = 0 - Increasing, starting with the lowest arbitration level = 1 - Decreasing, starting with the highest arbitration level Bit 1 - Transmit request arbitration level = 0 - No arbitration level was passed = 1 - Arbitration level was passed Bit 0 - Transmit default to PIO = 0 - Return error if DMA is not available = 1 - Use PIO mode if DMA is not available
Byte	4AH	Transmit arbitration level (if dynamic arbitration is supported)
Word	4BH	Reserved
Word	54H	Transmit-byte-pacing rate (maximum value = hex FF)
DWord	5EH	Transmit physical address buffer pointer
DWord	62H	Transmit-buffer transfer length

### Service-Specific Output

Size	Offset	Description
Word	16H	<b>Additional operation status</b> Bits 15 to 2 - Reserved Bit 1 - Transmit Byte Pacing interrupt = 0 - Did not occur = 1 - Occurred Bit 0 - Additional interrupts pending = 0 - No additional interrupts pending = 1 - Additional interrupts pending
Word	18H	<b>Enhanced mode</b> Bits 15 to 1 - Reserved Bit 0 - DMA-transmit mode = 0 - Disable = 1 - Enable
Word	34H	Transmit tail pointer
Word	36H	Reserved
Byte	4AH	Transmit arbitration level (if dynamic arbitration is supported)

Size	Offset	Description
Word	4BH	Operation status; return only from Interrupt routine
		Bits 15 to 8 - Reserved
		Bit 7 - Transmit buffer empty; Transmitter Holding register empty
		= 0 - Inactive
		= 1 - Active
		Bit 6 - Transmit buffer empty
		= 0 - Inactive
		= 1 - Active
		Bits 5 to 2 - Reserved
		Bit 1 - Transmit interrupt in progress
		= 0 - Inactive
		= 1 - Active
		Bit 0 - Reserved

### 0FH—Receive Interrupt

- This function provides several modes for transferring data from the receive buffer.
- Data pointers that describe a buffer area are passed in a request block to BIOS. The serial port receives characters and moves them into the buffer area. Each time the receive buffer is filled, a signal is sent to request service. Output parameters are returned to inform the program of receive-operation progress.
- There are two modes for receiving data: programmed input/output (PIO) mode and direct memory access (DMA) mode.
- A variety of receive options are available to support PIO mode and DMA mode. These options are controlled through bit settings in the Enhanced Mode field (offset hex 18) and the Receive Control field (offset hex 46). When a DMA-receive operation is requested, options can be selected to assist the program in completing the data transfer. If DMA mode is selected but is not available, an option is available to default to PIO mode or to return an error. Depending on the mode that is selected, logical pointers, physical pointers, or both are required. An option is also available to help service Receive interrupts at very high baud rates without overruns.
- The caller must initialize the Operation Status field to 0 before calling the Start routine. None of the bits in the Operation Status field takes precedence over any other bits in this field.
- When the Reset/Initialize function (hex 05) is executed, if the value of the Number of Bits Per Character field (offset hex 47) is less than 8, the high-order bits of each byte are set to 0 when data is received.

- If bit 5 (stop transmitter on any line error) of the Enhanced Mode field is set to 1, the transmitter is stopped after the shift register is emptied on any line error.
- If bit 4 (received-data status) of the Enhanced Mode field is set to 1, for every byte of data that is received, a byte of status is received in the succeeding byte in memory. The status of this bit is independent of the status of bit 1 (DMA-receive mode) of the Enhanced Mode field. If bit 0 (error/break) of the status byte is set to 1, it remains set for each character until the Reset Error/Break function (hex 19) is called. The received-data status byte has the following format:

Bit 7 - Data-carrier detect  
 Bit 6 - Clear to send  
 Bit 5 - Data set ready  
 Bit 4 - Break  
 Bit 3 - Framing  
 Bit 2 - Parity  
 Bit 1 - Overrun  
 Bit 0 - Error/break

- If bit 3 (interrupt on receiver-character-count=0) of the Enhanced Mode field is set to 1, the serial port is programmed to interrupt every time the value of the Receive Character Count field (offset hex 52) decreases to 0. When a Receive Character Count interrupt occurs, BIOS rewrites the value that is in the Receive Character Count field to the serial port. The caller can modify this value across calls to the Receive interrupt.
- Receive Character Count interrupts and Pre-Terminal Count interrupts both use the Receive Character Count field; therefore, only one of these interrupts can be selected at a time. If bit 3 of the Enhanced Mode field and bit 15 (receive pre-terminal count) of the Receive Control field are both set to 1 during a Receive Interrupt function (hex 0F) or a Combined Interrupts function (hex 10), bit 15 of the Receive Control field takes precedence.
- Receive Character Count interrupts and Pre-Terminal Count interrupts are not enabled if transmit byte pacing has been enabled by the Transmit Interrupt function (hex 0E). The functions that are associated with Bit 3 (interrupt on receiver-character-count=0) and bit 2 (transmit byte pacing) of the Enhanced Mode field are mutually exclusive at the serial port.
- The Null-Stripping Indicator field is independent of the status of bit 1 (DMA-receive mode) of the Enhanced Mode field. In DMA mode, BIOS requires the use of the Compare Character Match 3 field and the Compare Character Function 3 field. However, when null stripping and DMA mode are both enabled, the values

in these fields are ignored. If null stripping and PIO mode are both enabled, each data character of 0 that is received is not stored in the receive buffer. If a null data byte causes an overrun error, BIOS discards the null data byte. Bit 12 of the Operation Status field indicates whether this condition occurred.

- The Additional Operation Status field is defined only in systems that support DMA mode. Therefore, do not use the Additional Operation Status field if bit 2 of the FIFO-Mode Status field (offset hex 2B on output) is set to 0.
- The possible values of the Return Code field are hex 0000, 0001, 0005, 0081, 8000, 8001, 8081, 8082, 8083, 8084, 8085, 8086, 8087, and 9000.

### **Programmed Input/Output (PIO) Receive**

- Programmed input/output is a technique in which the receiver in a serial port is emptied through interrupts.
- Each time the receiver is filled, the serial port generates an interrupt, causing the processing unit to service the serial port and empty the receiver.
- In early serial ports, the receiver holds only one character at a time. In more advanced serial ports, the receiver holds 16 bytes of data, allowing more time for the processing unit to service the serial port and allowing more characters to be read each time the serial port is serviced.
- A trigger level can be specified through the Reset/Initialize function (hex 05) or the Set Receive Trigger Level function (hex 16) to cause an interrupt to be generated each time the receiver is filled to a specific level. When characters are received but do not fill the receiver to the trigger level, a Receiver Time-Out interrupt is generated if no more characters are received within four character time frames. This enables the serial port to empty the receiver when no more characters remain to be sent.
- When bit 0 of the Enhanced Mode field is set to 0, PIO mode is enabled.
- BIOS enables the Receive interrupt, and data is read from the serial port when a Receive interrupt is generated.
- The Receive Head Pointer field points to the first character position that is to be filled. The Receive Tail Pointer field points to the first received character that is to be removed by the caller.
- The values of the Receive Head Pointer field and the Receive Tail Pointer field are relative to the beginning of the receive buffer,

where a value of 0 indicates the first physical byte of the buffer, and the value of the Receive-Buffer Length field minus 1 indicates the last physical byte of the buffer.

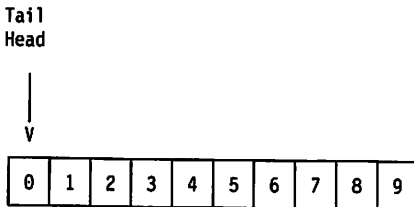
- The maximum number of characters that the receive buffer can indicate to be received at any time is the value of the Receive-Buffer Length field minus 1.
- When a Receive interrupt occurs, it increases the value in the Receive Head Pointer field by 1. Bit 0 of the Operation Status field (offset hex 4B on output) is set to 1. Because of the possibility that the buffer will be checked asynchronously by the Receive interrupt, the character is written to the receive buffer before the value of the Receive Head Pointer field is increased.
- As interrupts occur, the value of the Receive Head Pointer field approaches the value of the Receive Tail Pointer field. If the value of the Receive Head Pointer field reaches the end of the receive buffer, the Receive Head Pointer field is set to 0.
- A receive-buffer-full condition occurs when the value of the Receive Head Pointer field equals the value of the Receive Tail Pointer field minus 1. A receive-buffer-full condition occurs also when the value of the Receive Tail Pointer field is 0 and the value of the Receive Head Pointer field equals the value of the Receive Buffer Length field minus 1. Bit 4 of the Operation Status field is set to 1 to indicate a receive-buffer-full condition. If a Receive interrupt occurs while a receive-buffer-full condition exists, the current byte is lost, and bit 5 of the Operation Status field is set to 1. Bits 4 and 5 of the Operation Status field are mutually and exclusively set.
- A receive-buffer-empty condition occurs when the value of the Receive Head Pointer field equals the value of the Receive Tail Pointer field. BIOS never sets the value of the Receive Head Pointer field equal to the value of the Receive Tail Pointer field; however, the caller can set the value of the Receive Head Pointer field equal to the value of the Receive Tail Pointer field as the receive buffer is emptied and the value of the Receive Tail Pointer is logically increased.
- The values of the Receive-Buffer Segment field, the Receive-Buffer Offset field, the Receive Head Pointer field, and the Receive Tail Pointer field can be altered during calls to the Receive interrupt. Because BIOS puts data into the receive buffer before changing the value of the Receive Head Pointer field, the caller can remove received data from the receive buffer and logically increase the value of the Receive Tail Pointer field during processing of a Receive interrupt.



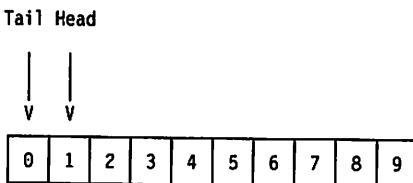
- A Receive interrupt can be terminated only by a call to the Cancel function (hex 12).
- If an error condition occurs at the serial port, ABIOS stores the current data byte in the receive buffer and returns to the caller. Bits 8 to 11 of the Operation Status field indicate the type of error that occurred. If an overrun error occurs, the overrun character is lost, and the data byte contains the valid character that caused the overrun. If a parity error occurs, the data byte contains the character that has the incorrect parity. If a framing error occurs, the data byte contains the character that does not have a valid stop bit. If a break error occurs, the data byte is set to 0.
- When Received Data Status mode is used in PIO mode, FIFO mode must be enabled.
- The following fields must be initialized and passed for a PIO-receive operation:
  - Enhanced Mode field
  - Compare Character Match 1 field (if the Compare Character Function 1 field is not set to 0)
  - Compare Character Function 1 field
  - Compare Character Match 2 field (if the Compare Character Function 2 field is not set to 0)
  - Compare Character Function 2 field
  - Compare Character Match 3 field (if the Compare Character Function 3 field is not set to 0)
  - Compare Character Function 3 field
  - Receive-Buffer Offset field
  - Receive-Buffer Segment field
  - Null-Stripping Indicator field
  - Receive-Buffer Length field
  - Receive Head Pointer field
  - Receive Tail Pointer field
  - Receive Character Count field (if bit 3 of the Enhanced Mode field is set to 1).
- In PIO mode, if the Receive-Buffer Length field (offset hex 38) is set to 0 when the Start routine is called, no action is performed, and the Return Code field is set to hex 0000 (Operation Successfully Completed).

The following is an example of a PIO-mode receive operation:

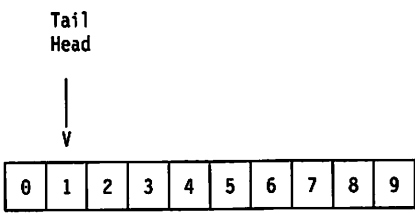
1. The caller initializes the receive request block and calls the Start routine. The Receive Head Pointer field and Receive Tail Pointer field must be set within the range from 0 to the value of the Receive Buffer Length field, inclusive. In this example, the buffer length is set to 10, and the receive buffer is full when the value of the Receive Head Pointer field is logically 1 byte less than the value of the Receive Tail Pointer field. The maximum number of characters that can be received without the caller changing the value of the Receive Tail Pointer field is 9 (bytes 0 to 8).



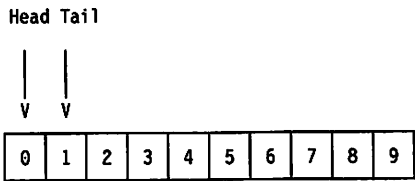
2. The Start routine enables the Receive interrupt but does not read any data.
3. When data is available at the serial port, an interrupt is generated.
4. The Receive interrupt is called. A character is read from the serial port, placed at location 0, and the value of the Receive Head Pointer field is increased by 1.



5. Eight locations are available in the buffer to receive data (bytes 1 to 8). To increase the number of bytes that are available in the buffer to receive data, the caller can read data from the buffer and logically increase the value of the Receive Tail Pointer field up to (but not exceeding) the value of the Receive Head Pointer field. If the value of the Receive Head Pointer field equals the value of the Receive Tail Pointer field, the buffer is empty.



6. Nine locations (bytes 1 to 9) are now available in the receive buffer. When the next Receive interrupt is generated, the Interrupt routine is called, and a character is read from the serial port. The value of the Receive Head Pointer field is increased by 1. Assuming that the caller does not change the value of the Receive Tail Pointer field, the process repeats as interrupts occur. BIOS receives a character, places the character at location 1, and increases the value of the Receive Head Pointer field by 1. This process is repeated until BIOS receives a character and places that character at location 9. Because the value of the Receive Head Pointer field, when increased by 1, is outside the buffer range, BIOS resets the value of the Receive Head Pointer field to 0.



7. The receive buffer is full when the value of the Receive Head Pointer field is 1 less than the value of the Receive Tail Pointer field, or if the value of the Receive Tail Pointer field is 0 and the value of the Receive Head Pointer field is 1 less than the value of the Receive-Buffer Length field. If the receive-buffer-full condition still exists when the Interrupt routine is called, the character that caused the interrupt is lost.

## **Direct Memory Access (DMA) Receive**

- In a system that supports DMA mode, data can be received through a DMA device. The DMA device transfers data from the receiver to memory until the receiver is empty or until the DMA device reaches the terminal count. This technique requires little processing-unit intervention.
- To determine whether DMA mode is supported, call the Read Device Parameters function (hex 03). Bit 2 of the FIFO-Mode Status field indicates whether DMA mode is available.
- When bit 0 of the Enhanced Mode field is set to 1, DMA mode is enabled. BIOS attempts to initialize the DMA device with the values that are in the Receive Physical Address Buffer Pointer field and the Receive-Buffer Transfer Length field.
- In DMA mode, first-in, first-out (FIFO) mode is automatically enabled and is always active.
- DMA-transmit operations are controlled through the trigger level that is specified through the Reset/Initialize function (hex 05) or the Set Receive Trigger Level function (hex 16). When characters are received but do not fill the receiver to the trigger level, a Receiver Time-Out interrupt is generated if no more characters are received within four character time frames. This enables the serial port to empty the receiver when no more characters remain to be sent.
- The following fields must be initialized and passed for a DMA-receive operation:
  - Enhanced Mode field
  - Compare Character Match 1 field (if the Compare Character Function 1 field is not set to 0)
  - Compare Character Function 1 field
  - Compare Character Match 2 field (if the Compare Character Function 2 field is not set to 0)
  - Compare Character Function 2 field
  - Compare Character Match 3 field (if the Compare Character Function 3 field is not set to 0)
  - Compare Character Function 3 field
  - Receive-Buffer Offset field (if bit 0 of the Receive Control field is set to 1)
  - Receive-Buffer Segment field (if bit 0 of the Receive Control field is set to 1)
  - Null-Stripping Indicator field
  - Receive-Buffer Length field (if bit 0 of the Receive Control field is set to 1)

- Receive Head Pointer field (if bit 0 of the Receive Control field is set to 1)
  - Receive Tail Pointer field (if bit 0 of the Receive Control field is set to 1)
  - Receive Control field (if bit 1 of the Enhanced Mode field is set to 1)
  - Operation Control field
  - Receive Character Count field (if bit 3 of the Enhanced Mode field is set to 1).
  - Receive-Buffer Switch Threshold field (if bit 15 of the Receive Control field is set to 1)
  - Receive Physical Address Buffer Pointer field
  - Receive-Buffer Transfer Length field.
- An arbitration level and a DMA channel must be allocated to the DMA device to accomplish a data transfer. The system operator assigns arbitration levels during setup. Asynchronous communication BIOS automatically allocates DMA channels before any DMA receive operations are started. If the arbitration level and DMA channel are available, the DMA device is initialized with the parameters that are passed in the request block, and the serial port immediately begins receiving data.
  - If bit 0 of the Receive Control field is set to 1 and either the arbitration level or the necessary DMA channel is not available, the serial port is initialized to operate in PIO mode, and any DMA-receive operation will default to PIO mode. If this occurs, BIOS sets bit 0 of the Enhanced Mode field to 0 to indicate that the request has defaulted to PIO mode, and operation continues as if the request had been made in PIO mode. When the receive operation defaults to PIO mode, the receiver receives data in FIFO mode, using the current trigger-level value. If bit 0 of the Transmit Control field is set to 0 and the arbitration level or DMA channel is not available, the Return Code field is set to hex 8081 (Arbitration Level Not Available), and no action is taken.
  - If bit 15 of the Receive Control field is set to 1, pre-terminal-count buffer transfer support is enabled. Pre-terminal-count buffer transfer support allows data to be transferred at higher baud rates without overruns, and it provides a way to receive an interrupt, before the terminal count is reached, while a DMA-receive operation is in progress. BIOS determines the rate at which interrupts occur, and interrupts occur continuously until the end of the buffer is within reach of the number of paragraphs (16 bytes) that are specified in the Receive-Buffer Switch Threshold field. The pre-terminal-count threshold must be less than the receive-buffer length divided by 16.

- BIOS toggles the buffer by reloading the DMA device with the values that are in the Receive Physical Address Buffer Pointer field and the Receive-Buffer Transfer Length field. Bit 3 of the Operation Status field is set to 1 to indicate that the buffer has been toggled. Any residual data that is received while the Pre-Terminal Count interrupt is being serviced is contained in the original buffer. Any data that is received after the DMA device has been reloaded is contained in the new buffer.
- If a receive request reaches terminal count and the caller has not allowed enough slack in the Receive-Buffer Switch Threshold field, the buffer is toggled, and the Return Code field is set to hex 0001 (Stage on Interrupt).
- When bit 15 of the Receive Control field and bit 3 of the Operation Status field are both set to 1, the caller should update the Receive Physical Address Buffer Pointer field and the Receive-Buffer Transfer Length field immediately after calling the Start routine.
- If bit 5 of the Receive Control field is set to 1, BIOS is forced to toggle the buffer at the next Receive interrupt. BIOS initiates another transfer operation by reloading the DMA device with the values that are in the Receive Physical Address Buffer Pointer field and the Receive-Buffer Transfer Length field.
- If a receive request reaches terminal count, BIOS toggles the buffer.
- The Receive Current Transfer Count field returns the number of transfers that remain before the terminal count will be reached. This field is valid only when all the following conditions exist:
  - The request was successfully started in DMA mode.
  - The request did not default to PIO mode.
  - The arbitration level was successfully allocated.
  - A receive-buffer toggle occurred.
- If bit 5 (stop transmitter on any line error) of the Enhanced Mode field is set to 1, the transmitter is stopped after the shift register is emptied on any received line error. In DMA mode, BIOS programs the serial port to interrupt on line-status errors. The Operation Status field returns the results of these interrupts. Note that a Transmit Interrupt function (hex 0E) must be outstanding.
- If bit 15 of the Enhanced Mode field is set to 1, BIOS assumes that another process (such as an external resource manager) will allocate and deallocate DMA arbitration-level resources, and it starts the DMA-receive operation without allocating the

arbitration level. When this bit is set to 1, ABIOS does not deallocate the arbitration level when the receive operation is complete. If bit 14 of the Enhanced Mode field is set to 0, ABIOS allocates the arbitration level before starting the DMA-receive operation.

- In DMA mode, if the Receive-Buffer Length field (offset hex 38) is set to 0 when the Start routine is called, a DMA channel is allocated, but no data transfers occur, and the Return Code field is set to hex 0000 (Operation Successfully Completed).

### Service-Specific Input

Size	Offset	Description
Word	18H	Enhanced mode <ul style="list-style-type: none"> <li>Bit 15 - Assume receive arbitration level               <ul style="list-style-type: none"> <li>= 0 - ABIOS does not assume that the arbitration level is assigned</li> <li>= 1 - ABIOS assumes that the arbitration level is assigned</li> </ul> </li> <li>Bits 14 to 6 - Reserved</li> <li>Bit 5 - Stop transmitter on any line error               <ul style="list-style-type: none"> <li>= 0 - Disable</li> <li>= 1 - Enable</li> </ul> </li> <li>Bit 4 - Received-data status               <ul style="list-style-type: none"> <li>= 0 - Disable</li> <li>= 1 - Enable</li> </ul> </li> <li>Bit 3 - Interrupt on receiver-character-count = 0               <ul style="list-style-type: none"> <li>= 0 - Disable</li> <li>= 1 - Enable</li> </ul> </li> <li>Bit 2 - Reserved</li> <li>Bit 1 - DMA-receive mode               <ul style="list-style-type: none"> <li>= 0 - Disable</li> <li>= 1 - Enable</li> </ul> </li> <li>Bit 0 - Reserved</li> </ul>
Byte	1AH	Compare character match 1

Size	Offset	Description
Byte	1BH	Compare character function 1 Bits 7 to 4 - Reserved Bit 3 - Start transmitter on a character match = 0 - Transmitter not started on a character match = 1 - Transmitter started on a character match (the transmitter will not start if bit 2 is set to 1) Bit 2 - Stop transmitter on a character match = 0 - Transmitter not stopped on a character match = 1 - Transmitter stopped on a character match (if the transmitter is stopped with a character match, a call to the Start Transmission function (hex 18) can be used to restart the transmitter) Bit 1 - Delete character on a character match = 0 - Character not deleted on a character match = 1 - Character deleted on a character match Bit 0 - Interrupt on character match = 0 - Interrupt not enabled on a character match = 1 - Interrupt enabled on a character match (the interrupt occurs as soon as there is a match with a received data character)
Byte	1CH	Compare character match 2
Byte	1DH	Compare character function 2 (see the description of the Compare Character Function 1 field)
Byte	1EH	Compare character match 3
Byte	1FH	Compare character function 3 (see the description of the Compare Character Function 1 field)
Word	20H	Reserved
Word	22H	Receive-buffer offset
Word	24H	Receive-buffer segment
Word	28H	Null-stripping indicator = 00H - Disable = 01H - Enable = 02H to FFH - Reserved
Byte	29H	Receive-buffer switch threshold
Word	38H	Receive-buffer length, in bytes
Word	3AH	Reserved
Word	3CH	Receive head pointer
Word	3EH	Reserved
Word	40H	Receive tail pointer
Word	42H	Reserved



Size	Offset	Description
Word	46H	Receive control <ul style="list-style-type: none"> <li>Bit 15 - Receive pre-terminal count               <ul style="list-style-type: none"> <li>= 0 - Disable</li> <li>= 1 - Enable</li> </ul> </li> <li>Bits 14 to 6 - Reserved</li> <li>Bit 5 - Buffer toggle               <ul style="list-style-type: none"> <li>= 0 - Do not force toggle</li> <li>= 1 - Force toggle</li> </ul> </li> <li>Bits 4, 3 - Reserved</li> <li>Bit 2 - Receive search arbitration level direction               <ul style="list-style-type: none"> <li>= 0 - Increasing, starting with the lowest arbitration level</li> <li>= 1 - Decreasing, starting with the highest arbitration level</li> </ul> </li> <li>Bit 1 - Receive request arbitration level               <ul style="list-style-type: none"> <li>= 0 - No arbitration level was passed</li> <li>= 1 - Arbitration level was passed</li> </ul> </li> <li>Bit 0 - Receive default to PIO               <ul style="list-style-type: none"> <li>= 0 - Return error if DMA is not available</li> <li>= 1 - Use PIO mode if DMA is not available</li> </ul> </li> </ul>
Byte	49H	Receive arbitration level (if dynamic arbitration is supported)
Word	4BH	Reserved
Word	52H	Receive character count (maximum value = hex FF)
DWord	66H	Receive physical address buffer pointer
DWord	6AH	Receive-buffer transfer length

## Service-Specific Output

Size	Offset	Description
Word	16H	Additional operation status <ul style="list-style-type: none"> <li>Bits 15 to 4 - Reserved</li> <li>Bit 3 - Receive Line-Status Error interrupt               <ul style="list-style-type: none"> <li>= 0 - Did not occur</li> <li>= 1 - Occurred</li> </ul> </li> <li>Bit 2 - Receive Time-Out interrupt               <ul style="list-style-type: none"> <li>= 0 - Did not occur</li> <li>= 1 - Occurred</li> </ul> </li> <li>Bit 1 - Reserved</li> <li>Bit 0 - Additional interrupts pending               <ul style="list-style-type: none"> <li>= 0 - No additional interrupts pending</li> <li>= 1 - Additional interrupts pending</li> </ul> </li> </ul>
Word	18H	Enhanced mode <ul style="list-style-type: none"> <li>Bits 15 to 2 - Reserved</li> <li>Bit 1 - DMA-receive mode               <ul style="list-style-type: none"> <li>= 0 - Disabled</li> <li>= 1 - Enabled</li> </ul> </li> <li>Bit 0 - Reserved</li> </ul>
Word	3CH	Receive head pointer
Word	3EH	Reserved

Size	Offset	Description
Word	46H	Receive control Bits 15 to 6 - Reserved Bit 5 - Buffer toggle = 0 - Toggle not forced = 1 - Toggle forced Bits 4, 3 - Reserved Bit 2 - Receive search arbitration level direction = 0 - Increasing, starting with the lowest arbitration level = 1 - Decreasing, starting with the highest arbitration level Bit 1 - Receive request arbitration level = 0 - No arbitration level was passed = 1 - Arbitration level was passed Bit 0 - Reserved
Byte	49H	Receive arbitration level (if dynamic arbitration is supported)
Word	4BH	Operation status; return only from Interrupt routine Bit 15 - Compare Character Match 3 interrupt = 0 - Did not occur = 1 - Occurred Bit 14 - Compare Character Match 2 interrupt = 0 - Did not occur = 1 - Occurred Bit 13 - Compare Character Match 1 interrupt = 0 - Did not occur = 1 - Occurred Bit 12 - Overrun error with null data byte found = 0 - Inactive = 1 - Active only if null data was found and discarded Bit 11 - Break detected = 0 - Inactive = 1 - Active Bit 10 - Framing error = 0 - Inactive = 1 - Active Bit 9 - Parity error = 0 - Inactive = 1 - Active Bit 8 - Overrun error = 0 - Inactive = 1 - Active Bits 7, 6 - Reserved Bit 5 - Receive buffer full; data discarded = 0 - Inactive = 1 - Active Bit 4 - Receive buffer full = 0 - Inactive = 1 - Active Bit 3 - Buffer toggle = 0 - Did not occur = 1 - Occurred

Size	Offset	Description
		Bit 2 - Receive Character Count/ Pre-Terminal-Count Buffer Transfer interrupt = 0 - Did not occur = 1 - Occurred
		Bit 1 - Reserved
		Bit 0 - Receive interrupt in progress = 0 - Inactive = 1 - Active
DWord	5AH	Receive current transfer count

## 10H—Combined (Transmit and Receive) Interrupts

- This function is used to perform a transmit operation and a receive operation in a single request. It can be used also to perform a receive-only operation.
- PIO-transmit and PIO-receive operations are defined as requests that are initiated in PIO mode or DMA requests that default to PIO mode. DMA-transmit and DMA-receive operations are defined as requests that are initiated in DMA mode and do not default to PIO mode. See “Programmed Input/Output (PIO) Transmit” on page 6-ID06-16 for an explanation of PIO-transmit operations. See “Programmed Input/Output (PIO) Receive” on page 6-ID06-27 for an explanation of PIO-receive operations. See “Direct Memory Access (DMA) Transmit” on page 6-ID06-21 for an explanation of DMA-transmit operations. See “Direct Memory Access (DMA) Receive” on page 6-ID06-32 for an explanation of DMA-receive operations.
- A receive operation can be initiated by the Start routine with only the receive parameters set (the Transmit-Buffer Length field is set to 0 for a PIO-receive operation, and the Transmit-Buffer Transfer Length field is set to 0 for a DMA-receive operation). If this function was initially called with only the receive operation enabled, the transmit operation can be enabled on a subsequent call with the Transmit-Buffer Length field (for PIO mode) or the Transmit-Buffer Transfer Length field (for DMA mode) set to a nonzero value. Both fields must be set to nonzero values when bit 0 (transmit default to PIO) of the Transmit Control field is set to 1.
- A transmit operation cannot be initiated on the first call to the Start routine.
- When the Start routine is called to enable a receive operation, the Return Code field must be set to hex FFFF (Return Code Not Valid). However, if the Start routine is called for a second time to enable the transmit operation, the previously-stored value of the Return Code field must not be changed. On return from the

second call to the Start routine, the value of the Return Code field is undefined and should be ignored.

- During receive operations in PIO mode, the transmit operation is never disabled; no test is performed to determine whether the value of the Transmit Head Pointer field is equal to the value of the Transmit Tail Pointer field.
- To optimize performance after a receive operation is serviced, BIOS checks for an interrupt-pending condition. If a transmit interrupt is pending, the interrupt is serviced, and the Return Code field is set to hex 0009 (Attention, Stage on Interrupt). If a second receive interrupt is pending, BIOS does not service the pending interrupt, and it returns to the caller.
- If the Return Code field is set to hex 0009 (Attention, Stage on Interrupt), the Operation Status of Previous Interrupt field contains the status that was returned when the first interrupt was serviced. The Return Code of Previous Interrupt field contains the return-code value that was returned from the first interrupt. The Operation Status field and the Return Code field contain the values from the second interrupt. The Operation Status of Previous Interrupt field and the Return Code of Previous Interrupt field must be initialized to 0 by the caller before the Start routine is called.
- In a combined operation, if the receive operation is completed, the transmit operation is not allowed to start, and the Return Code field is set to hex C020 (No Receive Active in Combined Interrupt). However, if the transmit operation is in DMA mode and the transmit-buffer length is 0, the transmit operation is allowed to start, to allocate the transmit arbitration level.
- When this function is called by the Start routine for the first time, if the value of the Transmit-Buffer Length field (in PIO mode) or the Transmit-Buffer Transfer Length field (in DMA mode) is 0, and the value of the Receive-Buffer Length field (in PIO mode) or the Receive-Buffer Transfer Length field (in DMA mode) is nonzero, this function acts as a Receive Interrupt function (hex 0F). When this function is called by the Start routine for the second time, if the value of the Transmit-Buffer Length field (in PIO mode) or the Transmit-Buffer Transfer Length field (in DMA mode) is 0, no action is performed, and the Return Code field is set to hex 0001 (Stage on Interrupt).
- If a Transmit interrupt and a Receive interrupt are both enabled and bit 3 (interrupt on receiver-character-count=0) and bit 2 (transmit byte pacing) of the Enhanced Mode field are set to 1, bit 3 takes precedence over bit 2. The functions that are associated

with bit 3 and bit 2 are mutually exclusive at the serial port. In the same way, bit 15 (receive pre-terminal count) of the Receive Control field takes precedence over bit 2 (transmit byte pacing) of the Enhanced Mode field.

- If bit 0 (transmit default to PIO) of the Transmit Control field and bit 0 (receive default to PIO) of the Receive Control field are both set to 0 and the necessary DMA channels are not available, the Return Code field is set to hex 8086 (No DMA Channel Available), and neither transfer is initiated.
- The possible values of the Return Code field are hex 0000, 0001, 0005, 0009, 0081, 8000, 8001, 8081, 8082, 8083, 8084, 8085, 8086, 8087, 9000, and C020.

### Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
Word	12H	Transmit-buffer offset
Word	14H	Transmit-buffer segment
Word	18H	Enhanced mode
		Bit 15 - Assume receive arbitration level
		= 0 - BIOS does not assume that the arbitration level is assigned
		= 1 - BIOS assumes that the arbitration level is assigned
		Bit 14 - Assume transmit arbitration level
		= 0 - BIOS does not assume that the arbitration level is assigned
		= 1 - BIOS assumes that the arbitration level is assigned
		Bits 13 to 6 - Reserved
		Bit 5 - Stop transmitter on any line error
		= 0 - Disable
		= 1 - Enable
		Bit 4 - Received-data status
		= 0 - Disable
		= 1 - Enable
		Bit 3 - Interrupt on receiver-character-count = 0
		= 0 - Disable
		= 1 - Enable
		Bit 2 - Transmit byte pacing
		= 0 - Disable
		= 1 - Enable
		Bit 1 - DMA-receive mode
		= 0 - Disable
		= 1 - Enable
		Bit 0 - DMA-transmit mode
		= 0 - Disable
		= 1 - Enable
Byte	1AH	Compare character match 1

Size	Offset	Description
Byte	1BH	Compare character function 1 Bits 7 to 4 - Reserved Bit 3 - Start transmitter on a character match = 0 - Transmitter not started on a character match = 1 - Transmitter started on a character match (the transmitter will not start if bit 2 is set to 1) Bit 2 - Stop transmitter on a character match = 0 - Transmitter not stopped on a character match = 1 - Transmitter stopped on a character match (if the transmitter is stopped with a character match, a call to the Start Transmission function (hex 18) can be used to restart the transmitter) Bit 1 - Delete character on a character match = 0 - Character not deleted on a character match = 1 - Character deleted on a character match Bit 0 - Interrupt on character match = 0 - Interrupt not enabled on a character match = 1 - Interrupt enabled on a character match (the interrupt occurs as soon as there is a match with a received data character)
Byte	1CH	Compare character match 2
Byte	1DH	Compare character function 2 (see the description of the Compare Character Function 1 field)
Byte	1EH	Compare character match 3
Byte	1FH	Compare character function 3 (see the description of the Compare Character Function 1 field)
Word	20H	Reserved
Word	22H	Receive-buffer offset
Word	24H	Receive-buffer segment
Byte	28H	Null-stripping indicator = 00H - Disabled = 01H - Enabled = 02H to FFH - Reserved
Byte	29H	Receive-buffer switch threshold/ receive demand-allocation threshold
Word	2CH	Transmit-buffer length, in bytes
Word	2EH	Reserved
Word	30H	Transmit head pointer
Word	32H	Reserved
Word	34H	Transmit tail pointer
Word	36H	Reserved
Word	38H	Receive-buffer length, in bytes
Word	3AH	Reserved
Word	3CH	Receive head pointer
Word	3EH	Reserved
Word	40H	Receive tail pointer
Word	42H	Reserved

Size	Offset	Description
Word	44H	<b>Transmit control</b> Bit 15 - Transmit arbitration level deallocation = 0 - Do not deallocate arbitration level when transfer is complete = 1 - Deallocate arbitration level when transfer is complete Bits 14 to 3 - Reserved Bit 2 - Transmit search arbitration level direction = 0 - Increasing, starting with the lowest arbitration level = 1 - Decreasing, starting with the highest arbitration level Bit 1 - Transmit request arbitration level = 0 - No arbitration level was passed = 1 - Arbitration level was passed Bit 0 - Transmit default to PIO = 0 - Return error if DMA is not available = 1 - Use PIO mode if DMA is not available
Word	46H	<b>Receive control</b> Bit 15 - Receive pre-terminal count = 0 - Disable = 1 - Enable Bits 14 to 6 - Reserved Bit 5 - Buffer toggle = 0 - Do not force toggle = 1 - Force toggle Bits 4 to 1 - Reserved Bit 0 - Receive default to PIO = 0 - Return error if DMA is not available = 1 - Use PIO mode if DMA is not available
Byte	49H	Receive arbitration level (if dynamic arbitration is supported)
Byte	4AH	Transmit arbitration level (if dynamic arbitration is supported)
Word	4BH	Operation control (reserved)
Word	52H	Receive character count (maximum value = hex FF)
Word	54H	Transmit-byte-pacing rate (maximum value = hex FF)
DWord	5EH	Transmit physical address buffer pointer
DWord	62H	Transmit-buffer transfer length
DWord	66H	Receive physical address buffer pointer
DWord	6AH	Receive-buffer transfer length

# Service-Specific Output

Size	Offset	Description
Word	16H	Additional operation status Bits 15 to 4 - Reserved Bit 3 - Receive Line-Status Error interrupt = 0 - Did not occur = 1 - Occurred Bit 2 - Receive Time-Out interrupt = 0 - Did not occur = 1 - Occurred Bit 1 - Transmit Byte Pacing interrupt = 0 - Did not occur = 1 - Occurred Bit 0 - Additional interrupts pending = 0 - No additional interrupts pending = 1 - Additional interrupts pending
Word	18H	Enhanced mode Bits 15 to 2 - Reserved Bit 1 - DMA-receive mode = 0 - Disabled = 1 - Enabled Bit 0 - DMA-transmit mode = 0 - Disabled = 1 - Enabled
Word	34H	Transmit tail pointer
Word	36H	Reserved
Word	3CH	Receive head pointer
Word	3EH	Reserved
Word	46H	Receive control Bits 15 to 6 - Reserved Bit 5 - Buffer toggle = 0 - Toggle not forced = 1 - Toggle forced Bits 4 to 0 - Reserved
Byte	49H	Receive arbitration level (if dynamic arbitration is supported)
Byte	4AH	Transmit arbitration level (if dynamic arbitration is supported)
Word	4BH	Operation status; return only from Interrupt routine Bit 15 - Compare Character Match 3 interrupt = 0 - Did not occur = 1 - Occurred Bit 14 - Compare Character Match 2 interrupt = 0 - Did not occur = 1 - Occurred Bit 13 - Compare Character Match 1 interrupt = 0 - Did not occur = 1 - Occurred Bit 12 - Overrun error with null data byte found = 0 - Inactive = 1 - Active only if null data was found and discarded



Size	Offset	Description
		Bit 11 - Break detected = 0 - Inactive = 1 - Active
		Bit 10 - Framing error = 0 - Inactive = 1 - Active
		Bit 9 - Parity error = 0 - Inactive = 1 - Active
		Bit 8 - Overrun error = 0 - Inactive = 1 - Active
		Bit 7 - Transmit buffer empty; Transmitter Holding register empty = 0 - Inactive = 1 - Active
		Bit 6 - Transmit buffer empty = 0 - Inactive = 1 - Active
		Bit 5 - Receive buffer full; data discarded = 0 - Inactive = 1 - Active
		Bit 4 - Receive buffer full = 0 - Inactive = 1 - Active
		Bit 3 - Buffer toggle = 0 - Did not occur = 1 - Occurred
		Bit 2 - Receive Character Count/ Pre-Terminal-Count Buffer Transfer interrupt = 0 - Did not occur = 1 - Occurred
		Bit 1 - Transmit interrupt in progress = 0 - Inactive = 1 - Active
		Bit 0 - Receive interrupt in progress = 0 - Inactive = 1 - Active
Word	4DH	Operation status of previous interrupt
Word	4FH	Return code of previous interrupt
DWord	56H	Reserved
DWord	5AH	Receive current transfer count

## 11H—Modem Status Interrupt

- This function enables the Modem Status interrupt. The Modem Status field is returned to the caller on exit from the Start and Interrupt routines.
- In PIO mode, if the modem-status-interrupt request block is processed before the receive or transmit request block at interrupt time, the BIOS routine detects a change in the modem status, even if the higher-priority interrupts in the Interrupt Identification register are still pending. This enables an interrupt handler to detect a change in modem status before any data is received or transmitted.
- The 'high-data-rate signal selector' signal from dual-range asynchronous data-communication equipment is used to select between two ranges of data-signaling rates. When bit 15 of the Enhanced Function Control field is set to 1, an interrupt occurs when the 'high-data-rate signal selector' signal changes state. The current state of the 'high-data-rate signal selector' signal is returned in bit 8 of the Enhanced Function Status field.
- The possible values of the Return Code field are hex 0001, 0005, 0081, 8000, and 8001.

### Service-Specific Input

Size	Offset	Description
Word	18H	Enhanced function control Bit 15 - 'High data rate' signal interrupt = 0 - Disable = 1 - Enable Bits 14 to 0 - Reserved
DWord	1AH	Reserved

### Service-Specific Output

Size	Offset	Description
Word	18H	Enhanced function status Bits 15 to 9 - Reserved Bit 8 - 'High data rate' signal status = 0 - 'High data rate' signal is inactive = 1 - 'High data rate' signal is active Bits 7 to 0 - Reserved

Size Byte	Offset 4AH	Description
		Modem status
		Bit 7 - Data-carrier detect = 0 - Inactive = 1 - Active
		Bit 6 - Ring indicator = 0 - Inactive = 1 - Active
		Bit 5 - Data set ready = 0 - Inactive = 1 - Active
		Bit 4 - Clear to send = 0 - Inactive = 1 - Active
		Bit 3 - Delta data-carrier detect = 0 - Inactive = 1 - Active
		Bit 2 - Trailing-edge ring indicator = 0 - Inactive = 1 - Active
		Bit 1 - Delta data set ready = 0 - Inactive = 1 - Active
		Bit 0 - Delta clear to send = 0 - Inactive = 1 - Active

## 12H—Cancel

- This function cancels a requested interrupt operation. Any combinations of interrupts can be canceled. Associated interrupts are disabled on return from this function. If an outstanding request is associated with the interrupt type that is canceled, that request block is considered canceled and must be appropriately deallocated.
- DMA transmit and receive operations can be canceled individually within a Combined Interrupt function (hex 10).
- 'High Data Rate' Signal interrupts are canceled when the Modem Status Interrupt function (hex 11) is canceled.
- If a Receive interrupt and a Transmit interrupt are both enabled in a Combined Interrupts function (hex 10) and the Cancel function is called to cancel both interrupts, the combined request block is considered canceled.
- If a Receive interrupt and a Transmit interrupt are both enabled in a Combined Interrupts function (hex 10) and the Cancel function is called to cancel either the Transmit interrupt or the Receive interrupt, but not both, the request block is considered active.
- If a Receive interrupt and a Transmit interrupt are both enabled in a Combined Interrupts function (hex 10) and the Cancel function

is called to cancel the Transmit interrupt, a call to the Start routine to reenable the Transmit interrupt is permitted.

- If a Receive interrupt and a Transmit interrupt are both enabled in a Combined Interrupts function (hex 10) and the Receive interrupt has been cancelled through the Receive interrupt, to reenable the Receive interrupt through the Combined Interrupts function (hex 10), the caller must first cancel the Transmit interrupt, then call the Combined Interrupts function (hex 10) with the Receive interrupt enabled.
- If only the Receive interrupt is enabled in a Combined Interrupts function (hex 10) and the Cancel function is called to cancel the Receive interrupt, the combined request block is considered canceled.
- If the Receive Interrupt function is canceled, the Receive interrupt and the Line Status interrupt are disabled. Canceling the Receive interrupt also disables the Interrupt On Receiver-Character-Count=0 interrupt and the three Compare Character functions.
- If the Transmit Interrupt function is canceled, the Transmit interrupt and the Transmit Byte Pacing interrupt are disabled. Canceling the Transmit Interrupt function also cancels the Stop Transmission and Send Data function (hex 17).
- The Transmit Current Transfer Count field and the Receive Current Transfer Count field return the number of transfers that remain before DMA terminal count will be reached.
- See page 6-ID06-23 for an explanation of bit 14 (assume transmit arbitration level) of the Enhanced Mode field. See page 6-ID06-34 for an explanation of bit 15 (assume receive arbitration level) of the Enhanced Mode field.
- The possible values of the Return Code field are hex 0000, 8000, 8001, 8081, 8082, 8083, 8084, 8085, 8086, and 8087.

**Service-Specific Input**

Size	Offset	Description
Word	18H	Enhanced mode Bit 15 - Assume receive arbitration level = 0 - ABIOs does not assume that the arbitration level is assigned = 1 - ABIOs assumes that the arbitration level is assigned Bit 14 - Assume transmit arbitration level = 0 - ABIOs does not assume that the arbitration level is assigned = 1 - ABIOs assumes that the arbitration level is assigned Bits 13 to 0 - Reserved
Byte	51H	Interrupt type to be canceled Bit 7 - Receive arbitration level deallocation = 0 - Do not deallocate = 1 - Deallocate Bit 6 - Transmit arbitration level deallocation = 0 - Do not deallocate = 1 - Deallocate Bit 5 - Modem status = 0 - Do not disable = 1 - Disable Bit 4 - Reserved Bit 3 - Transmit = 0 - Do not disable = 1 - Disable Bit 2 - Receive = 0 - Do not disable = 1 - Disable Bit 1 - Stop transmission and send data = 0 - Do not disable = 1 - Disable Bit 0 - 'High data rate' signal = 0 - Do not cancel request = 1 - Cancel request

**Service-Specific Output**

Size	Offset	Description
DWord	56H	Transmit current transfer count
DWord	5AH	Receive current transfer count

**13H—Return Line Status**

- This function returns the line status.
- The value of the Return Code field is hex 0000.

**Service-Specific Input**

Size	Offset	Description
None		

### Service-Specific Output

Size	Offset	Description
Byte	49H	Line status
		Bit 7 - Error in receiver FIFO
		= 0 - No error in FIFO
		= 1 - Error in receiver FIFO
		Bit 6 - Transmitter empty
		= 0 - Inactive
		= 1 - Active
		Bit 5 - Transmitter holding register empty
		= 0 - Inactive
		= 1 - Active
		Bit 4 - Break interrupt
		= 0 - Inactive
		= 1 - Active
		Bit 3 - Framing error
		= 0 - Inactive
		= 1 - Active
		Bit 2 - Parity error
		= 0 - Inactive
		= 1 - Active
		Bit 1 - Overrun error
		= 0 - Inactive
		= 1 - Active
		Bit 0 - Data ready
		= 0 - Inactive
		= 1 - Active

### 14H—Return Modem Status

- This function returns the modem status. If this function is called and the Modem Status interrupt is enabled, the Return Code field is set to hex 8000 (Device Busy).
- The possible values of the Return Code field are hex 0000 and 8000.

### Service-Specific Input

Size	Offset	Description
None		

### Service-Specific Output

Size	Offset	Description
Word	18H	Enhanced function status
		Bits 15 to 9 - Reserved
		Bit 8 - 'High data rate' signal status
		= 0 - 'High data rate' signal is inactive
		= 1 - 'High data rate' signal is active
		Bits 7 to 0 - Reserved

Size	Offset	Description
Byte	4AH	Modem status
		Bit 7 - Data-carrier detect
		= 0 - Inactive
		= 1 - Active
		Bit 6 - Ring indicator
		= 0 - Inactive
		= 1 - Active
		Bit 5 - Data set ready
		= 0 - Inactive
		= 1 - Active
		Bit 4 - Clear to send
		= 0 - Inactive
		= 1 - Active
		Bit 3 - Delta data-carrier detect
		= 0 - Inactive
		= 1 - Active
		Bit 2 - Trailing-edge ring indicator
		= 0 - Inactive
		= 1 - Active
		Bit 1 - Delta data set ready
		= 0 - Inactive
		= 1 - Active
		Bit 0 - Delta clear to send
		= 0 - Inactive
		= 1 - Active

## 15H—Reserved

## 16H—Set Receive Trigger Level

- This function sets the first-in, first-out (FIFO) trigger level and enables the FIFO mode for the serial port, according to the input parameters.
- The possible values of the Return Code field are hex 0000 and 8001.

## Service-Specific Input

Size	Offset	Description
Word	18H	Enhanced function control
		Bits 15 to 4 - Reserved
		Bit 3 - Reset transmitter FIFO
		= 0 - Take no action
		= 1 - Discard all characters
		in transmitter FIFO
		Bit 2 - Reset receiver FIFO
		= 0 - Take no action
		= 1 - Discard all characters
		in receiver FIFO
		Bits 1, 0 - FIFO enable/disable control
		= 00 - No action taken
		= 10 - Reserved
		= 01 - Disable FIFO
		= 11 - Enable FIFO

Size	Offset	Description
Byte	28H	Receive trigger level = 00H - 1 byte = 01H - 4 bytes = 02H - 8 bytes = 03H - 14 bytes = 04H to FFH - Reserved

## Service-Specific Output

Size	Offset	Description
None		

### 17H—Stop Transmission and Send Data

- This function empties the shift register and stops transmitting, regardless of the number of characters that remain in the FIFO buffer. When the transmitter is stopped, a string of characters, a single character, or no character is transmitted, as specified in bits 2, 1, and 0 of the Function Control field. The string of characters to which the Logical String Pointer field points is loaded into the shift register, and each character is transmitted individually.
- The Transmitter Holding Register Empty interrupt is enabled to signal to the system when the single-character transmission is complete. Each time the interrupt occurs, a call to the Interrupt routine must be executed with the Stop Transmission and Send Data function request block as a parameter. This process is repeated for each character until the value of the Character Count to Be Transmitted field is exhausted. If multiple characters are sent, multiple interrupts occur. Calling the Interrupt routine to pass the suspended Transmit interrupt function (hex 0E) request block or the suspended Combined Interrupts function (hex 10) request block is not necessary.
- Before the Start Transmission (hex 18) function is issued to restart the DMA transfer, each character to which the Logical String Pointer field points must have been transmitted, and the interrupt must have been serviced.
- If this function is executed with the Function Control field set to hex 00, the transmitter is stopped, no characters are sent to the Transmitter Holding Register, and the Return Code field is set to hex 0000 (Operation Successfully Completed).
- If this function is executed with the Function Control field set to hex 01, the transmitter is stopped, BIOS writes a single character to the Transmitter Holding register, and the Return Code field is set to hex 0001 (Stage on Interrupt). When the



Transmitter Holding register is emptied, the interrupt occurs. A call to the Interrupt routine causes the character to be written to the Transmitter Holding register, and the Return Code field is set to hex 0000 (Operation Successfully Completed).

- If this function is executed with the Function Control field set to hex 02, the transmitter is stopped, ABIOS sends a string of characters to the Transmitter Holding register, and the Return Code field is set to hex 0001 (Stage on Interrupt) after the first character and any subsequent characters are written to the Transmitter Holding register. When the entire character string has been sent, the Return Code field is set to hex 0000 (Operation Successfully Completed).
- This function can be stopped by setting bit 1 of the Interrupt Type to Be Canceled field in the Cancel function (hex 12) to 1 and executing the Cancel function.
- A transmit operation that is interrupted with this function should be restarted through the Start Transmission function (hex 18).
- The possible values of the Return Code field are hex 0000, 0001, 0005, 0081, 8000 and 8001.

### Service-Specific Input

Size	Offset	Description
DWord	12H	Logical string pointer
Word	18H	Reserved
Byte	1AH	Character to be transmitted
Byte	1BH	Function control <ul style="list-style-type: none"><li>= 00H - Stop transmitter</li><li>= 01H - Stop transmitter and send a character</li><li>= 02H - Stop transmitter and send a string</li></ul>
Word	2CH	Character count to be transmitted
Word	2EH	Reserved

### Service-Specific Output

Size	Offset	Description
Word	2CH	Number of characters that were transmitted
Word	2EH	Reserved

### 18H—Start Transmission

- This function is used to restart or continue a transmit request that has been interrupted by the Stop Transmission and Send Data function (hex 17) or by a stop-transmitter-on-a-character-match condition in the Receive Interrupt function (hex 0F) or in the Combined Interrupts function (hex 10).
- No attempt is made to verify that a transmit request is outstanding.
- If this function is called while a request to the Stop Transmission and Send Data function is outstanding, the Return Code field is set to hex 8000 (Device Busy, Request Refused).
- The possible values of the Return Code field are hex 0000, 8000, and 8001.



### Service-Specific Input

Size	Offset	Description
Word	18H	Reserved

### Service-Specific Output

Size	Offset	Description
None		



### 19H—Reset Error/Break

- This function resets bit 0 (error/break) in the received-data status byte, which is optionally stored with received data. The Reset command affects the next status byte, which is stored in the FIFO buffer.
- A program should check each byte in which the error/break bit is set to 1 and issue a Reset command each time an error is found. This procedure will minimize the overhead that is associated with error detection.
- This function can be used with the Receive Interrupt function (hex 0F) or with the Combined Interrupts function (hex 10) to determine error status with greater accuracy. When bit 4 (received-data status) of the Enhanced Mode function is set to 1 in the Receive Interrupt function or the Combined Interrupts function, a byte of status is received with each character. If a line-status error occurs, the error/break bit of the received-data status byte of each received character is set to 1 until the Reset Error/Break function is called.



- The possible values of the Return Code field are hex 0000 and 8001.

### Service-Specific Input

Size	Offset	Description
Word	18H	Reserved

### Service-Specific Output

Size	Offset	Description
None		

### 1AH—Return Current Transfer Counts

- This function returns the current transfer counts for transmit operations and receive operations.
- On return, the Return Code field (offset hex 0C) contains the status of the receive transfer counts. The Return Code of Transmit Transfer Count field (offset hex 4F) contains the status of the transmit transfer counts. Both return-code fields should be checked before the Transmit Current Transfer Count field (offset hex 56) and the Receive Current Transfer Count field (offset hex 5A) are read.
- If this function is called with no transmit or receive requests outstanding, the Transmit Current Transfer Count field and the Receive Current Transfer Count field are set to 0 on return.
- The possible values of the Return Code fields are hex 0000, 8001, 8082, and 8083.

### Service-Specific Input

Size	Offset	Description
Word	18H	Reserved

### Service-Specific Output

Size	Offset	Description
Word	4FH	Return code of transmit transfer count
DWord	56H	Transmit current transfer count
DWord	5AH	Receive current transfer count

## **1BH—Set Compare Character Match and Function**

- This function sets the three Compare Character registers and their associated Compare Character functions.
- A Receive Interrupt function (hex 0F) or a Combined Interrupts function (hex 10) with the receive operation enabled should be outstanding before this function is called. If neither of these functions is outstanding, this function writes the Character Compare registers and Character Compare functions to the hardware.
- The three Compare Character Match fields enable programming of receive-match characters in receive operations.
- If a character that is received from the controller matches the value in one of the three enabled Compare Character Match fields, an action is performed as defined by the three Compare Character Function fields.
- The data in a Compare Character Match field is an 8-bit match character. If the word length is less than 8 bits, the match character must be right justified, and any unused bits must be set to 0.
- If null stripping is enabled, the Compare Character Match 3 field is unavailable.
- The three Compare Character Function fields correspond to the three Compare Character Match fields. They enable programming of the controller to start the transmitter, stop the transmitter, delete a match character from an incoming data stream, or to interrupt.
- Multiple Compare Character Match functions per Compare Character function are supported.
- If bit 3 (start transmitter) and bit 2 (stop transmitter) of the Compare Character Function field are both set to 1, bit 2 takes precedence.
- If bit 0 (interrupt) is set to 1, the interrupt occurs as soon as a match with a received data character is detected.
- To disable a Compare Character Match field, set the corresponding Compare Character Function field to 0 before calling the Set Compare Character Match and Function function (hex 1B).
- The possible values of the Return Code field are hex 0000 and 8001.

## Service-Specific Input

Size	Offset	Description
Word	18H	Reserved
Byte	1AH	Compare character match 1
Byte	1BH	Compare character function 1
		Bits 7 to 4 - Reserved
		Bit 3 - Start transmitter on a character match
		= 0 - Transmitter not started on a character match
		= 1 - Transmitter started on a character match
		(the transmitter will not start if bit 2 is set to 1)
		Bit 2 - Stop transmitter on a character match
		= 0 - Transmitter not stopped on a character match
		= 1 - Transmitter stopped on a character match
		(if the transmitter is stopped with a character match, a call to the Start Transmission function (hex 18) can be used to restart the transmitter)
		Bit 1 - Delete character on a character match
		= 0 - Character not deleted on a character match
		= 1 - Character deleted on a character match
		Bit 0 - Interrupt on character match
		= 0 - Interrupt not enabled on a character match
		= 1 - Interrupt enabled on a character match
		(the interrupt occurs as soon as there is a match with a received data character)
Byte	1CH	Compare character match 2
Byte	1DH	Compare character function 2
		(see the description of the Compare Character Function 1 field)
Byte	1EH	Compare character match 3
Byte	1FH	Compare character function 3
		(see the description of the Compare Character Function 1 field)

## Service-Specific Output

Size	Offset	Description
None		

## Return Codes

Return codes are returned at offset hex 0C.

Value	Description
0000H	Operation Successfully Completed
0001H	Incomplete – Stage on Interrupt
0005H	Incomplete – Not My Interrupt, Stage on Interrupt
0009H	Attention, Stage on Interrupt
0081H	Unexpected Interrupt Reset, Stage on Interrupt
8000H	Device Busy, Request Refused
8001H	Async Hardware Level Not Available
8081H	Arbitration Level Not Available
8082H	Arbitration Level Not Allocated
8083H	Arbitration Level Disabled
8084H	DMA Transfer in Progress
8085H	No DMA Transfer in Progress
8086H	DMA Channel Not Available
8087H	Arbitration Level Not Disabled
9000H	Bad Communication Port
C000H	Invalid Logical ID (ABIOS transfer convention only)
C001H	Invalid Function
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length
C020H	No Receive Active in Combined Interrupt

*Figure 6-14. Asynchronous Communication Return Codes*

## Programming Considerations

- Asynchronous communication ABIOS can be initialized and operated in PIO mode without data dependencies other than what can be obtained in ABIOS. For DMA operation, additional data is expected to be returned from system services. The following data is required to support DMA operation:
  - Interrupt level
  - Transmit and receive arbitration levels
  - Base port pairs
  - Enhanced port pairs
  - Dynamic arbitration allocation ports
  - Fixed/dynamic arbitration information
  - Interface description information.

This information is used to initialize the machine state for the asynchronous-communication logical ID that is being initialized. If this information is not available, the serial port is initialized to operate as a character device. Serial channel 1 is initialized to operate on interrupt level 4. All others operate on interrupt level 3.

- **Asynchronous DMA operation also requires the use of the DMA device in the system. During initialization for DMA transfers, information from the DMA device is required. For this information to be available, the DMA device must be initialized before asynchronous communication ABIOS is initialized. If either the receive arbitration level or the transmit arbitration level is disabled for a serial port, ABIOS configures the serial port in the FIFO mode.**

**Notes:**





## Device ID 07H—System Timer

### Functions

The following are the system timer functions. The Default Interrupt Handler function (hex 00) and the Return Logical ID Parameters function (hex 01) are described in "Request Block" in the "Transfer Conventions" section.

**00H—Default Interrupt Handler**

**01H—Return Logical ID Parameters**

**02H—Reserved**

**03H—Read Device Parameters (Reserved)**

**04H—Set Device Parameters (Reserved)**

**05H—Reset/Initialize (Reserved)**

**06H—Enable (Reserved)**

**07H—Disable (Reserved)**

**08H—Read (Reserved)**

**09H—Write (Reserved)**

**0AH—Additional Data Transfer (Reserved)**

### Return Codes

Return codes are returned at offset hex 0C.

Value	Description
0000H	Operation Successfully Completed
0001H	Stage on Interrupt
0005H	Not My Interrupt, Stage on Interrupt
C000H	Invalid Logical ID (ABIOS transfer convention only)
C001H	Invalid Function Number
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length

*Figure 6-15. System Timer Return Codes*

## **Programming Consideration**

The system timer interrupt is handled through the default interrupt handler.



## Device ID 08H—Real-Time Clock

### Functions

The following are the real-time clock functions. The Default Interrupt Handler function (hex 00) and the Return Logical ID Parameters function (hex 01) are described in "Request Block" in the "Transfer Conventions" section.

**Note:** All reserved input fields must be set to 0.

### 00H—Default Interrupt Handler

### 01H—Return Logical ID Parameters

### 02H—Reserved

### 03H—Read Device Parameters

- This function returns the current settings of the real-time clock.
- If the alarm interrupts and periodic interrupts are enabled, this function returns the previously-set values for these interrupts.
- The Periodic Interrupt Rate field is valid only when the periodic interrupt function is enabled.
- The Alarm Hours, Alarm Minutes, and Alarm Seconds fields are valid only when the alarm interrupt function is enabled.
- If the real-time clock is in a clock-update cycle, the Return Code field is set to hex 8000 (Device in Use).
- If the real-time clock is not started, the Return Code field is set to hex 8001 (Real-Time Clock Not Started), and no other output fields are valid.
- The possible values of the Return Code field are hex 0000, 8000, and 8001.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
Byte	10H	Periodic interrupt rate Bits 7 to 4 - Reserved Bits 3 to 0 - Rate value set = 0000 - None = 0001 - 30.517 microseconds = 0010 - 61.035 microseconds = 0011 - 122.070 microseconds = 0100 - 244.141 microseconds = 0101 - 488.281 microseconds = 0110 - 976.562 microseconds = 0111 - 1.953125 milliseconds = 1000 - 3.90625 milliseconds = 1001 - 7.8125 milliseconds = 1010 - 15.625 milliseconds = 1011 - 31.250 milliseconds = 1100 - 62.500 milliseconds = 1101 - 125.00 milliseconds = 1110 - 250.00 milliseconds = 1111 - 500.00 milliseconds
Byte	11H	Real-time-clock status byte Bit 7 - Set bit status = 0 - Clock started = 1 - Clock not started Bit 6 - Periodic interrupt = 0 - Disabled = 1 - Enabled Bit 5 - Alarm interrupt = 0 - Disabled = 1 - Enabled Bit 4 - Update-ended interrupt = 0 - Disabled = 1 - Enabled Bits 3, 2 - Reserved Bit 1 - Clock mode = 0 - 12-hour mode = 1 - 24-hour mode Bit 0 - Reserved
Byte	12H	Alarm hours, in binary coded decimal (00 to 23D)
Byte	13H	Alarm minutes, in binary coded decimal (00 to 59D)
Byte	14H	Alarm seconds, in binary coded decimal (00 to 59D)

### 04H—Set Device Parameters

- This function sets the real-time clock to its different modes.
- The value of the Return Code field is hex 0000.

## Service-Specific Input

Size	Offset	Description
Word	16H	Reserved
Byte	19H	Real-time-clock mode settings
		Bits 7 to 2 - Reserved
		Bit 1 - Hour mode
		= 0 - 12-hour mode
		= 1 - 24-hour mode
		Bit 0 - Reserved

## Service-Specific Output

Size	Offset	Description
None		

**05H—Reset/Initialize (Reserved)**

**06H—Enable (Reserved)**

**07H—Disable (Reserved)**

**08H—Read (Reserved)**

**09H—Write (Reserved)**

**0AH—Additional Data Transfer (Reserved)**

**0BH—Set Alarm Interrupt**

- This function sets the alarm time according to the input values.
- The interrupt is continuous until it is canceled.
- If the interrupt is already enabled when this function is called, the request is refused, and the Return Code field is set to hex 8002 (Interrupt Already Enabled). The request must be canceled by the Cancel Alarm Interrupt function (hex 0C) before it can be restarted.
- If the real-time clock is in a clock-update cycle, the Return Code field is set to hex 8000 (Device in Use).
- If more than one request is returned with the Return Code field set to hex 0001 (Stage on Time), the caller is required to call only the first request. Return code hex 0005 (Not My Interrupt, Stage on Interrupt) indicates that the interrupt is not from the real-time-clock device. Return code hex 0001 (Stage on Interrupt) indicates that the interrupt is from the real-time-clock device but is not necessarily associated with the Request Block function.

- The Interrupt-Pending Status field indicates which interrupts occurred on return from the Interrupt routine when the Return Code field is set to hex 0001 (Stage on Interrupt).
- The real-time clock must be started through the Write Time and Date function (hex 12) before the Set Alarm Interrupt function (hex 0B) is called.
- A single real-time clock interrupt can indicate the occurrence of multiple interrupt types (alarm, periodic, or update-ended interrupt). These interrupt types are activated individually through the Set Alarm Interrupt function (hex 0B), Set Periodic Interrupt function (hex 0D), or Set Update-Ended Interrupt function (hex 0F). When a real-time clock interrupt occurs with multiple interrupt-type requests active, the caller must make a single call with any one of the outstanding request blocks to service the multiple outstanding requests. On return, the Interrupt-Pending Status field indicates which interrupts occurred and were serviced.
- The Cancel Alarm Interrupt function (hex 0C) can be used to cancel an outstanding Set Alarm Interrupt function request and must be called before a previously-set alarm time is changed.
- The possible values of the Return Code field are hex 0001, 0005, 8000, 8001, and 8002.

### Service-Specific Input

Size	Offset	Description
Byte	12H	Alarm hours, in binary coded decimal (00 to 23D)
Byte	13H	Alarm minutes, in binary coded decimal (00 to 59D)
Byte	14H	Alarm seconds, in binary coded decimal (00 to 59D)
DWord	16H	Reserved

### Service-Specific Output

Size	Offset	Description
Byte	1AH	Interrupt-pending status
		Bit 7 - Reserved
		Bit 6 - Periodic interrupt
		= 0 - Disabled
		= 1 - Enabled
		Bit 5 - Alarm interrupt
		= 0 - Disabled
		= 1 - Enabled
		Bit 4 - Update-ended interrupt
		= 0 - Disabled
		= 1 - Enabled
		Bits 3 to 0 - Reserved

**0CH—Cancel Alarm Interrupt**

- This function disables the alarm interrupt.
- The value of the Return Code field is hex 0000.

**Service-Specific Input**

Size	Offset	Description
Word	16H	Reserved

**Service-Specific Output**

Size	Offset	Description
None		

**0DH—Set Periodic Interrupt**

- This function sets the interval for the periodic interrupt.
- The interrupt is continuous until it is canceled.
- If the interrupt is already enabled when this function is called, the request is refused, and the Return Code field is set to hex 8002 (Interrupt Already Enabled). The request must be canceled by the Cancel Alarm Interrupt function (hex 0C) before it can be restarted.
- The real-time clock must be started through the Write Time and Date function (hex 12) before the Set Periodic Interrupt function (hex 0D) is called.
- If more than one request is returned with the Return Code field set to hex 0001 (Stage on Time), the caller is required to call only the first request. Return code hex 0005 (Not My Interrupt, Stage on Interrupt) indicates that the interrupt is not from the real-time-clock device. Return code hex 0001 (Stage on Interrupt) indicates that the interrupt is from the real-time-clock device but is not necessarily associated with the Request Block function.
- The Interrupt-Pending Status field indicates which interrupts occurred on return from the Interrupt routine when the Return Code field is set to hex 0001 (Stage on Interrupt).
- A single real-time clock interrupt can indicate the occurrence of multiple interrupt types (alarm, periodic, or update-ended interrupt). These interrupt types are activated individually through the Set Alarm Interrupt function (hex 0B), Set Periodic Interrupt function (hex 0D), or Set Update-Ended Interrupt function (hex 0F). When a real-time clock interrupt occurs with multiple interrupt-type requests active, the caller must make a single call

with any one of the outstanding request blocks to service the multiple outstanding requests. On return, the Interrupt-Pending Status field indicates which interrupts occurred and were serviced.

- The Cancel Periodic Interrupt function (hex 0E) can be used to cancel an outstanding Set Periodic Interrupt function request and must be called before a previously-set periodic interrupt interval is changed.
- The possible values of the Return Code field are hex 0001, 0005, 8001, and 8002.

### Service-Specific Input

Size	Offset	Description
Byte	10H	Periodic interrupt rate Bits 7 to 4 - Reserved Bits 3 to 0 - Rate value set = 0000 - None = 0001 - 30.517 microseconds = 0010 - 61.035 microseconds = 0011 - 122.070 microseconds = 0100 - 244.141 microseconds = 0101 - 488.281 microseconds = 0110 - 976.562 microseconds = 0111 - 1.953125 milliseconds = 1000 - 3.90625 milliseconds = 1001 - 7.8125 milliseconds = 1010 - 15.625 milliseconds = 1011 - 31.250 milliseconds = 1100 - 62.500 milliseconds = 1101 - 125.00 milliseconds = 1110 - 250.00 milliseconds = 1111 - 500.00 milliseconds
DWord	16H	Reserved

### Service-Specific Output

Size	Offset	Description
Byte	1AH	Interrupt-pending status Bit 7 - Reserved Bit 6 - Periodic interrupt = 0 - Disabled = 1 - Enabled Bit 5 - Alarm interrupt = 0 - Disabled = 1 - Enabled Bit 4 - Update-ended interrupt = 0 - Disabled = 1 - Enabled Bits 3 to 0 - Reserved



## 0EH—Cancel Periodic Interrupt

- This function disables the periodic interrupt.
- The value of the Return Code field is hex 0000.

## Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
None		

## 0FH—Set Update Ended Interrupt

- This function enables the Update Ended interrupt.
- The interrupt is continuous until it is canceled.
- If the interrupt is already enabled when this function is called, the request is refused, and the Return Code field is set to hex 8002 (Interrupt Already Enabled). The request must be canceled by the Cancel Alarm Interrupt function (hex 0C) before it can be restarted.
- The real-time clock must be started through the Write Time and Date function (hex 12) before the Set Update-Ended Interrupt function (hex 0D) is called.
- If more than one request is returned with the Return Code field set to hex 0001 (Stage on Time), the caller is required to call only the first request. Return code hex 0005 (Not My Interrupt, Stage on Interrupt) indicates that the interrupt is not from the real-time-clock device. Return code hex 0001 (Stage on Interrupt) indicates that the interrupt is from the real-time-clock device but is not necessarily associated with the Request Block function.
- The Interrupt-Pending Status field indicates which interrupts occurred on return from the Interrupt routine when the Return Code field is set to hex 0001 (Stage on Interrupt).
- A single real-time clock interrupt can indicate the occurrence of multiple interrupt types (alarm, periodic, or update-ended interrupt). These interrupt types are activated individually through the Set Alarm Interrupt function (hex 0B), Set Periodic Interrupt function (hex 0D), or Set Update-Ended Interrupt function (hex 0F). When a real-time clock interrupt occurs with multiple interrupt-type requests active, the caller must make a single call

with any one of the outstanding request blocks to service the multiple outstanding requests. On return, the Interrupt-Pending Status field indicates which interrupts occurred and were serviced.

- The Cancel Update Ended Interrupt function (hex 10) can be used to cancel an outstanding Set Update Ended Interrupt function request and must be called before an Update Ended interrupt is restarted.
- The possible values of the Return Code field are hex 0001, 0005, 8001, and 8002.

### Service-Specific Input

Size	Offset	Description
DWord	16H	Reserved

### Service-Specific Output

Size	Offset	Description
Byte	1AH	Interrupt-pending status
		Bit 7 - Reserved
		Bit 6 - Periodic interrupt
		= 0 - Disabled
		= 1 - Enabled
		Bit 5 - Alarm interrupt
		= 0 - Disabled
		= 1 - Enabled
		Bit 4 - Update-ended interrupt
		= 0 - Disabled
		= 1 - Enabled
		Bits 3 to 0 - Reserved

### 10H—Cancel Update Ended Interrupt

- This function disables the Update Ended interrupt.
- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

### Service-Specific Output

Size	Offset	Description
None		

## 11H—Read Time and Date

- This function reads the current setting of the real-time clock.
- The real-time clock must be started through the Write Time and Date function (hex 12) before the Read Time and Date function (hex 11) is called.
- The Time and Date fields are valid only when the Return Code field is set to hex 0000 (Operation Successfully Completed).
- The possible values of the Return Code field are hex 0000, 8000, and 8001.

### Service-Specific Input

Size	Offset	Description
Word	10H	Reserved

### Service-Specific Output

Size	Offset	Description
Byte	12H	Hours, in binary coded decimal (00 to 23D)
Byte	13H	Minutes, in binary coded decimal (00 to 59D)
Byte	14H	Seconds, in binary coded decimal (00 to 59D)
Byte	15H	Century, in binary coded decimal (19D or 20D)
Byte	16H	Year, in binary coded decimal (00 to 99D)
Byte	17H	Month, in binary coded decimal (01D to 12D)
Byte	18H	Day, in binary coded decimal (01D to 31D)

## 12H—Write Time and Date

- This function starts the clock, if it is not already started, and sets the time and date information according to the input parameters.
- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
Byte	12H	Hours, in binary coded decimal (00 to 23D)
Byte	13H	Minutes, in binary coded decimal (00 to 59D)
Byte	14H	Seconds, in binary coded decimal (00 to 59D)
Byte	15H	Century, in binary coded decimal (19D or 20D)
Byte	16H	Year, in binary coded decimal (00 to 99D)
Byte	17H	Month, in binary coded decimal (01D to 12D)
Byte	18H	Day, in binary coded decimal (01D to 31D)

### Service-Specific Output

Size	Offset	Description
None		

## Return Codes

Return codes are returned at offset hex 0C.

Value	Description
0000H	Operation Successfully Completed
0001H	Stage on Interrupt
0005H	Not My Interrupt, Stage on Interrupt
8000H	Device in Use
8001H	Real-Time Clock Not Started
8002H	Interrupt Already Enabled
C000H	Invalid Logical ID (ABIOS transfer convention only)
C001H	Invalid Function Number
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length

*Figure 6-16. Real-Time Clock Return Codes*

---

## Device ID 09H—System Services

### Functions

The following are the system services functions. The Default Interrupt Handler function (hex 00) and the Return Logical ID Parameters function (hex 01) are described in "Request Block" in the "Transfer Conventions" section.

**Note:** All reserved input fields must be set to 0.

**00H—Default Interrupt Handler**

**01H—Return Logical ID Parameters**

**02H—Reserved**

**03H—Used Internally by BIOS**

**04H—Set Device Parameters (Reserved)**

**05H—Reset/Initialize (Reserved)**

**06H—Enable (Reserved)**

**07H—Disable (Reserved)**

**08H—Read (Reserved)**

**09H—Write (Reserved)**

**0AH—Additional Data Transfer (Reserved)**

**0BH—Switch to Real Mode**

- This function switches the processor into real mode and disables address line 20.
- All interrupts, including the nonmaskable interrupt (NMI), are disabled. BIOS gives control back to the caller at the location pointed to by the Resume Pointer field. The caller must reenables all interrupts, including the NMI.

- For 80386, 80386 SX, 80486, and 80486 SX systems, the selector to a Dummy Descriptor field in the caller's global descriptor table must have its segment limit set to the maximum (hex 0FFFF). The base address can be any value, and the access-rights byte is set to:
  - Expand upward (E=0)
  - Writable (W=1)
  - Present (P=1).
- For all calls to this function, all registers must be saved and restored in the proper mode. A call to this function modifies all registers, including the FS and GS segment registers, in an 80386-, 80386 SX-, 80486-, or 80486 SX-compatible environment. Other BIOS services depend on the value of the FS and GS segment registers. An 80286 operating system that uses BIOS must dynamically determine the presence of the 80386 or compatible processor and save and restore the FS and GS segment registers.
- The Return Code field is not updated in this function unless a parameter error occurs. The AH register should be set to a value other than 0 so that it can be checked on return.

### Service-Specific Input

Size	Offset	Description
DWord	10H	Reserved
DWord	60H	Resume pointer
Word	64H	Selector to a dummy descriptor field

### Service-Specific Output

Size	Offset	Description
None		

### 0CH—Used Internally by BIOS

### 0DH—Enable Address Line 20

- The possible value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
DWord	10H	Reserved

## Service-Specific Output

Size	Offset	Description
None		

### 0EH—Disable Address Line 20

- The possible value of the Return Code field is hex 0000.

## Service-Specific Input

Size	Offset	Description
DWord	10H	Reserved

## Service-Specific Output

Size	Offset	Description
None		

### 0FH—Speaker

- This function enables the system speaker with the input frequency and duration.
- If the value of the Frequency Divisor field is 0, or the value of the Duration Counter field is 0, no action is performed, and the Return Code field is set to hex 0000 (Operation Successfully Completed).
- The possible value of the Return Code field is hex 0000.

## Service-Specific Input

Size	Offset	Description
DWord	10H	Reserved
Word	60H	Frequency divisor (1.19 MHz/freq. div. = frequency freq. div. = 1331 for 886 Hz freq.)
Byte	66H	Duration counter, in 1/64 seconds

## Service-Specific Output

Size	Offset	Description
None		

### 13H—Processor Functions

- This function enables or disables processor functions for testing and recovery.
- The possible values of the Return Code field are hex 0000, 80FE, 9000, 9001, 9002, and C001.

## Service-Specific Input

**Note:** For an 80486 processor, disable any external caches when the on-chip cache (L1) is disabled.

Size	Offset	Description
Word	10H	Reserved
Byte	12H	Reserved
Byte	13H	Reserved
Word	14H	Processor functions
		= 0001H - Disable L1 cache
		= 0002H - Enable L1 cache
		= 0003H - Disable L2 cache
		= 0004H - Enable L2 cache
		= 0005H - Disable both caches
		= 0006H - Enable both caches
		= 0007H - Status of both caches
		= 0008H to FFFFH - Reserved

## Service-Specific Output

Size	Offset	Description
Word	0CH	Return status
Byte	10H	Status of L1 cache, returned for function code hex 0007; the value is one of the following: = 00H - L1 cache is enabled = 01H - L1 cache is disabled = 02H - L1 cache is disabled because of a cache test error; the cache cannot be enabled
Byte	11H	Status of L2 cache, returned for function code hex 0007; the value is one of the following: = 00H - L2 cache is enabled = 01H - L2 cache is disabled = 02H - L2 cache is disabled because of a cache test error; the cache cannot be enabled

## Return Codes

Return codes are returned at offset hex 0C.

Value	Description
0000H	Operation Successfully Completed
80FEH	Bad NVRAM
9000H	Cannot Perform Operation Because of State of Other Level of Cache
9001H	Cache Test Error
9002H	No Level 2 Cache Present
C000H	Invalid Logical ID (ABIOS transfer convention only)
C001H	Invalid Function Number
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length

**Figure 6-17.** System Services Return Codes



## Device ID 0AH—Nonmaskable Interrupt (NMI)

The nonmaskable interrupt (NMI) handler is used to process severe errors. In most cases, the fault is in the hardware, but it is also possible for a software error to force an NMI to occur.

### Functions

The following are the nonmaskable interrupt (NMI) functions. The Default Interrupt Handler function (hex 00) and the Return Logical ID Parameters function (hex 01) are described in "Request Block" in the "Transfer Conventions" section.

**Note:** All reserved input fields must be set to 0.

**00H—Default Interrupt Handler (Reserved)**

**01H—Return Logical ID Parameters**

**02H—Reserved**

**03H—Read Device Parameters (Reserved)**

**04H—Set Device Parameters (Reserved)**

**05H—Reset/Initialize (Reserved)**

**06H—Enable**

- This function unmaskes the NMI.
- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Word	21H	Reserved

### Service-Specific Output

Size	Offset	Description
None		

### 07H—Disable

- This function disables the NMI for memory parity and I/O channel checks.
- The DMA bus time-out NMI and the watchdog time-out NMI cannot be disabled through BIOS.
- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Word	21H	Reserved

### Service-Specific Output

Size	Offset	Description
None		

### 08H—Continuous Read

- This function starts the monitoring of NMI status.
- If the NMI is caused by a direct memory access (DMA) arbitration bus time-out, the arbitration level that caused the time-out is returned in the request block.
- If the NMI is caused by a channel check, the slot number of the adapter that caused the NMI is returned in the request block.
- When the update of this request block is completed, the NMI is disabled. The caller should reenable the NMI through the Enable function (hex 06) if nonmaskable interrupts are desired.
- The possible values of the Return Code field are hex 0001, 0005, and 0009. Return code hex 0009 (Attention, Stage on Interrupt) indicates that an update has occurred and needs to be handled. In preparation for the next NMI, the caller should change the value of the Return Code field from hex 0009 to hex 0001 (Stage on Interrupt) after the request block has been reviewed.

### Service-Specific Input

Size	Offset	Description
Word	21H	Reserved

## Service-Specific Output

Size	Offset	Description
Word	10H	Type of NMI = 00H - Reserved = 01H - Parity = 02H - Channel check = 03H - DMA bus time-out = 04H - Watchdog time-out = 05H to FFFFH - Reserved
Byte	1EH	DMA arbitration level that caused the DMA bus time-out
Byte	1FH	Slot number that caused the I/O channel check

### 09H—Write (Reserved)

### 0AH—Additional Data Transfer (Reserved)

## Return Codes

Return codes are returned at offset hex 0C.

Value	Description
0000H	Operation Successfully Completed
0001H	Stage on Interrupt
0005H	Not My Interrupt, Stage on Interrupt
0009H	Attention, Stage on Interrupt
C000H	Invalid Logical ID (ABIOS transfer convention only)
C001H	Invalid Function Number
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length

**Figure 6-18. Nonmaskable Interrupt Return Codes**

**Notes:**



## Device ID 0BH—Pointing Device

### Functions

The following are the pointing device functions. The Default Interrupt Handler function (hex 00) and the Return Logical ID Parameters function (hex 01) are described in "Request Block" in the "Transfer Conventions" section.

**Note:** All reserved input fields must be set to 0.

### 00H—Default Interrupt Handler

### 01H—Return Logical ID Parameters

### 02H—Reserved

### 03H—Read Device Parameters

- This function returns the current pointing-device status and package-size setting.
- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Word	1CH	Reserved

### Service-Specific Output

Size	Offset	Description
Byte	10H	Interface status Bits 7 to 6 - Reserved Bit 5 - Interface enablement = 0 - Disabled = 1 - Enabled Bits 4 to 0 - Reserved
Byte	11H	Data-package size = 00H - Reserved = 01H - 1 byte = 02H - 2 bytes = 03H - 3 bytes = 04H - 4 bytes = 05H - 5 bytes = 06H - 6 bytes = 07H - 7 bytes = 08H - 8 bytes = 09H to FFH - Reserved

Size	Offset	Description
Word	12H	Current pointing-device status Bits 15 to 7 - Reserved Bit 6 - Mode = 0 - Stream mode = 1 - Remote/poll mode Bit 5 - Status = 0 - Disabled = 1 - Enabled Bit 4 - Scaling = 0 - Scaling set to 1:1 = 1 - Scaling set to 2:1 Bit 3 - Reserved Bit 2 - Left-button status = 0 - Not pressed = 1 - Pressed Bit 1 - Reserved Bit 0 - Right-button status = 0 - Not pressed = 1 - Pressed
Word	14H	Current resolution = 00H - 1 count per millimeter = 01H - 2 counts per millimeter = 02H - 4 counts per millimeter = 03H - 8 counts per millimeter = 04H to FFFFH - Reserved
Word	16H	Current sample rate = 0AH - 10 reports per second = 14H - 20 reports per second = 28H - 40 reports per second = 3CH - 60 reports per second = 50H - 80 reports per second = 64H - 100 reports per second = C8H - 200 reports per second
DWord	18H	Time to wait before resuming request, in microseconds

## 04H—Set Device Parameters (Reserved)

## 05H—Reset/Initialize Pointing Device

- This function resets the pointing device. On completion of this function, the pointing device is initialized as follows:
  - The package size remains unchanged.
  - Resolution = 4 counts per millimeter.
  - Sample rate = 100 reports per second.
  - Scaling = 1:1.
  - The pointing device is disabled.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 0005, 8000, 8001, 8002, and 8003.



### Service-Specific Input

Size	Offset	Description
Word	1CH	Reserved

### Service-Specific Output

Size	Offset	Description
Byte	10H	Pointing-device completion code
Byte	11H	Pointing-device identification code
DWord	18H	Time to wait before resuming request, in microseconds

### 06H—Enable

- This function enables the pointing device.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 0005, 8000, 8001, 8002, and 8003.

### Service-Specific Input

Size	Offset	Description
Word	1CH	Reserved



### Service-Specific Output

Size	Offset	Description
DWord	18H	Time to wait before resuming request, in microseconds

### 07H—Disable

- This function disables the pointing device.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 0005, 8000, 8001, 8002, and 8003.

### Service-Specific Input

Size	Offset	Description
Word	1CH	Reserved

### Service-Specific Output

Size	Offset	Description
DWord	18H	Time to wait before resuming request, in microseconds



## 08H—Read

- This function is called to read the pointing device; it leaves the pointing device disabled. This function must be called before any other pointing-device function except the Return Logical ID Parameters function (hex 01). After the Read function (hex 08) has been called, the pointing device can be enabled through the Enable function (hex 06).
- The possible values of the Return Code field are hex 0001, 0002, 0005, 0009, 8000, 8003, and C005.

## Service-Specific Input

Size	Offset	Description
Byte	10H	Package size = 00H - Reserved = 01H - 1 byte = 02H - 2 bytes = 03H - 3 bytes = 04H - 4 bytes = 05H - 5 bytes = 06H - 6 bytes = 07H - 7 bytes = 08H - 8 bytes = 09H to FFH - Reserved
DWord	12H	Reserved

## Service-Specific Output

Size	Offset	Description
Byte	1CH	First byte from the pointing device – status Bit 7 - Y-data overflow = 0 - No overflow = 1 - Overflow Bit 6 - X-data overflow = 0 - No overflow = 1 - Overflow Bit 5 - Y-data sign = 0 - Positive = 1 - Negative Bit 4 - X-data sign = 0 - Positive = 1 - Negative Bit 3 - Reserved (set to 1) Bit 2 - Reserved (set to 0) Bit 1 - Right-button status = 0 - Not pressed = 1 - Pressed Bit 0 - Left-button status = 0 - Not pressed = 1 - Pressed
Byte	1DH	Second byte from the pointing device – X data
Byte	1EH	Third byte from the pointing device – Y data



Size	Offset	Description
5 bytes	1FH to 23H	Fourth byte to eighth byte from the pointing device (raw data)
4 bytes	24H to 27H	Ninth byte to twelfth byte from the pointing device (reserved)

### 09H—Write (Reserved)

### 0AH—Additional Data Transfer (Reserved)

### 0BH—Set Sample Rate

- This function sets the pointing-device sample rate.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 0005, 8000, 8001, 8002, and 8003.

### Service-Specific Input

Size	Offset	Description
Word	12H	Sample rate
		= 0AH - 10 reports per second
		= 14H - 20 reports per second
		= 28H - 40 reports per second
		= 3CH - 60 reports per second
		= 50H - 80 reports per second
		= 64H - 100 reports per second
		= C8H - 200 reports per second

### Service-Specific Output

Size	Offset	Description
DWord	18H	Time to wait before resuming request, in microseconds

### 0CH—Set Resolution

- This function sets the pointing-device resolution.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 0005, 8000, 8001, 8002, and 8003.

### Service Specific Input

Size	Offset	Description
Word	12H	Resolution
		= 00H - 1 count per millimeter
		= 01H - 2 counts per millimeter
		= 02H - 4 counts per millimeter
		= 03H - 8 counts per millimeter
		= 04H to FFFFH - Reserved
Word	1CH	Reserved

### Service-Specific Output

Size	Offset	Description
DWord	18H	Time to wait before resuming request, in microseconds

### 0DH—Set Scaling

- This function sets the pointing-device scaling value.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 0005, 8000, 8001, 8002, and 8003.

### Service-Specific Input

Size	Offset	Description
Byte	10H	Scaling value = 00H - Reserved = 01H - Set scaling to 1:1 = 02H - Set scaling to 2:1 = 03H to FFH - Reserved
Word	1CH	Reserved

### Service-Specific Output

Size	Offset	Description
DWord	18H	Time to wait before resuming request, in microseconds

### 0EH—Read Pointing-Device Identification Code

- This function returns the pointing-device identification code.
- The possible values of the Return Code field are hex 0000, 0001, 0002, 0005, 8000, 8001, 8002, and 8003.

### Service-Specific Input

Size	Offset	Description
Word	1CH	Reserved

### Service-Specific Output

Size	Offset	Description
Byte	10H	Pointing-device identification code
DWord	18H	Time to wait before resuming request, in microseconds

## Return Codes

Return codes are returned at offset hex 0C.

Value	Description
0000H	Operation Successfully Completed
0001H	Stage On Interrupt
0002H	Stage On Time
0005H	Not My Interrupt, Stage on Interrupt
0009H	Attention, Stage on Interrupt, Data Available
8000H	Device in Use
8001H	Resend
8002H	Two Consecutive Resends Found
8003H	System Lock
8004H	Nonacknowledgment Response
9100H	Controller Failure
C000H	Invalid Logical ID (ABIOS transfer convention only)
C001H	Invalid Function Number
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length
C005H	Invalid Pointing-Device Parameter

*Figure 6-19. Pointing Device Return Codes*

## Programming Consideration

The Read Pointing-Device Identification Code function (hex 0E) returns the device ID that is read from the auxiliary device interface. The IBM pointing device ID is hex 00.

## Notes:

## Device ID 0EH—Nonvolatile Random Access Memory (NVRAM)

### Functions

The following are the NVRAM functions. The Default Interrupt Handler function (hex 00) and the Return Logical ID Parameters function (hex 01) are described in “Request Block” in the “Transfer Conventions” section.

**Note:** All reserved input fields must be set to 0.

**00H—Default Interrupt Handler**

**01H—Return Logical ID Parameters**

**02H—Reserved**

**03H—Read Device Parameters (Reserved)**

**04H—Set Device Parameters (Reserved)**

**05H—Reset/Initialize (Reserved)**

**06H—Enable (Reserved)**

**07H—Disable (Reserved)**

**08H—Read NVRAM**

- This function returns the data that is currently stored in the specified data buffer of the specified RAM (64-byte RAM or extended RAM).
- If the value of the Number of Bytes to Be Transferred field is 0, no action is performed, and the Return Code field is set to hex 0000 (Operation Successfully Completed).
- If the value of the Number of Bytes to Be Transferred field plus the value of the Starting RAM Address field is greater than the maximum amount of RAM, no action is performed, and the Return Code field is set to hex C005 (Invalid NVRAM Parameter).
- Because the Read NVRAM function (hex 08) allows access to real-time-clock data, the real-time clock must be checked to determine whether it is in a clock-update cycle before it can

return valid data. If the real-time clock is in a clock-update cycle, the Return Code field is set to hex 9101 (Retryable Device Error).

- The possible values of the Return Code field are hex 0000, 80FE, 80FF, 9101, C004, and C005.

### Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Pointer to data buffer
Word	16H	Reserved
DWord	1AH	Reserved
Word	20H	Flag word
		Bit 15 - NMI state on exit
		= 0 - NMI enabled
		= 1 - NMI disabled
		Bits 14 to 1 - Reserved
		Bit 0 - RAM type
		= 0 - 64-byte RAM
		= 1 - Extended RAM
Word	22H	Starting RAM address
Word	24H	Number of bytes to be transferred

### Service-Specific Output

Size	Offset	Description
None		

### 09H—Write NVRAM

- This function writes the data that is in the specified data buffer to the specified location of the specified RAM (64-byte RAM or extended RAM).
- If the value of the Number of Bytes to Be Transferred field is 0, no action is performed, and the Return Code field is set to hex 0000 (Operation Successfully Completed).
- If the value of the Number of Bytes to Be Transferred field plus the value of the Starting RAM Address field is greater than the maximum amount of RAM, no action is performed, and the Return Code field is set to hex C005 (Invalid NVRAM Parameter).
- Because the Write NVRAM function (hex 09) allows access to real-time clock data, the real-time clock must be checked to determine whether it is in a clock-update cycle before it can return valid data. If the real-time clock is in a clock-update cycle, the Return Code field is set to hex 9101 (Retryable Device Error).
- The possible values of the Return Code field are hex 0000, 80FE, 80FF, 9101, C004, and C005.

## Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Pointer to data buffer
Word	16H	Reserved
DWord	1AH	Reserved
Word	20H	Flag word
		Bit 15 - NMI state on exit
		= 0 - NMI enabled
		= 1 - NMI disabled
		Bits 14 to 1 - Reserved
		Bit 0 - RAM type
		= 0 - 64-byte RAM
		= 1 - Extended RAM
Word	22H	Starting RAM address
Word	24H	Number of bytes to be transferred

## Service-Specific Output

Size	Offset	Description
None		

### 0AH—Additional Data Transfer Function (Reserved)

### 0BH—Recompute Checksum

- This function recomputes the checksum for the specified RAM (64-byte or extended RAM).
- The possible values of the Return Code field are hex 0000, 80FE, C004, and 80FF.

## Service-Specific Input

Size	Offset	Description
Word	16H	Reserved
Word	20H	Flag word
		Bit 15 - NMI state on exit
		= 0 - NMI enabled
		= 1 - NMI disabled
		Bits 14 to 1 - Reserved
		Bit 0 - RAM type
		= 0 - 64-byte RAM
		= 1 - Extended RAM

## Service-Specific Output

Size	Offset	Description
None		

## Return Codes

Return codes are returned at offset hex 0C.

Value	Description
0000H	Operation Successfully Completed
80FEH	NVRAM Checksum Invalid
80FFH	NVRAM Battery Bad
9101H	Retryable Device Error
C000H	Invalid Logical ID (ABIOS transfer convention only)
C001H	Invalid Function Number
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length
C005H	Invalid NVRAM Parameter

**Figure 6-20. Nonvolatile Random Access Memory (NVRAM) Return Codes**



## Device ID 0FH—Direct Memory Access (DMA)

### Functions

The following are the direct memory access (DMA) functions. The Return Logical ID Parameters function (hex 01) is described in “Request Block” in the “Transfer Conventions” section.

**Note:** All reserved input fields must be set to 0.

### 00H—Reserved

### 01H—Return Logical ID Parameters

### 02H—Reserved

### 03H—Read Device Parameters

- This function returns the DMA device parameters.
- The DMA functional enhancements that are provided when the functional-support bit in the Parameter Flags field is set to 1 are as follows:
  - The Allocate Arbitration Level function (hex 0B) can allocate an arbitration level with or without allocating a channel.
  - Bits 0 and 1 of the Abort Arbitration Level Information field (offset hex 16) of the Abort Arbitration Level function (hex 0F) indicates whether an arbitration level is to be aborted, disabled, and deallocated; whether an arbitration level is to be aborted or disabled; or whether an arbitration level is only to be aborted.
  - The Read from Memory and Write to I/O function (hex 10), the Read from I/O and Write to Memory function (hex 11), and the Verify function (hex 12) each have 2 bits that select whether to search for an available arbitration level on which to perform the transfer and at what priority level the search is to start.
  - The bit numbers in the bit maps correspond to arbitration levels or channels. For example, bit 0 at offset hex 18 corresponds to arbitration level 0; bit 0 at offset hex 1C corresponds to DMA channel 0. If a bit in the Bit Map of Virtual Arbitration Levels field (offset hex 18) is set to 1, the corresponding arbitration level is programmable. If a bit in the Bit Map of Virtual DMA Channels field (offset hex 1C) is set to 1, the corresponding DMA channel is programmable.

- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

### Service-Specific Output

Size	Offset	Description
Word	10H	Maximum address, in 1MB (the actual value is 1 byte less than this value)
Word	12H	Maximum direct memory access transfer size, in 1KB
Byte	14H	Number of arbitration levels
Byte	15H	Number of direct memory access channels
Byte	16H	Parameter flags <ul style="list-style-type: none"> <li>Bits 7 to 1 - Reserved</li> <li>Bit 0 - Functional support <ul style="list-style-type: none"> <li>= 0 - DMA functional enhancements not supported</li> <li>= 1 - DMA functional enhancements supported</li> </ul> </li> </ul>
DWord	18H	Bit map of virtual (programmable) arbitration levels
DWord	1CH	Bit map of virtual (programmable) DMA channels

### 04H—Set Device Parameters (Reserved)

### 05H—Reset/Initialize (Reserved)

### 06H—Enable for Interrupts (Reserved)

### 07H—Disable for Interrupts (Reserved)

### 08H—Read (Reserved)

### 09H—Write (Reserved)

### 0AH—Additional Data Transfer (Reserved)

### 0BH—Allocate Arbitration Level

- This function allocates an arbitration level.
- Bit 0 of the Allocate Arbitration Bit Level Information field determines whether a channel is to be allocated for the arbitration level. This is useful for bus masters that need an arbitration level but not a channel.
- The possible values of the Return Code field are hex 0000, 8001, 8006, and C006.

## Service-Specific Input

Size	Offset	Description
Byte	16H	Allocate arbitration bit level information Bits 7 to 1 - Reserved Bit 0 - Channel allocation = 0 - Allocate channel for requested arbitration level = 1 - No channel needed for this arbitration level
Byte	17H	Reserved
Byte	1FH	Arbitration level to be allocated

## Service-Specific Output

Size	Offset	Description
None		

### 0CH—Deallocate Arbitration Level

- This function makes available a previously-allocated arbitration level. It also deallocates the channel that is associated with the arbitration level, if one had been allocated.
- The possible values of the Return Code field are hex 0000, 8002, 8004, 8007, and C006.

## Service-Specific Input

Size	Offset	Description
Word	16H	Reserved
Byte	1FH	Arbitration level to be deallocated

## Service-Specific Output

Size	Offset	Description
None		

**0DH—Disable Arbitration Level**

- This function disables the arbitration level in the DMA controller.
- The possible values of the Return Code field are hex 0000, 8002, 8004, and C006.

**Service-Specific Input**

Size	Offset	Description
Word	16H	Reserved
Byte	1FH	Arbitration level to be disabled

**Service-Specific Output**

Size	Offset	Description
None		

**0EH—Transfer Status**

- This function returns the number of transfers that are left to be completed, as read from the DMA controller.
- The possible values of the Return Code field are hex 0000, 8002, 8003, and C006.

**Service-Specific Input**

Size	Offset	Description
Word	16H	Reserved
Byte	1FH	Arbitration level to be checked

**Service-Specific Output**

Size	Offset	Description
DWord	18H	Number of transfers left

## 0FH—Abort

- This function accesses the DMA controller to disable the channel and to read both the physical address and the number of DMA transfers that were left when the Abort function was issued.
- The Abort function has additional capabilities available through the Abort Arbitration Level Information field (offset hex 16). If bit 1 is set to 1, the transfer is aborted, the channel is disabled, and the arbitration level is deallocated. If bit 1 is set to 0 and bit 0 is set to 1, the transfer is aborted, and the channel is disabled. If both bit 1 and bit 0 are set to 0, the transfer that is in progress is aborted. The physical address and the number of transfers that remained when the transfer was aborted are returned.
- The possible values of the Return Code field are hex 0000, 8002, 8003, 8005, and C006.

## Service-Specific Input

Size	Offset	Description
Byte	16H	Abort arbitration level information <ul style="list-style-type: none"><li>Bits 7 to 2 - Reserved</li><li>Bit 1 - Abort, disable, and deallocate arbitration level<ul style="list-style-type: none"><li>= 0 - Do not deallocate the arbitration level; refer to bit 0 for channel action</li><li>= 1 - Abort transfer, disable channel, and deallocate arbitration level (not dependent on bit 0)</li></ul></li><li>Bit 0 - Abort or disable channel<ul style="list-style-type: none"><li>= 0 - Abort transfer that is in progress</li><li>= 1 - Abort or disable with no contingency on a transfer being in progress</li></ul></li></ul>
Byte	17H	Reserved
Byte	1FH	Arbitration level on which to abort operation

## Service-Specific Output

Size	Offset	Description
DWord	10H	Physical address when abort was issued
DWord	18H	Number of transfers remaining when abort was issued

## 10H—Read from Memory and Write to I/O

- This function starts a DMA transfer from memory to an I/O device by programming the DMA controller with the indicated values from the request block.
- The Number of DMA Transfers field specifies the number of transfers that are to be executed by the DMA channel. The number of transfers is always 1 more than the specified count. For example, if the count is 0, one transfer is executed.

- The Mode Control field provides the following functions:
  - Bit 3 is used only if bit 1 is set to 1. Bit 3 determines whether the search for an available arbitration level starts with the lowest-priority or the highest-priority arbitration level. When bit 3 is set to 0, the search starts with the highest priority.
  - Bit 2 indicates that the I/O address will be programmed to the DMA controller. When bit 2 is set to 1, the DMA controller drives the I/O address that is specified at offset hex 14 of the request block onto the address bus during the DMA cycles, instead of driving an I/O address of 0.
  - Bit 1 specifies whether to use the arbitration level that is already allocated or to search for an available level. When bit 1 is set to 1, BIOS searches for an available level, starts the transfer, and returns the allocated arbitration level at offset hex 1F. When bit 1 is set to 0, BIOS uses the arbitration level that is specified at offset hex 1F. Bit 3 determines whether the search starts with the highest priority or the lowest priority.
  - Bit 0 causes the DMA controller to automatically initialize the transfer count and memory address to their previously-programmed values after the terminal count is reached.
- The Transfer Control field provides the following functions:
  - Bits 1 and 0 indicate whether the transfer size is 8 bits or 16 bits. The transfer-count size that is returned is in bytes if the device size is 8 bits; it is in words if the device size is 16 bits.
  - Bit 2 specifies whether the physical address of memory is increased or decreased during the transfer.
- The possible values of the Return Code field are hex 0000, 8001, 8002, 8004, 8006, C005, and C006.

## Service-Specific Input

Size	Offset	Description
DWord	10H	Physical address of memory
DWord	14H	Physical address of I/O
DWord	18H	Number of DMA data transfers
Byte	1CH	Mode control
		Bits 7 to 4 - Reserved
		Bit 3 = 1 - Start search with highest priority
		Bit 2 = 1 - Program I/O address
		Bit 1 = 1 - Allocate any arbitration level for transfer
		Bit 0 = 1 - Autoinitialization

Size	Offset	Description
Byte	1DH	Transfer control Bits 7 to 3 - Reserved Bit 2 - Count control = 0 - Increase = 1 - Decrease Bits 1, 0 - Device size = 00 - 8 bits = 01 - 16 bits = 10 - Reserved = 11 - Reserved
Byte	1FH	Arbitration level to be used; input only if bit 1 of the Mode Control field is set to 0

## Service-Specific Output

Size	Offset	Description
Byte	1FH	Allocated arbitration level

## 11H—Read from I/O and Write to Memory

- This function starts a DMA transfer from an I/O device to memory by programming the DMA controller with the indicated values from the request block.
- The Number of DMA Transfers field specifies the number of transfers that are to be executed by the DMA channel. The number of transfers is always 1 more than the specified count. For example, if the count is 0, one transfer is executed.
- The Mode Control field provides the following functions:
  - Bit 3 is used only if bit 1 is set to 1. Bit 3 determines whether the search for an available arbitration level starts with the lowest-priority or the highest-priority arbitration level. When bit 3 is set to 0, the search starts with the highest priority.
  - Bit 2 indicates that the I/O address will be programmed to the DMA controller. When bit 2 is set to 1, the DMA controller drives the I/O address that is specified at offset hex 14 of the request block onto the address bus during the DMA cycles, instead of driving an I/O address of 0.
  - Bit 1 specifies whether to use the arbitration level that is already allocated or to search for an available level. When bit 1 is set to 1, BIOS searches for an available level, starts the transfer, and returns the allocated arbitration level at offset hex 1F. When bit 1 is set to 0, BIOS uses the arbitration level that is specified at offset hex 1F. Bit 3 determines whether the search starts with the highest priority or the lowest priority.

- Bit 0 causes the DMA controller to automatically initialize the transfer count and memory address to their previously-programmed values after the terminal count is reached.
- The Transfer Control field provides the following functions:
  - Bits 1 and 0 indicate whether the transfer size is 8 bits or 16 bits. The transfer-count size that is returned is in bytes if the device size is 8 bits; it is in words if the device size is 16 bits.
  - Bit 2 specifies whether the physical address of memory is increased or decreased during the transfer.
- The possible values of the Return Code field are hex 0000, 8001, 8002, 8004, 8006, C005, and C006.

### Service-Specific Input

Size	Offset	Description
DWord	10H	Physical address of memory
DWord	14H	Physical address of I/O
DWord	18H	Number of DMA data transfers
Byte	1CH	Mode control <ul style="list-style-type: none"> <li>Bits 7 to 4 - Reserved</li> <li>Bit 3 = 1 - Start search with highest priority</li> <li>Bit 2 = 1 - Program I/O address</li> <li>Bit 1 = 1 - Allocate any arbitration level for transfer</li> <li>Bit 0 = 1 - AutoInitialization</li> </ul>
Byte	1DH	Transfer control <ul style="list-style-type: none"> <li>Bits 7 to 3 - Reserved</li> <li>Bit 2 - Count control               <ul style="list-style-type: none"> <li>= 0 - Increase</li> <li>= 1 - Decrease</li> </ul> </li> <li>Bits 1, 0 - Device size               <ul style="list-style-type: none"> <li>= 00 - 8 bits</li> <li>= 01 - 16 bits</li> <li>= 10 - Reserved</li> <li>= 11 - Reserved</li> </ul> </li> </ul>
Byte	1FH	Arbitration level to be used; input only if bit 1 of the Mode Control field is set to 0

### Service-Specific Output

Size	Offset	Description
Byte	1FH	Allocated arbitration level



## 12H—Verify

- This function sets up the DMA controller, using the indicated values from the request block, to perform a read verification. The DMA controller performs a memory read without actually performing the transfer, until the number of DMA data transfers is reached.
- The Number of DMA Transfers field specifies the number of transfers that are to be executed by the DMA channel. The number of transfers is always 1 more than the specified count. For example, if the count is 0, one transfer is executed.
- The Mode Control field provides the following functions:
  - Bit 3 is used only if bit 1 is set to 1. Bit 3 determines whether the search for an available arbitration level starts with the lowest-priority or the highest-priority arbitration level. When bit 3 is set to 0, the search starts with the highest priority.
  - Bit 2 indicates that the I/O address will be programmed to the DMA controller. When bit 2 is set to 1, the DMA controller drives the I/O address that is specified at offset hex 14 of the request block onto the address bus during the DMA cycles, instead of driving an I/O address of 0.
  - Bit 1 specifies whether to use the arbitration level that is already allocated or to search for an available level. When bit 1 is set to 1, BIOS searches for an available level, starts the transfer, and returns the allocated arbitration level at offset hex 1F. When bit 1 is set to 0, BIOS uses the arbitration level that is specified at offset hex 1F. Bit 3 determines whether the search starts with the highest priority or the lowest priority.
  - Bit 0 causes the DMA controller to automatically initialize the transfer count and memory address to their previously-programmed values after the terminal count is reached.
- The Transfer Control field provides the following functions:
  - Bits 1 and 0 indicate whether the transfer size is 8 bits or 16 bits. The transfer-count size that is returned is in bytes if the device size is 8 bits; it is in words if the device size is 16 bits.
  - Bit 2 specifies whether the physical address of memory is increased or decreased during the transfer.
- The possible values of the Return Code field are hex 0000, 8001, 8002, 8004, 8006, C005, and C006.

## Service-Specific Input

Size	Offset	Description
DWord	10H	Physical address of memory
DWord	14H	Physical address of I/O
DWord	18H	Number of DMA data transfers
Byte	1CH	Mode control Bits 7 to 4 - Reserved Bit 3 = 1 - Start search with highest priority Bit 2 = 1 - Program I/O address Bit 1 = 1 - Allocate any arbitration level for transfer Bit 0 = 1 - Autoinitialization
Byte	1DH	Transfer control Bits 7 to 3 - Reserved Bit 2 - Count control = 0 - Increase = 1 - Decrease Bits 1, 0 - Device size = 00 - 8 bits = 01 - 16 bits = 10 - Reserved = 11 - Reserved
Byte	1FH	Arbitration level to be used; input only if bit 1 of the Mode Control field is set to 0

## Service-Specific Output

Size	Offset	Description
Byte	1FH	Allocated arbitration level

### 13H—Abort and Start New DMA Transfer

- This function accesses the DMA controller to disable the channel and to read both the physical address and the number of DMA transfers that were left when the abort request was issued. Then the DMA controller is programmed for the next DMA operation with the indicated values from the request block.
- The Number of DMA Transfers field specifies the number of transfers that are to be executed by the DMA channel. The number of transfers is always 1 more than the specified count. For example, if the count is 0, one transfer is executed.

- The Mode Control field provides the following functions:
  - Bit 3 is used only if bit 1 is set to 1. Bit 3 determines whether the search for an available arbitration level starts with the lowest-priority or the highest-priority arbitration level. When bit 3 is set to 0, the search starts with the highest priority.
  - Bit 2 indicates that the I/O address will be programmed to the DMA controller. When bit 2 is set to 1, the DMA controller drives the I/O address that is specified at offset hex 14 of the request block onto the address bus during the DMA cycles, instead of driving an I/O address of 0.
  - Bit 1 specifies whether to use the arbitration level that is already allocated or to search for an available level. When bit 1 is set to 1, BIOS searches for an available level, starts the transfer, and returns the allocated arbitration level at offset hex 1F. When bit 1 is set to 0, BIOS uses the arbitration level that is specified at offset hex 1F. Bit 3 determines whether the search starts with the highest priority or the lowest priority.
  - Bit 0 causes the DMA controller to automatically initialize the transfer count and memory address to their previously-programmed values after the terminal count is reached.
- The Transfer Control field provides the following functions:
  - Bits 1 and 0 indicate whether the transfer size is 8 bits or 16 bits. The transfer-count size that is returned is in bytes if the device size is 8 bits; it is in words if the device size is 16 bits.
  - Bit 2 specifies whether the physical address of memory is increased or decreased during the transfer.
- The possible values of the Return Code field are hex 0000, 8001, 8002, 8004, 8006, C005, and C006.

### Service-Specific Input

Size	Offset	Description
DWord	10H	Physical address of memory
DWord	14H	Physical address of I/O
DWord	18H	Number of DMA data transfers
Byte	1CH	Mode control
		Bits 7 to 4 - Reserved
		Bit 3 = 1 - Start search with highest priority
		Bit 2 = 1 - Program I/O address
		Bit 1 = 1 - Allocate any arbitration level for transfer
		Bit 0 = 1 - Autoinitialization

Size	Offset	Description
Byte	1DH	Transfer control Bits 7 to 3 - Reserved Bit 2 - Count control = 0 - Increase = 1 - Decrease Bits 1, 0 - Device size = 00 - 8 bits = 01 - 16 bits = 10 - Reserved = 11 - Reserved
Byte	1FH	Arbitration level to be used; input only if bit 1 of the Mode Control field is set to 0

### Service-Specific Output

Size	Offset	Description
DWord	10H	Physical address when abort was issued
DWord	18H	Number of transfers left when abort was issued

### 14H—Get Current Allocation Status

- This function returns the current allocation status for DMA channels and arbitration levels.
- The bit numbers in the bit maps correspond to arbitration levels or channels. For example, bit 0 at offset hex 10 corresponds to arbitration level 0; bit 0 at offset hex 14 corresponds to DMA channel 0. If a bit in the Bit Map of Allocated Arbitration Levels field (offset hex 10) is set to 1, the corresponding arbitration level is allocated. If a bit in the Bit Map of Allocated DMA Channels field (offset hex 14) is set to 1, the corresponding DMA channel is allocated. There are 15 arbitration levels and eight channels; undefined bits in the field are set to 0.
- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
DWord	1AH	Reserved

### Service-Specific Output

Size	Offset	Description
DWord	10H	Bit map of allocated arbitration levels
DWord	14H	Bit map of allocated DMA channels
Byte	18H	Number of arbitration levels not currently allocated
Byte	19H	Number of DMA channels not currently allocated

## Return Codes

Return codes are returned at offset hex 0C.

Value	Description
0000H	Operation Successfully Completed
0005H	Not My Interrupt, Stage on Interrupt
8000H	Device in Use
8001H	Arbitration Level Not Available
8002H	Arbitration Level Not Allocated
8003H	Arbitration Level Disabled
8004H	Transfer in Progress
8005H	No Transfer in Progress
8006H	No Channel Available
8007H	Arbitration Level Not Disabled
C000H	Invalid ID (ABIOS transfer convention only)
C001H	Invalid Function Number
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length
C005H	Invalid DMA Parameters
C006H	Invalid DMA Arbitration Level Specified

*Figure 6-21. Direct Memory Access (DMA) Return Codes*

## Programming Considerations

- DMA channels are either physical or virtual. A physical channel can have only one arbitration level assigned to it. A virtual channel can be programmed to use any arbitration level that is not currently assigned to a different channel.
- There is no difference in function between physical and virtual channels. Priority of the channels is determined by the arbitration level; arbitration level hex 00 has the highest priority, and arbitration level hex 0E has the lowest priority. Arbitration level hex 0F is reserved.
- To perform a DMA transfer operation, a caller performs the following steps:
  1. Request an arbitration level.
  2. Set up a transfer to a device.
  3. Disable the arbitration level.
  4. Deallocate the arbitration level.
- Direct reading or writing of the DMA controller ports can cause unpredictable results.

**Notes:**

## Device ID 10H—Programmable Option Select (POS)

### Functions

The following are the POS functions. The Default Interrupt Handler function (hex 00) and the Return Logical ID Parameters function (hex 01) are described in "Request Block" in the "Transfer Conventions" section.

**Note:** All reserved input fields must be set to 0.

**00H—Default Interrupt Handler**

**01H—Return Logical ID Parameters**

**02H—Reserved**

**03H—Read Device Parameters (Reserved)**

**04H—Set Device Parameters (Reserved)**

**05H—Reset/Initialize (Reserved)**

**06H—Enable (Reserved)**

**07H—Disable (Reserved)**

**08H—Read (Reserved)**

**09H—Write (Reserved)**

**0AH—Additional Data Transfer Function (Reserved)**

**0BH—Read Stored POS Data to Memory**

- This function returns the programmable option select data that is currently stored in 64-byte RAM or extended RAM for the specified slot.
- For system-board option select data, the output buffer contains the system-board option select byte.
- For adapter option select data, the output buffer contains 4 bytes of data (adapter option select bytes 1, 2, 3, and 4).

- If the value of the Slot Number field is greater than the maximum number of slots, no action is performed, and the Return Code field is set to hex C005 (Invalid POS Parameter).
- The possible values of the Return Code field are hex 0000, 80FE, 80FF, and C005.

### Service-Specific Input

Size	Offset	Description
Byte	10H	Slot number Bits 7 to 4 - Reserved Bits 3 to 0 - Slot number (values in binary) = 0000 - System board (planar) = 0001 - Slot 1 = 0010 - Slot 2 = 0011 - Slot 3 = 0100 - Slot 4 = 0101 - Slot 5 = 0110 - Slot 6 = 0111 - Slot 7 = 1000 - Slot 8
Byte	11H	Reserved
Word	14H	Reserved
DWord	16H	Pointer to data buffer
Word	1CH	Reserved

### Service-Specific Output

Size	Offset	Description
Word	12H	Adapter ID

### Output Buffer (System-Board POS)

Size	Offset	Description
Byte	00H	System-board POS data

### Output Buffer (Adapter POS)

Size	Offset	Description
Byte	00H	Adapter option-select byte 1
Byte	01H	Adapter option-select byte 2
Byte	02H	Adapter option-select byte 3
Byte	03H	Adapter option-select byte 4

### 0CH—Write Stored POS Data from Memory

- This function writes the programmable option select data to the specified slot locations of the appropriate RAM (64 byte RAM or extended RAM).
- For system-board option select data, the output buffer contains the system-board option select byte.



- For adapter option select data, the output buffer contains 4 bytes of data (adapter option select bytes 1, 2, 3, and 4).
- If the value of the Slot Number field is greater than the maximum number of slots, no action is performed, and the Return Code field is set to hex C005 (Invalid POS Parameter).
- The possible values of the Return Code field are hex 0000, 80FE, 80FF, and C005.

### Service-Specific Input

Size	Offset	Description
Byte	10H	Slot number Bits 7 to 4 - Reserved Bits 3 to 0 - Slot number (values in binary) = 0000 - System board (planar) = 0001 - Slot 1 = 0010 - Slot 2 = 0011 - Slot 3 = 0100 - Slot 4 = 0101 - Slot 5 = 0110 - Slot 6 = 0111 - Slot 7 = 1000 - Slot 8
Byte	11H	Reserved
Word	12H	Adapter ID
Word	14H	Reserved
DWord	16H	Pointer to data buffer
Word	1CH	Reserved

### Service-Specific Output

Size	Offset	Description
None		

#### | Output Buffer (System-Board POS)

Size	Offset	Description
Byte	00H	System-board POS data

#### | Output Buffer (Adapter POS)

Size	Offset	Description
Byte	00H	Adapter option-select byte 1
Byte	01H	Adapter option-select byte 2
Byte	02H	Adapter option-select byte 3
Byte	03H	Adapter option-select byte 4

### 0DH—Read Dynamic POS Data to Memory

- This function reads the supplied programmable option select data to the adapter in the specified slot.

- For system-board option select data, the output buffer contains the system-board option select byte.
- For adapter option select data, the output buffer contains 4 bytes of data (adapter option select bytes 1, 2, 3, and 4).
- If the value of the Slot Number field is greater than the maximum number of slots, no action is performed, and the Return Code field is set to C005 (Invalid POS Parameter).
- The possible values of the Return Code field are hex 0000 and C005.

### Service-Specific Input

Size	Offset	Description
Byte	10H	Slot number (values in binary) Bits 7 to 4 - Reserved Bits 3 to 0 - Slot number = 0000 - System board (planar) = 0001 - Slot 1 = 0010 - Slot 2 = 0011 - Slot 3 = 0100 - Slot 4 = 0101 - Slot 5 = 0110 - Slot 6 = 0111 - Slot 7 = 1000 - Slot 8
Byte	11H	Reserved
Word	14H	Reserved
DWord	16H	Pointer to data buffer
Word	1CH	Reserved

### Service-Specific Output

Size	Offset	Description
Word	12H	Card ID

### | Output Buffer (System-Board POS)

Size	Offset	Description
Byte	00H	System-board POS data

### | Output Buffer (Adapter POS)

Size	Offset	Description
Byte	00H	Adapter option-select byte 1
Byte	01H	Adapter option-select byte 2
Byte	02H	Adapter option-select byte 3
Byte	03H	Adapter option-select byte 4

**0EH—Write Dynamic POS Data from Memory**

- This function writes the supplied programmable option select data to the adapter in the specified slot.
- For system-board option select data, the output buffer contains the system-board option select byte.
- For adapter option select data, the output buffer contains 4 bytes of data (adapter option select bytes 1, 2, 3, and 4).
- If the value of the Slot Number field is greater than the maximum number of slots, no action is performed, and the Return Code field is set to C005 (Invalid POS Parameter).
- The possible values of the Return Code field are hex 0000 and C005.

**Service-Specific Input**

Size	Offset	Description
Byte	10H	Slot number (values in binary) Bits 7 to 4 - Reserved Bits 3 to 0 - Slot Number = 0000 - System board (planar) = 0001 - Slot 1 = 0010 - Slot 2 = 0011 - Slot 3 = 0100 - Slot 4 = 0101 - Slot 5 = 0110 - Slot 6 = 0111 - Slot 7 = 1000 - Slot 8
Byte	11H	Reserved
Word	14H	Reserved
DWord	16H	Pointer to data buffer
Word	1CH	Reserved

**Service-Specific Output**

Size	Offset	Description
None		

**Input Buffer (System-Board POS)**

Size	Offset	Description
Byte	00H	System-board POS data

## **| Input Buffer (Adapter POS)**

<b>Size</b>	<b>Offset</b>	<b>Description</b>
Byte	00H	Adapter option-select byte 1
Byte	01H	Adapter option-select byte 2
Byte	02H	Adapter option-select byte 3
Byte	03H	Adapter option-select byte 4

## **Return Codes**

Return codes are returned at offset hex 0C.

<b>Value</b>	<b>Description</b>
0000H	Operation Successfully Completed
80FEH	NVRAM Checksum Invalid
80FFH	NVRAM Battery Bad
C000H	Invalid Logical ID (ABIOS transfer convention only)
C001H	Invalid Function Number
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length
C005H	Invalid POS Parameter

*Figure 6-22. Programmable Option Select (POS) Return Codes*

## Device ID 16H—Keyboard Security

### Functions

The following are the keyboard security functions. The Default Interrupt Handler function (hex 00) and the Return Logical ID Parameters function (hex 01) are described in "Request Block" in the "Transfer Conventions" section.

**Note:** All reserved input fields must be set to 0.

#### 00H—Default Interrupt Handler (Reserved)

#### 01H—Return Logical ID Parameters

#### 02H—Reserved

#### 03H—Read Device Parameters

- This function returns the maximum password length.
- The possible value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Byte	11H	Reserved

### Service-Specific Output

Size	Offset	Description
Byte	10H	Maximum password length

#### 04H—Set Device Parameters (Reserved)

#### 05H—Reset/Initialize (Reserved)

#### 06H—Enable

- This function enables password security.
- The possible values of the Return Code field are hex 0000, 8000, and 8003.

### Service-Specific Input

Size	Offset	Description
Byte	11H	Reserved

### Service-Specific Output

Size	Offset	Description
None		

**07H—Disable (Reserved)**

**08H—Read (Reserved)**

**09H—Write (Reserved)**

**0AH—Additional Data Transfer (Reserved)**

**0BH—Write Password**

- This function changes the password.
- Password scan codes are written to the keyboard controller. A password can consist of from 1 to 7 bytes of keyboard-make scan codes. It can consist of letters, numbers, and other characters; keys that are tagged with a null byte (such as Alt, Caps Lock, Ctrl, Num Lock, Shift, and Scroll Lock) are not valid.
- If the value of the Password Length field is 0 or is greater than the maximum password length, no action is performed, and the Return Code field is set to hex C005 (Invalid Keyboard-Security Parameter).
- The maximum password length is returned in the Read Device Parameters function (hex 03).
- The possible values of the Return Code field are hex 0000, 8000, and 8003.

### Service-Specific Input

Size	Offset	Description
Byte	10H	Password length (bytes)
Byte	11H	Reserved
Byte	12H	First scan code
Byte	13H	Second scan code
Byte	14H	Third scan code
Byte	15H	Fourth scan code
Byte	16H	Fifth scan code
Byte	17H	Sixth scan code
Byte	18H	Seventh scan code

### Service-Specific Output

Size	Offset	Description
None		

### 0CH—Write Invocation Byte

- This function changes the invocation-byte scan code, which is used to signal to the system that keyboard security has been activated with a valid password. After keyboard security is activated, the system sends this byte (by using the Keyboard interrupt) to the operating system as if the byte were a scan code. If the invocation byte is set to 0, the system does not send this byte after keyboard security has been activated.
- The possible values of the Return Code field are hex 0000, 8000, and 8003.

#### Service-Specific Input

Size	Offset	Description
Byte	10H	Invocation-byte scan code
Byte	11H	Reserved

#### Service-Specific Output

Size	Offset	Description
None		

### 0DH—Write Match Byte

- This function changes the match-byte scan code, which is used to signal to the system that keyboard security has been deactivated with the correct password. After the correct sequence is typed, the system sends this byte (by using the Keyboard interrupt) to the operating system as if the byte were a scan code. If the match byte is set to 0, the system does not send this byte after keyboard security has been deactivated.
- The possible values of the Return Code field are hex 0000, 8000, and 8003.

#### Service-Specific Input

Size	Offset	Description
Byte	10H	Match-byte scan code
Byte	11H	Reserved

#### Service-Specific Output

Size	Offset	Description
None		

**0EH—Write Filter Byte 1**

- This function changes filter byte 1. Filter bytes are scan codes that are ignored during password validation. For example, it might be preferable to ignore the scan code for the shift keys.
- The possible values of the Return Code field are hex 0000, 8000, and 8003.

**Service-Specific Input**

Size	Offset	Description
Byte	10H	Filter byte 1
Byte	11H	Reserved

**Service-Specific Output**

Size	Offset	Description
None		

**0EH—Write Filter Byte 2**

- This function changes filter byte 2. Filter bytes are scan codes that are ignored during password validation. For example, it might be preferable to ignore the scan code for the shift keys.
- The possible values of the Return Code field are hex 0000, 8000, and 8003.

**Service-Specific Input**

Size	Offset	Description
Byte	10H	Filter byte 2
Byte	11H	Reserved

**Service-Specific Output**

Size	Offset	Description
None		



## Return Codes

Return codes are returned at offset hex 0C.

Value	Description
0000H	Operation Successfully Completed
8000H	Device Busy
8003H	Device Inhibited
C000H	Invalid Logical ID (ABIOS transfer convention only)
C001H	Invalid Function Number
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length
C005H	Invalid Keyboard-Security Parameter

*Figure 6-23. Keyboard-Security Return Codes*

**Notes:**



# Device ID 17H—SCSI Subsystem Interface

## Functions

The functions that are described in this section control the small computer system interface (SCSI) subsystem. Some of these functions affect all devices that are attached to the SCSI subsystem; therefore, the programmer should have a thorough understanding of the operation of the SCSI bus and subsystem before using these functions (see the technical reference manual for the subsystem).

The Default Interrupt Handler function (hex 00) and the Return Logical ID Parameters function (hex 01) are described in "Request Block" in the "Transfer Conventions" section.

**Note:** All reserved input fields must be set to 0.

### 00H—Default Interrupt Handler

### 01H—Return Logical ID Parameters

### 02H—Reserved

### 03H—Read Device Parameters

- This function returns information about the SCSI subsystem.
- All return-code values are possible.

### Service-Specific Input

Size	Offset	Description
Word	28H	Reserved

### Service-Specific Output

Size	Offset	Description
Word	10H	Length of device configuration table, in bytes (see function hex 0B)
Byte	12H	Subsystem control block (SCB) architecture card compatibility level
Word	14H	Reserved

### 04H—Set Device Parameters (Reserved)

## **05H—Reset/Initialize**

- This function issues either a hard reset or a soft reset to the subsystem. The Reset Type field selects the type of reset that will be issued.
- The subsystem and its devices must be in a stable state throughout this function.
- This function affects the SCSI and disk BIOS routines.
- If the reset type is not within the range that is defined in the Reset Type field (offset hex 10), the Return Code field is set to hex C005 (Invalid Parameter), and no action is taken.

### **Hard Reset**

- The subsystem stops all current activity, including any active requests that the subsystem is maintaining for attached devices. However, the devices can continue to process the requests, and the programs that are controlling the devices can appear to still have active requests, even though these requests will not be successfully completed. This results in an apparent time-out condition for each logical ID that has an outstanding request. Invoke the Time-Out routine for those logical IDs.
- For reset type 3, ABIOs restores the DMA pacing value to the value that was in effect before the reset. Also, the device time-out states are restored (on a device-by-device basis) to the states that were in effect before the reset.
- For reset type 2, the parameters remain at their default values.
- The assignments are always reestablished.
- The subsystem-retry state remains at the default setting (enabled).
- All return-code values are possible.

### **Soft Reset**

- The subsystem stops all current activity and activates the SCSI reset signal.
- A soft reset does not change the adapter DMA pacing and time-out states.
- All return-code values are possible.

## Service-Specific Input

Size	Offset	Description
Byte	10H	Reset type = 00H to 01H - Reserved = 02H - Hard reset; leaves subsystem states at default values = 03H - Hard reset; restores subsystem states to their pre-reset values = 04H - Soft reset; issues 'SCSI bus reset' signal = 05H to FFH - Reserved
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
DWord	28H	Time to wait before resuming request, in microseconds
Byte	32H	Status Bits 7 to 2 - Reserved Bit 1 - Command-complete status indicator (see "Programming Considerations" on page 6-ID17-9) = 0 - Command-complete status not required = 1 - Command-complete status should be requested for more information Bit 0 - Reserved

**06H—Enable (Reserved)**

**07H—Disable (Reserved)**

**08H—Read (Reserved)**

**09H—Write (Reserved)**

**0AH—Additional Data Transfer (Reserved)**

**0BH—Return Device Configuration Table**

- This function builds a device configuration table at the specified address. This table describes the physical devices that are attached to the SCSI subsystem.
- The first entry in the device configuration table is one word long. It specifies the number of different peripheral types that are in the system.

- For each peripheral type, there is a 2-byte entry in the table that indicates the peripheral type and the number of physical devices of that type that are attached to the SCSI subsystem. Each physical-device entry has the following format:

Number of Devices	Peripheral Type
-------------------	-----------------

If a peripheral type is not present in the system, it does not have an entry in the table. However, if a peripheral type is present in the system but is not attached to the SCSI subsystem, the device count is 0. This helps a controlling program establish the SCSI configuration across all SCSI subsystems.

- The entire length of this table (in bytes) can be obtained through the Read Device Parameters function (hex 03).
- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Word	10H	Reserved
DWord	12H	Logical pointer to table-build area
Word	16H	Reserved

### Service-Specific Output

The following table is built at the specified address:

Size	Offset	Description
Word	0000H	Number of peripheral-device-type entries ( <i>m</i> )
Word	0002H	Peripheral-device-type entry 1
Word	0004H	Peripheral-device-type entry 2
Word	( <i>m</i> ×2)H	Peripheral-device-type entry <i>m</i>

Figure 6-24. Device Configuration Table Format

### 0CH—Return Interrupting Logical ID

- This function returns the interrupt-pending status of the SCSI subsystem. If there is an interrupt pending, this function also returns the logical ID that is associated with the interrupt. If no interrupt is pending, the Interrupting Logical ID field is undefined.
- The value of the Return Code field is hex 0000.

## Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
Byte	10H	Interrupt status Bits 7 to 1 - Reserved Bit 0 - Status = 0 - Interrupt not pending = 1 - Interrupt pending
Word	12H	Interrupting logical ID; valid only if an interrupt is pending

## 10H—Set SCSI Subsystem DMA Pacing Factor

- This function programs the SCSI subsystem with the specified pacing value. All devices that are attached to the SCSI subsystem are affected.
- The pacing value is expressed as a percentage in the range from 25 to 100, inclusive.
- BIOS does not check the range of the pacing value. Using a value outside the specified range can cause an adapter error.
- All devices must be inactive at the time of this request.
- All return-code values are possible.

## Service-Specific Input

Size	Offset	Description
Byte	10H	Pacing factor (from 25% to 100%)
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
DWord	28H	Time to wait before resuming request, in microseconds
Byte	32H	Status Bits 7 to 2 - Reserved Bit 1 - Command-complete status indicator (see "Programming Considerations" on page 6-ID17-9) = 0 - Command-complete status not required = 1 - Command-complete status should be requested for more information Bit 0 - Reserved

## 11H—Return Device DMA Pacing Factor

- This function returns the current pacing factor for the SCSI subsystem that is specified by the logical ID.
- The pacing value is expressed as a percentage in the range from 25 to 100, inclusive.
- The value of the Return Code field is hex 0000.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

### Service-Specific Output

Size	Offset	Description
Byte	10H	Current pacing factor

## 12H—Transfer SCB

- This function programs the SCSI subsystem to process the subsystem control block (SCB) that is pointed to by the Physical Pointer to SCB field.
- BIOS does not check the validity of the SCB.
- The SCB chain header has the following format:

Size	Offset	Description
Word	00H	Reserved (set to 0)
DWord	02H	Logical pointer to next SCB header in chain, or chain-ending indicator (0)
Word	06H	Reserved
Word	08H	Reserved
DWord	0AH	Logical pointer to termination status block (TSB) that is associated with this SCB
Word	0EH	Reserved

- A logical pointer of 0 ends the SCB chain.
- The chain must have an ending.
- If the Logical Pointer to SCB Chain Header field (offset hex 16 on input) is set to 0, BIOS does not initiate the SCB transfer, and the Return Code field is set to hex 0000 (Operation Successfully Completed).
- The chain header must immediately precede the SCB.
- All return-code values are possible.



- See the “Chain Example” diagram in “Transfer SCB Request Block” in “Device ID 02H—Fixed Disk.”

### Service-Specific Input

Size	Offset	Description
DWord	10H	Physical pointer to SCB
Word	14H	Reserved
DWord	16H	Logical pointer to SCB chain header
Word	1CH	Reserved
Word	26H	Reserved
Word	2CH	Reserved
Byte	2EH	Flags
		Bits 7 to 1 - Reserved (set to 0)
		Bit 0 - Length
		= 0 - Normal-length SCB
		= 1 - Long SCB

### Service-Specific Output

Size	Offset	Description
DWord	28H	Time to wait before resuming request, in microseconds
Byte	32H	Status
		Bits 7 to 2 - Reserved
		Bit 1 - Command-complete status indicator
		(see “Programming Considerations” on page 6-ID17-9)
		= 0 - Command-complete status not required
		= 1 - Command-complete status should be requested
		for more information
		Bit 0 - Reserved

### 13H—Reserved

## Return Codes

Return codes are returned at offset hex 0C.

Value	Description
0000H	Operation Successfully Completed
0001H	Incomplete – Stage on Interrupt
0002H	Incomplete – Stage on Time
0005H	Incomplete – Not My Interrupt, Stage on Interrupt
8000H	Device Busy, Request Refused
8003H	LID Not Associated with This Adapter
8100H	Device Busy, Request Refused
9000H	Operation Ended in Error
900CH	Command Completed with Failure
900EH	Command Error
900FH	Sequence Error
9020H	Bad Controller
9100H	Operation Ended in Error
910CH	Command Completed with Failure
910EH	Command Error
910FH	Sequence Error
9120H	Bad Controller
A000H	Operation Ended in Time-Out
A020H	Bad Controller
A100H	Operation Ended in Time-Out
A120H	Bad Controller
B000H	Time-Out Routine Failed
B020H	Bad Controller
B100H	Time-Out Routine Failed
B120H	Bad Controller
C000H	Invalid Logical ID (ABIOS transfer convention only)
C001H	Invalid Function
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length
C005H	Invalid Parameter
FFFFH	Return Code Not Valid

**Figure 6-25.** *SCSI Subsystem Interface Return Codes*

## **Programming Considerations**

- Some error conditions do not result in a termination status block (TSB) being automatically stored. When a TSB is not stored, bit 1 of the Status field (offset hex 32 on output) is set to 1 to indicate that a Get Command-Complete Status SCB should be built and sent through the Transfer SCB function (hex 12) for more information. If the command does not result in an error, this bit is set to 0. The controlling program can still request command-complete status. This bit is defined for return codes hex 9000 to hex BFFF only.
- The SCSI-adaptor device ID and the DMA device ID must be initialized before the SCSI peripheral-type device ID is initialized.
- BIOS calls must not be made in an ABIOS environment.
- During ABIOS initialization, the following BIOS services are used:
  - Interrupt 13H (the BIOS hardware interrupt service must be in place)
  - Interrupt 15H.

**Notes:**

## Device ID 18H—SCSI Peripheral Type

### Functions

Logical IDs are allocated to specific small computer system interface (SCSI) devices on demand. Through the Allocate SCSI Peripheral Device function (hex 15), the device driver specifies a SCSI peripheral type, a removable-media indicator, and a relative unit number. If a device exists and is unallocated, BIOS assigns a logical ID to that device. The controlling program can then use the logical ID to make device requests through the SCSI adapter. The structure for input to the SCSI adapter is the subsystem control block (SCB). The Transfer SCB function (hex 12) transfers SCBs to the adapter.

The Deallocate SCSI Peripheral Device function (hex 14) removes the association between a specific device and a logical ID. The controlling program uses this function to release a device that it does not support. The program issues the Deallocate SCSI Peripheral Device function (hex 14), allocates another device to the logical ID, and determines whether it supports that device. Only one device can be allocated to a logical ID, and only one logical ID can be associated with a device at any time.

**Note:** All reserved input fields must be set to 0.

### 00H—Default Interrupt Handler

- This is a standard BIOS function call. It requires device allocation before it can be executed. If this function is called without device allocation, the return code is hex 0005, not hex 8003.
- This function is described fully in "Request Block" in the "Transfer Conventions" section.

**01H—Return Logical ID Parameters**

- This is a standard BIOS function call. If a device has not been allocated to this logical ID, the Hardware Interrupt Level field (offset hex 10) and the Arbitration Level field (offset hex 11) are set to hex 0FD to indicate that the information is not currently available. When an allocation is successful, this function is called to obtain the values for the Hardware Interrupt Level field and the Arbitration Level field.
- The value of the Return Code field is hex 0000.
- This function is described fully in “Request Block” in the “Transfer Conventions” section.

**02H—Reserved**

**03H—Read Device Parameters**

- This function returns information about the SCSI device.
- This function requires device allocation before it can be executed.
- The device-power status and device-defective status are set during POST and are returned at offset hex 14 of the request block. BIOS does not attempt to reset these flags or make any determinations on the basis of the state of these flags.
- All return-code values are possible.

**Service-Specific Input**

Size	Offset	Description
Word	28H	Reserved

**Service-Specific Output**

Size	Offset	Description
Word	10H	Reserved
Byte	12H	SCB architecture card compatibility level
Byte	13H	Adapter index (0 based)
Word	14H	Device flags
		Bits 15 to 2 - Reserved
		Bit 1 - Device power during POST
		= 0 - Power on
		= 1 - Power off
		Bit 0 - Device-defective error during POST
		= 0 - Device not defective
		= 1 - Device defective
Byte	16H	Logical unit number (LUN)
Byte	17H	Physical unit number (PUN)

**04H—Set Device Parameters (Reserved)**

**05H—Reset/Initialize**

- This function issues a Reset command to the physical device. All logical units on the device are affected.
- This function requires device allocation before it can be executed.
- All return-code values are possible.

**Service-Specific Input**

Size	Offset	Description
Word	16H	Reserved

**Service-Specific Output**

Size	Offset	Description
DWord	28H	Time to wait before resuming request, in microseconds
Byte	32H	Status <ul style="list-style-type: none"><li>Bits 7 to 2 - Reserved</li><li>Bit 1 - Command-complete status indicator (see "Programming Considerations" on page 6-ID18-11)<ul style="list-style-type: none"><li>= 0 - Command-complete status not required</li><li>= 1 - Command-complete status should be requested for more information</li></ul></li><li>Bit 0 - Reserved</li></ul>

**06H—Enable (Reserved)**

**07H—Disable (Reserved)**

**08H—Read (Reserved)**

**09H—Write (Reserved)**

**0AH—Additional Data Transfer (Reserved)**

**0BH to 0FH—Reserved**

## 10H—Set Device Time-Out

- This function programs the adapter time-out value for a device. It does not affect any other device on the adapter.
- This function requires device allocation before it can be executed.
- If the time-out count is set to 0, the adapter is programmed not to time out for commands to the device. This allows for operations that can take longer than the 127-minute maximum that the interface provides. On all subsequent stage-on-interrupt returns, the Time-Out Value field is set to 0 to show that there is no time-out for the operation. The controlling program must call the time-out entry point to terminate the request, if the interrupt has not occurred after an appropriate length of time.
- All return-code values are possible.

### Service-Specific Input

Size	Offset	Description
Byte	10H	Time-out value Bit 7 - Granularity = 0 - Time-out count is expressed in seconds = 1 - Time-out count is expressed in minutes Bits 6 to 0 - Time-out count
Byte	11H	Reserved
Word	16H	Reserved

### Service-Specific Output

Size	Offset	Description
DWord	28H	Time to wait before resuming request, in microseconds
Byte	32H	Status Bits 7 to 2 - Reserved Bit 1 - Command-complete status indicator (see "Programming Considerations" on page 6-ID18-11) = 0 - Command-complete status not required = 1 - Command-complete status should be requested for more information Bit 0 - Reserved

## 11H—Read Device Time-Out

- This function returns the current time-out value for a device. It does not affect any other device on the adapter.
- This function requires device allocation before it can be executed.
- The possible values of the Return Code field are hex 0000 and 8003.



## Service-Specific Input

Size	Offset	Description
Byte	11H	Reserved
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
Byte	10H	Time-out value Bit 7 - Granularity = 0 - Time-out count is expressed in seconds = 1 - Time-out count is expressed in minutes Bits 6 to 0 - Time-out count

## 12H—Transfer SCB

- This function transfers the subsystem control block (SCB) that is pointed to by the Physical Pointer to SCB field to the adapter that controls the specified device.
- This function requires device allocation before it can be executed.
- ABIOS does not check the validity of the SCBs.
- The SCB chain header has the following format:

Size	Offset	Description
Word	00H	Reserved (set to 0)
DWord	02H	Logical pointer to next SCB header in chain, or chain-ending indicator (0)
Word	06H	Reserved
Word	08H	Reserved
DWord	0AH	Logical pointer to termination status block (TSB) that is associated with this SCB
Word	0EH	Reserved

- A logical pointer of 0 ends the SCB chain.
- The chain must have an ending.
- If the Logical Pointer to SCB Chain Header field (offset hex 16 on input) is set to 0, ABIOS does not initiate the SCB transfer, and the Return Code field is set to hex 0000 (Operation Successfully Completed).
- The chain header must immediately precede the SCB.
- All return-code values are possible.
- See the "Chain Example" diagram in "Transfer SCB Request Block" in "Device ID 02H—Fixed Disk."

**Note:** The Read and Write SCB commands are the primary interface to devices such as IBM-type fixed disk drives. Do not use the

Send Other SCSI command to read from or write to these devices, because it can delete data from the disk.

### Service-Specific Input

Size	Offset	Description
DWord	10H	Physical pointer to SCB
Word	14H	Reserved
DWord	16H	Logical pointer to SCB chain header
Word	1CH	Reserved
Word	26H	Reserved
Word	2CH	Reserved
Byte	2EH	Flags
		Bits 7 to 1 - Reserved
		Bit 0 - Length
		= 0 - Normal-length SCB
		= 1 - Long SCB

### Service-Specific Output

Size	Offset	Description
DWord	28H	Time to wait before resuming request, in microseconds
Byte	32H	Status
		Bits 7 to 2 - Reserved
		Bit 1 - Command-complete status indicator
		(see "Programming Considerations" on page 6-ID18-11)
		= 0 - Command-complete status not required
		= 1 - Command-complete status should be requested for more information
		Bit 0 - Reserved

### 14H—Deallocate SCSI Peripheral Device

- This function deallocates the SCSI peripheral-device type that is assigned to a logical ID.
- If the device type is not currently allocated, the Return Code field is set to hex 8003 (Device Not Allocated to This Logical ID).
- The possible values of the Return Code field are hex 0000, 8000, and 8003.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

### Service-Specific Output

Size	Offset	Description
None		

## 15H—Allocate SCSI Peripheral Device

- This function allocates the  $n$ th available device of the specified SCSI device type in the request block to a logical ID, where  $0 \leq n \leq$  the device count (see the Return Peripheral-Type Count function (hex 16)). When  $n$  is 0, the next available device of the specified type is allocated. Any other value of  $n$  causes the  $n$ th device of the specified peripheral type to be allocated.
- This function must be executed, because no device is associated with a logical ID after BIOS initialization. The controlling program fills in the request-block fields that are needed to describe the requested device. These fields are:
  - Peripheral-device type (offset hex 10)
  - Removable-media indicator (bit 7 of offset hex 11)
  - $n$ th device of this type (offset hex 12).
- Some BIOS functions can be called without a device being allocated to a logical ID. (See Figure 6-27 on page 6-ID18-11 to determine which functions require device allocation.)
- Only one SCSI device can be allocated to a logical ID.
- If the requested device does not exist, the Return Code field is set to hex 8002 (Device Not Available).
- If a device is already allocated to a logical ID, the Return Code field is set to hex 8004 (A Device Is Already Allocated to This Logical ID).
- If the requested device exists but is currently allocated to another logical ID, the Return Code field is set to hex 8005 (Requested Device Allocated to Another Logical ID).
- If the requested device exists, but its associated arbitration level could not be allocated, the Return Code field is set to hex 8006 (Arbitration Level Could Not Be Allocated for Requested Device).
- The possible values of the Return Code field are hex 0000, 8002, 8004, 8005, and 8006.

### Service-Specific Input

Size	Offset	Description
Byte	10H	Peripheral-device type
Byte	11H	Device-type flags <ul style="list-style-type: none"><li>Bit 7 - Removable-media indicator<ul style="list-style-type: none"><li>= 0 - Device media is not removable</li><li>= 1 - Device media is removable</li></ul></li><li>Bits 6 to 0 - Reserved</li></ul>
Word	12H	$n$ th device of this type
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
None		

### 16H—Return Peripheral-Type Count

- This function returns the number of SCSI devices that are available for the specified type. The controlling program fills in the request-block fields that are needed to describe the requested device. These fields are:
  - Peripheral-device type (offset hex 10)
  - Removable-media indicator (bit 7 of offset hex 11).
- A count of 0 indicates that no devices of the specified type were found.
- The value of the Return Code field is hex 0000.

## Service-Specific Input

Size	Offset	Description
Byte	10H	Peripheral-device type
Byte	11H	Device-type flags <ul style="list-style-type: none"><li>Bit 7 - Removable-media indicator<ul style="list-style-type: none"><li>= 0 - Device media is not removable</li><li>= 1 - Device media is removable</li></ul></li><li>Bits 6 to 0 - Reserved</li></ul>
Word	16H	Reserved

## Service-Specific Output

Size	Offset	Description
Word	14H	Count of requested device type

### 17H—Abort

- This function issues an Abort command to the specified device.
- This function requires allocation of the device to the logical ID.
- This function can be issued regardless of whether another request is outstanding.
- To abort an existing request, a new request block for the Abort function is created; the pending request block must be kept active. If the Abort command cannot be issued, the Return Code field is set to hex 8000 (Device Busy, Request Refused). If the Return Code field is set to hex FFFF or 8000 and an interrupt occurs for the device, the pending request block must be passed to BIOS to enable BIOS to handle the interrupt. If the Return Code field is not set to hex FFFF or 8000, the request block is

used to answer the interrupt, and the previous request block is considered to be complete.

- All return-code values are possible.

### Service-Specific Input

Size	Offset	Description
Word	16H	Reserved

### Service-Specific Output

Size	Offset	Description
DWord	28H	Time to wait before resuming request, in microseconds
Byte	32H	Status
		Bits 7 to 2 - Reserved
		Bit 1 - Command-complete status indicator
		(see "Programming Considerations" on page 6-ID18-11)
		= 0 - Command-complete status not required
		= 1 - Command-complete status should be requested
		for more information
		Bit 0 - Reserved

## Return Codes

Return codes are returned at offset hex 0C.

Value	Description
0000H	Operation Successfully Completed
0001H	Incomplete – Stage on Interrupt
0002H	Incomplete – Stage on Time
0005H	Incomplete – Not My Interrupt, Stage on Interrupt
8000H	Device Busy, Request Refused
8002H	Device Not Available
8003H	Device Not Allocated to This Logical ID
8004H	A Device Is Already Allocated to This Logical ID
8005H	Requested Device Allocated to Another Logical ID
8006H	Arbitration Level Could Not Be Allocated for Requested Device
8100H	Device Busy, Request Refused
9000H	Operation Ended in Error
900CH	Command Failure
900EH	Invalid Adapter Command/Parameter
900FH	Sequence Error
9020H	Bad Controller
9100H	Operation Ended in Error
910CH	Command Failure
910EH	Unsupported Adapter Command/Parameter
910FH	Sequence Error
9120H	Bad Controller
A000H	Operation Ended in Time-Out, No Additional Status
A080H	Operation Ended in Time-Out
A100H	Operation Ended in Time-Out, No Additional Status
A180H	Operation Ended in Time-Out
B000H	Time-Out Routine Failed
B00CH	Time-Out Routine Completed with Failure
B00EH	Time-Out Routine Command Error
B00FH	Time-Out Routine Sequence Error
B020H	Bad Controller
B080H	Time-Out
B100H	Time-Out Routine Failed
B10CH	Time-Out Routine Completed with Failure
B10EH	Time-Out Routine Command Error
B10FH	Time-Out Routine Sequence Error
B120H	Bad Controller
B180H	Time-Out
C000H	Invalid Logical ID (ABIOS transfer convention only)
C001H	Invalid Function
C003H	Invalid Unit Number
C004H	Invalid Request-Block Length
C005H	Invalid ABIOS Parameter
C008H	Cache Buffer Not Supported
FFFFH	Return Code Not Valid

Figure 6-26. SCSI Peripheral Type Return Codes

## Programming Considerations

- The following figure identifies the calling restrictions that are on the SCSI peripheral-type functions.

Function Number	Requires Device Allocation
0000H	Yes*
0001H	No**
0003H	Yes
0005H	Yes
0010H	Yes
0011H	Yes
0012H	Yes
0014H	Yes
0015H	No
0016H	No
0017H	Yes

\* Returns hex 0005 (Not My Interrupt—Stage on Interrupt) if device is not allocated

\*\* Returns hex FD for arbitration and interrupt level if device is not allocated

**Figure 6-27. Function Restrictions**

- Some error conditions do not result in a termination status block (TSB) being automatically stored. When a TSB is not stored, bit 1 of the Status field (offset hex 32 on output) is set to 1 to indicate that a Get Command-Complete Status SCB should be built and sent through the Transfer SCB function (hex 12) for more information. If the command does not result in an error, this bit is set to 0. The controlling program can still request command-complete status. This bit is defined for return codes hex 9000 to hex BFFF only.
- The SCSI-adaptor device ID and the DMA device ID must be initialized before the SCSI peripheral-type device ID is initialized.
- BIOS calls must not be made in an ABIOS environment.
- The activity LED is defined for fixed-disk devices only. If a device is a fixed disk drive (512-byte blocks, nonremovable media, peripheral type 0), the activity LED is used.
- During ABIOS initialization, the following BIOS services are used:
  - Interrupt 13H (the BIOS hardware interrupt service must be in place)
  - Interrupt 15H.

**Notes:**





