IBM Personal System/2® and Personal Computer BIOS Interface Technical Reference

15F0306

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

Second Edition (May, 1988)

Preface

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

- IBM PC Convertible
- IBM PC*jr*™
- IBM Personal Computer
- IBM Personal Computer AT®
- IBM Personal Computer XT™
- IBM Personal Computer XT Model 286
- IBM *Portable* Personal Computer
- IBM Personal System/2® Models 25, 30, 50, 60, 70, and 80
- IBM Color/Graphics Monitor Adapter
- IBM Enhanced Graphics Adapter
- IBM Monochrome Display and Printer Adapter
- IBM devices using ESDI-type commands.

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 three parts: **BIOS**, **Advanced BIOS**, and **Supplements**.

BIOS contains the following:

Section 1, "Introduction to BIOS," provides an overview of BIOS, interrupts, parameter passing, data areas and read-only memory

PC*jr* and Personal Computer XT are trademarks of the International Business Machines Corporation.

Personal Computer AT and Personal System/2 are registered trademarks of the International Business Machines Corporation.

(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 and ROM Tables," contains detailed information about regular data areas, extended data areas, and ROM tables for system and adapter ROM BIOS.

Section 4, "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, multitasking provisions, system identification bytes, keyboard keys, and scan code/character code combinations.

An index is provided for BIOS.

Advanced BIOS contains the following sections:

Section 1, "Introduction to Advanced BIOS," provides an overview of Advanced BIOS, Data Structures, Initialization, Request Blocks and 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 necessary to make the ABIOS interface operational.

Section 4, "Transfer Conventions," describes the methods 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, extending ABIOS, and operating system implementation considerations.

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

An index is also provided for Advanced BIOS.

Supplements is reserved for additional BIOS and Advanced BIOS interface information. The Asynchronous Communications Supplement and the Programmable Option Select Supplement are already provided. Record on the *Contents - Supplements* page any supplements that you add. Supplements to this technical reference will be offered for sale as additional BIOS and Advanced BIOS interface information becomes available.

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.

Important: Information added to the **Supplements** area of this technical reference may have new information about subjects covered in other parts of this technical reference. Refer to **Supplements** for information that could affect your hardware or software development decisions.

Notes:

Contents - BIOS

Figures - BIOS

Section 1. Introduction to BIOS

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. The 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 rather than 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 by software interrupts; each BIOS entry point is available through its own interrupt. The AH register, where appropriate, indicates the specific routine within the overall interrupt function that is being executed.

Software interrupts INT 10H through INT 1AH each access different BIOS routines. For example, INT 12H invokes the BIOS routine for determining memory size and returns the value to the caller.

See Section 2, "Interrupts," for additional information.

Parameter Passing

All parameters passed to and from the BIOS routines go through the microprocessor registers. Each BIOS interrupt routine indicates the registers used on the call and 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 except (AX), the flags, and those registers that return a value to the caller. In some cases, other registers are modified. See Section 2, “Interrupts,” for additional information.

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

Data Areas and ROM Tables

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

See Section 3, “Data Areas and ROM Tables,” for additional information.

BIOS Level Determination

The BIOS is contained in ROM modules located 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.

The BIOS has been amended several times since its inception. 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, run the following BASIC program. The date that is displayed is the version date of the 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” on page 4-18 for a list of IBM products and their BIOS version dates. To access this information, see INT 15H, “(AH) = C0H Return System Configuration Parameters” on page 2-92.

System Groups

In this technical reference, IBM systems are categorized into groups with similar BIOS interfaces. These groups are referred to with any exceptions noted. The groups with similar interfaces include:

- Personal System/2 products - all models
- Personal Computer XT products - includes *Portable* Personal Computer
- Personal Computer AT products - all models.

Important: Information added to the **Supplements** area of this technical reference may have new information about subjects covered in other parts of this technical reference. Refer to the supplements for information that could affect your hardware or software development decisions.

Notes:

Section 2. Interrupts

Notes:

The following figure lists each interrupt, its function, and, where applicable, the location of a more detailed description of the interrupt.

Important: The **Supplements** section of this book may contain additional information that could affect your hardware or software development decisions.

Interrupt Number (Hex)	**Interrupt Function**
00	Divide by 0
01	Single Step
02	Nonmaskable (NMI) (See page 2-5)
03	Breakpoint
04	Overflow
05	Print Screen (See page 2-7)
06 to 07	– Reserved –
08	System Timer (See page 2-8)
09	Keyboard (See page 2-9)
0A to 0D	– Reserved –
0E	Diskette (See INT 13H on page 2-48)
0F	– Reserved –
10	Video (See page 2-11)
11	Equipment Determination (See page 2-46)
12	Memory Size Determination (See page 2-47)
13	Fixed Disk/Diskette (See pages 2-48 and 2-58)
14	Asynchronous Communications (See page 2-69)
15	System Services (See page 2-73)
16	Keyboard (See page 2-102)
17	Printer (See page 2-110)
18	Resident BASIC
19	Bootstrap Loader (See page 2-113)
1A	System Timer and Real-Time Clock Services (See page 2-114)
1B	Keyboard Break (See INT 09H on page 2-9)
1C	User Timer Tick (See INT 08H on page 2-8)
1D	Video Parameters
1E	Diskette Parameters (See "Diskette Drive Parameter Table" on page 3-26)
1F	Video Graphics Characters

Figure 2-1 (Part 1 of 2). Interrupts

Interrupt Number (Hex)	Interrupt Function
20 to 3F	– Reserved for Disk Operating System (DOS) –
40	Diskette BIOS Revector
41	Fixed Disk Parameters (See INT 13H on page 2-58)
42 to 45	– Reserved –
46	Fixed Disk Parameters (See INT 13H on page 2-58)
47 to 49	– Reserved –
4A	User Alarm (See INT 08H on page 2-8 and INT 70H on page 2-121)
4B to 5F	– Reserved –
60 to 67	– Reserved for User Program Interrupts –
68 to 6F	– Reserved –
70	Real-Time Clock (See page 2-121)
71 to 74	– Reserved –
75	Redirect to NMI Interrupt (See INT 02H on page 2-5)
76 to 7F	– Reserved –
80 to 85	– Reserved –
86 to F0	Used by BASIC Interpreter when Running BASIC
F1 to FF	– Reserved for User Program Interrupts –

Figure 2-1 (Part 2 of 2). Interrupts

Interrupt 02H - Nonmaskable Interrupt (NMI)

For PC*jr* the nonmaskable interrupt (NMI) is attached to the keyboard interrupt.

For PC, PC XT, AT, and Personal System/2 Model 25 and Model 30, this interrupt handler displays PARITY CHECK 1 indicating a parity error occurred on the system board, or PARITY CHECK 2 indicating a parity error occurred on the I/O channel (assumes I/O channel memory). This interrupt handler attempts to find the storage location containing the bad parity, and if it is found, the segment address is displayed. If no parity error is found, **?????** appears in place of the address, indicating an intermittent read problem.

For Personal System/2 products except Model 25 and Model 30, the above paragraph applies except PARITY CHECK 1 and PARITY CHECK 2 are replaced by error codes, **110** and **111**, respectively. In addition, the NMI detects two other errors. The error codes are as follows:

110	System Board Memory Failure
111	I/O Channel Check Activated (assumes I/O channel memory)
112	Watchdog Time-Out
113	Direct Memory Access (DMA) Bus Time-Out

When the Watchdog Time-Out is enabled and a missing timer interrupt (IRQ 0) is detected, the system generates the NMI. If this occurs, the NMI interrupt handler displays **112**, indicating an expected timer interrupt was missed. Also, when a DMA-driven device uses the bus longer than the allowed 7.8 microseconds, the central arbitration control point generates the NMI and **113** is displayed, indicating a DMA bus time-out has occurred.

When an NMI occurs, the central arbitration control point is implicitly disabled. The NMI interrupt handler explicitly reenables the central arbitration control point by writing a 00H to port 90H.

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.

Notes:

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

Interrupt 05H - Print Screen

This interrupt handler prints the screen to printer 1. When INT 05H is issued, the cursor position is saved and is restored upon completion of the interrupt. INT 05H runs with interrupts enabled. Additional print screen requests are ignored when a print screen is already in progress. An initial status error from the printer ends the print request. Data area address hex 50:00 contains the status of the print screen. The supported status values for hex 50:00 are as follows:

00	Print Screen not called or, on return, operation successfully completed
01	Print Screen in progress, ignore request
FF	Error encountered during printing

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

Interrupt 08H - System Timer

This interrupt handler controls the timer interrupt from channel 0 of the system timer. The input frequency is 1.19318 MHz and the divisor is 65536, resulting in approximately 18.2 interrupts every second.

The interrupt handler:

- Maintains a count of interrupts at data area address hex 40:6C (timer counter) since power-on that may be used to establish time of day. After 24 hours of operation, hex 40:70 (timer overflow) is increased (made non 0).
- Decrements 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 hex 40:3F (motor status).
- Calls a user routine through software interrupt 1CH every timer tick.

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

Interrupt 09H - Keyboard

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

For ASCII keys, when a make code is read from port 60H, the character code and scan code are placed in the 32-byte keyboard buffer that begins at data area address hex 40:1E, at the address pointed to by hex 40:1C (keyboard buffer tail pointer). 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, the BIOS data areas hex 40:17 and hex 40:18 (keyboard control) and hex 40:96 (keyboard mode state and type flags) are updated.

The Ctrl-Alt-Del sequence causes the handler to set hex 40:72 (reset flag) to hex 1234 (bypass memory test), then jump to the power-on self-test (POST). The POST checks hex 40:72 (reset flag) and does not retest memory if it finds hex 1234. For PC Convertible, instead of a jump to POST, a processor reset is done, causing POST to execute.

The Pause key sequence causes the handler to loop until a valid ASCII keystroke is pressed. The PC Convertible issues INT 15H, (AH) = 41H (Wait on External Event) to wait for a valid ASCII keystroke.

The print screen key sequence issues an INT 05H (Print Screen).

The Ctrl-Break sequence issues an INT 1BH (Keyboard Break).

For PC XT BIOS dated 1/10/86 and after, AT, PC XT Model 286, PC Convertible, and Personal System/2 products, System Request causes the handler to issue an INT 15H, (AH) = 85H (System Request Key Pressed) to inform the system of a System Request key make or break operation. Also, the keyboard interrupt issues an INT 15H, (AH) = 91H (Interrupt Complete) with (AL) = 02H (Type = Keyboard), indicating that a keystroke is available.

For AT BIOS dated 6/10/85 and after, PC XT Model 286, PC Convertible, and Personal System/2 products, INT 15H, (AH) = 4FH (Keyboard Intercept), is issued after reading the scan code from port 60H. This allows the system to replace or absorb the scan code. End of Interrupt (EOI) processing is done upon return.

Interrupt 10H - Video

The following is a summary of the video functions of INT 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 – Alternate Select
(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 2-2. INT 10H - Video Functions

(AH) = 00H - Set Mode

```
(AL) - Requested video mode
```

The following table describes the supported video modes:

Mode (Hex)	**Type**	**Maximum Colors**	**Alpha Format**	**Buffer Start**
0, 1	A/N	16	40x25	B8000
2, 3	A/N	16	80x25	B8000
4, 5	APA	4	40x25	B8000
6	APA	2	80x25	B8000
7	A/N	Mono	80x25	B0000
8	APA	16	20x25	B0000
9	APA	16	40x25	B0000
A	APA	4	80x25	B0000
B, C	—Reserved—			
D	APA	16	40x25	A0000
E	APA	16	80x25	A0000
F	APA	Mono	80x25	A0000
10	APA	16	80x25	A0000
11	APA	2	80x30	A0000
12	APA	16	80x30	A0000
13	APA	256	40x25	A0000

APA — All Points Addressable (Graphics)
A/N — Alphanumeric (Text)

Figure 2-3. Video Modes

The following table lists hardware specific video mode characteristics:

Mode (Hex)	**Display Size**	**Box Size**	**Supporting IBM Products**	**Maximum Pages**
0, 1	320x200	8x8	PC*jr*, 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	PC*jr*, 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	PC*jr*, CGA, EGA, and Personal System/2 products	1
6	640x200	8x8	PC*jr*, 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

Figure 2-4 (Part 1 of 2). Hardware Specific Video Mode Characteristics

Mode (Hex)	Display Size	Box Size	Supporting IBM Products	Maximum Pages
8	160x200	8x8	PC*jr*	1
9	320x200	8x8	PC*jr*	1
A	640x200	8x8	PC*jr*	1
B, C	—Reserved—			
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

Figure 2-4 (Part 2 of 2). Hardware Specific Video Mode Characteristics

Notes:

1. PC*jr* and IBM Color/Graphics Monitor Adapter (CGA):

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

2. IBM Enhanced Graphics Adapter (EGA):

   a. The cursor is not displayed in graphics (APA) modes.
   b. Modes 0, 2, and 5 are identical to modes 1, 3, and 4 except color burst is not enabled. Color burst on enables color information on composite displays. Color burst off disables

color information on composite displays. RGB displays are not affected by the state of color burst.

c. The power-on default mode is based on switch settings on the adapter.
d. During mode set, if bit 7 of (AL) is set, the video buffer is not cleared.

See BIOS data area address hex 40:A8 on page 3-13 for save pointer dynamic overrides.

3. PC Convertible:

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

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

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

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

   a. The cursor is not displayed in graphics (APA) modes.
   b. Modes 0, 2, and 5 are identical to modes 1, 3, and 4.
   c. The power-on default mode with an analog color display attached is 3.
   d. The power-on default mode with an analog monochrome display attached is 7.

e. During mode set, if bit 7 of (AL) is set, the video buffer is not cleared.
f. For all modes except mode 13, the first 64 color registers are initialized and the values in the remaining 192 color registers are undefined.
g. Refer to (AH) = 12H, (BL) = 30H to select alpha mode scan lines (200, 350, or 400.)

See BIOS data area address hex 40:A8 on page 3-13 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. The BIOS maintains only one cursor type for all video pages.
2. For Personal System/2 Model 25 and Model 30, before writing to the hardware video ports, (CH) is multiplied by 2, and (CL) is multiplied by 2 and increased by 1.

(AH) = 02H - Set Cursor Position

```
(DH,DL) - Row, column (0,0 is upper left)
(BH) - Page number (0-based), see Figure 2-4 on page 2-13 for
       maximum pages
```

(AH) = 03H - Read Cursor Position

```
(BH) - Page number (0-based), see Figure 2-4 on page 2-13 for
       maximum pages

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

(AH) = 04H - Read Light Pen Position

For PC Convertible and Personal System/2 products:

```
On Return:
  (AH) = 00H - Light pen is not supported
  (BX, CX, DX) are altered on return
```

For all others:

```
On Return:
  (AH) = 00H - Light pen switch not activated
  (BX, CX, DX) are altered on return

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

(AH) = 05H - Select Active Display Page

For PC*jr*:

```
(AL) = 80H - Read cathode ray tube (CRT) and microprocessor
             page registers

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

(AL) = 82H - Set CRT page register
     (BH) - CRT page register

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

On Return for all:
  (BH) - CRT page register
  (BL) - Microprocessor page register
```

For all others:

```
(AL) - New page number (0-based), see Figure 2-4 on page 2-13 for
       maximum pages
```

(AH) = 06H - Scroll Active Page Up

```
(AL) - Number of lines blanked at bottom of window
     = 00H - Blank entire window
(CH,CL) - Row, column of upper left corner of scroll
(DH,DL) - Row, column of lower right corner of scroll
(BH) - Attribute to use on blank line
```

(AH) = 07H - Scroll Active Page Down

```
(AL) - Number of input lines blanked at top of window
     = 00H - Blank entire window
(CH,CL) - Row, column of upper left corner of scroll
(DH,DL) - Row, column of lower right corner of scroll
(BH) - Attribute to use on blank line
```

(AH) = 08H - Read Attribute/Character at Current Cursor Position

```
(BH) - Page number (0-based), see Figure 2-4 on page 2-13 for
       maximum pages

On Return:
  (AL) - Character read
  (AH) - Attribute of character read (alpha modes only)
```

(AH) = 09H - Write Attribute/Character at Current Cursor Position

For the read/write character interface while in graphics modes 4, 5, and 6, the characters are formed from a character generator maintained in the system ROM that contains only the first 128 characters. To read or write the second 128 characters, initialize the pointer at INT 1FH (location 0007CH) to point to the 1KB (KB = 1,024 bytes) table containing the code points for the second 128 characters (128-255). For all other graphics modes, 256 graphics characters are supplied in the system ROM.

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

```
(BH) - Page number (0-based), see Figure 2-4 on page 2-13 for
       maximum pages
(CX) - Count of characters to write
(AL) - Character to write
(BL) - Attribute of character (alpha)/color of
       character (graphics)
```

Notes:

1. Functions (AH) = 09H and (AH) = 0AH are similar. Use (AH) = 09H for graphics modes.
2. For graphics modes, if bit 7 of (BL) = 01H, then color value is exclusive ORed with current video memory (except in mode 13H).
3. For mode 13H, the value passed in (BH) is used as the background color.

(AH) = 0AH - Write Character at Current Cursor Position

```
(BH) - Page number (0-based), see Figure 2-4 on page 2-13 for
       maximum pages
(CX) - Count of characters to write
(AL) - Character to write
```

Note: Use (AH) = 09H for graphics modes.

(AH) = 0BH - Set Color Palette

```
(BH) - Color ID being set (0 to 1)
(BL) - Color value to be used with color ID


(BH) = 00H - Set background color for 320x200 graphics modes
           - Set border color for alphanumeric modes
           - Set foreground color for 640x200 graphics
(BL) = (0 to 31)


(BH) = 01H - Select palette for 320x200 graphics
(BL) = 0 - Green (1)/red (2)/brown (3)
     = 1 - Cyan (1)/magenta (2)/white (3)
```

Notes:

1. This interface has meaning for 320x200 graphics only.
2. In 40x25 or 80x25 alpha modes, the value set for palette color 0 indicates the border color to use (0 to 31), where values 16 to 31 select the high-intensity background set.
3. For EGA and Personal System/2 products, when in 640x200 graphics mode and color ID = 0, the background color is set.

(AH) = 0CH - Write Dot

```
(DX) - Row number
(CX) - Column number
(AL) - Color value
```

Note: If bit 7 of (AL) = 01H, then the color value is exclusive ORed with the current contents of the dot (except in mode 13H).

For graphics modes supporting more than one page:

```
(BH) - Page number (0-based), see Figure 2-4 on page 2-13 for
       maximum pages
```

(AH) = 0DH - Read Dot

```
(DX) - Row number
(CX) - Column number
```

For graphics modes supporting more than one page:

```
(BH) - Page number (0-based), see Figure 2-4 on page 2-13 for
       maximum pages

On Return:
  (AL) returns dot read
```

(AH) = 0EH - Write Teletype to Active Page

```
(AL) - Character to write
(BL) - Foreground color in graphics mode
```

Notes:

1. The screen width is controlled by the mode currently set.
2. Carriage Return, Line Feed, Backspace and Bell are treated as commands rather than 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:
  (AL) - Mode currently set
         [see (AH) = 00H for explanation]
  (AH) - Number of character columns on screen
  (BH) - Current active page number (0-based), see
         Figure 2-4 on page 2-13 for maximum pages
```

(AH) = 10H - Set Palette Registers

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

```
(AL) = 00H - Set individual palette register
     (BL) - Palette register to set
     (BH) - Value to set


(AL) = 01H - Set overscan register
     (BH) - Value to set


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


(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 read (range 0 to 15)

On Return:
     (BH) - Value read


(AL) = 08H - Read overscan register

On Return:
     (BH) - Value 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
             Bytes 0 to 15 - Palette values
             Byte 16 - Overscan value


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

(AL) = 11H - Reserved


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


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


     (BL) = 01H - Select page
          (BH) - Page number (0-based), see Figure 2-4 on page 2-13
                 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
```

Note: Function (AH) = 00H (Set Mode) 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 set. When using page selection, initialize alternate blocks of the color registers.

```
(AL) = 14H - Reserved

(AL) = 15H - Read individual color register

     (BX) - Color register to read


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


(AL) = 16H - Reserved


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

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


(AL) = 18H to 19H - Reserved


(AL) = 1AH - Read color page state

On Return:
  (BL) - Current paging mode
  (BH) - Current page
```

Note: See (AL) = 13H on page 2-22 for paging modes and page information.

```
(AL) = 1BH - Sum color values to gray shades

     (BX) - First color register to sum
     (CX) - Number of color registers to sum
```

Note: This call reads red, green, and blue values found in 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 (original data is not retained).

For Personal System/2 Model 25 and Model 30:

```
(AL) = 00H
     (BX) = 0712H - Color registers 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 set
     (DH) - Red value to set
     (CH) - Green value to set
     (CL) - Blue value to set

(AL) = 11H - Reserved

(AL) = 12H - Set block of color registers
     (ES:DX) - Pointer to table of color values
               Table format: red, green, blue, red,
               green, blue

     (BX) - First color register to set
     (CX) - Number of color registers to set


(AL) = 13H to 14H - Reserved


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

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


(AL) = 16H - Reserved


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

     (BX) - First color register to read
     (CX) - Number of color registers to read

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 sum
     (CX) - Number of color registers to sum
```

Note: This call reads red, green, and blue values found in color registers, performs a weighted sum (30% red + 59% green + 11% blue), then writes result into each red, green, and blue component of the color register (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 set, completely resetting the video environment but maintaining the regenerator buffer.

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


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

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


(AL) = 03H - Set block specifier (valid in alpha modes)
     (BL) - Character generator block selects
          Character attribute byte, bit 3 = 0:
            (BL) bits 1, 0 select a block from blocks 0 to 3
          Character attribute byte, bit 3 = 1:
            (BL) bits 3, 2 select a block from blocks 0 to 3
```

For example:

- To set a 256-character set using block 3, set (BL) = 0FH; this selects a single block. Character attribute bit 3 turns foreground intensity on or off.

- To specify a 512 character set as active using blocks 0 and 3, set (BL) = 0CH; this selects block 0 when character attribute bit 3 = 0, and block 3 when character attribute bit 3 = 1.

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

When 512 characters are active, a function call with (AX) = 1000H and (BX) = 0712H is recommended to set the color planes with eight consistent colors.

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

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

   ```
   INT [(200 or 350) / points] - 1
   ```

4. The length of the regenerative buffer is calculated as follows:

   ```
   (Number of rows on screen) x (Number of columns on screen) x 2
   ```

5. The CRT controller registers are reprogrammed as follows:

   ```
   R09H = Points - 1                                   Maximum scan line
   R0AH = Points - 2                                   Cursor start
   R0BH = Points - 1                                   Cursor end
   R12H = [(Number of rows on screen) x Points] - 1   Vertical display end
   R14H = Points - 1                                   Underline location.
          (Done in mode 7H only)
   ```

Note: The preceding register calculations must be close to the original table values or the results may be unpredictable.

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


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

(AL) = 12H - ROM 8x8 double dot
     (BL) - Block to load
```

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


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


(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 set is issued, or the results may not be predictable.

```
(AL) = 30H - Information
     (BH) - Font pointer
          = 00H - Return current INT 1FH pointer
          = 01H - Return current INT 44H pointer
          = 02H - Return ROM 8x14 font pointer
          = 03H - Return ROM double dot pointer
          = 04H - Return ROM double dot pointer (top)
          = 05H - Return ROM alpha alternate 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 alpha load
     (ES:BP) - Pointer to user table
     (CX) - Count to store
     (DX) - Character offset into table
     (BL) - Block to load
     (BH) - Number of bytes per character
```

```
(AL) = 01H - ROM 8x14 font
     (BL) - Block to load

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


(AL) = 03H - Set block specifier (valid in alpha modes)
     (BL) - Character generator block selects
          Character attribute byte bit 3 = 0:
            (BL) bits 4, 1, 0 select a block from blocks 0 to 7
          Character attribute byte bit 3 = 1:
            (BL) bits 5, 3, 2 select a block from blocks 0 to 7
```

For example:

- To set a 256-character set using block 6, set (BL) = 03AH; this selects a single block. Character attribute bit 3 turns foreground intensity on or off.
- To specify a 512-character set as active using blocks 0 and 6, set (BL) = 028H; this selects block 0 active when character attribute bit 3 = 0, and block 6 active when character attribute bit 3 = 1.

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

When 512 characters are active, a function call with (AX) = 1000H and (BX) = 0712H is recommended to set color planes with eight consistent colors.

```
(AL) = 04H - ROM 8x16 Font
     (BL) - Block to load
```

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

1. Page 0 is active.
2. Points (bytes per character) are recalculated.
3. Rows are calculated as follows:

   ```
   INT[(200, 350, or 400) / points] - 1
   ```

4. The length of the regenerative buffer is calculated as follows:

   ```
   (Number of rows on screen) x (Number of columns on screen) x 2
   ```

5. The CRT controller registers are reprogrammed as follows:

```
R09H = Points - 1                         Maximum scan line
R0AH = Points - 2                         Cursor start
R0BH = Points - 1                         Cursor end
R12H = Vertical displacement end
     For 350 and 400 scan line modes:
       [(Number of rows on screen) x Points] - 1
     For 200 scan line modes:
       {[(Number of rows on screen) x Points] x 2} - 1
R14H = Points - 1                         Underline location
       (Done in mode 7H only)
```

Note: The preceding register calculations must be close to the original table values or the results may be unpredictable.

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


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

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


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


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


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


(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 set is issued, or the results may not be predictable.

```
(AL) = 30H - Information
     (BH) - Font pointer
          = 00H - Return current INT 1FH pointer
          = 01H - Return current INT 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 alternate
          = 06H - Return ROM 8x16 pointer
          = 07H - Return ROM 9x16 font alternate

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 alpha load
     (ES:BP) - Pointer to user table
     (CX) - Count to store
     (DX) - Character offset into table
     (BL) - Block to load
     (BH) = 16 bytes per character for 400 scan lines
```

Note: If (BH) = 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 called, (AL) = 04H executed]

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

```
(AL) = 03H - Set block specifier (valid in alpha modes)
     (BL) - Character generator block selects
            Character attribute byte bit 3 = 0:
              (BL) bits 1, 0 select a block from blocks 0 to 3
            Character attribute byte bit 3 = 1:
              (BL) bits 3, 2 select a block from blocks 0 to 3
```

For example:

- To specify a 256-character set active using block 2, set (BL) = 0AH; this selects a single block. Character attribute bit 3 turns foreground intensity on or off.
- To specify a 512-character set active using blocks 0 and 2, set (BL) = 08H; this selects block 0 active when character attribute bit 3 = 0, and block 2 active when character attribute bit 3 = 1.

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

When 512 characters are active, a function call with (AX) = 1000H and (BX) = 0712H is recommended to set color registers, resulting in eight consistent colors.

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

```
(AL) = 04H - ROM 8x16 font
     (BL) - Block to load
```

The following register values are reserved. Calls to (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 called - (AL) = 00H executed]

(AL) = 11H - Reserved
     [if called - (AL) = 01H executed]

(AL) = 12H - Reserved
     [if called - (AL) = 02H executed]
```

```
(AL) = 14H - Reserved
     [if called - (AL) = 04H executed]


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


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


(AL) = 22H - Reserved
     [if called, (AL) = 24H 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 set is issued, or the results may not be predictable.

```
(AL) = 30H - Information
     (BH) - Font pointer
          = 00H - Return current INT 1FH pointer
          = 01H - Return current INT 43H pointer
          = 02H - Reserved (if called, ROM 8x16 pointer 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 - Alternate Select

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 alternate 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 next mode set)
     (AL) = 0 - 200 scan lines
          = 1 - 350 scan lines
          = 2 - 400 scan lines


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


(BL) = 31H - Default palette loading during set mode
     (AH) = 00H
     (AL) = 0 - Enable default palette loading
          = 1 - Disable default palette loading


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

Note: The EGA 16-palette registers, the overscan register, and the 256 color registers are not altered during any mode set when in the disabled state.

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


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

Note: The decode for the video I/O port addresses and the regenerator buffer addresses is enabled/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 supported
```

Note: When enabled, summing occurs during (AH) = 00H (Set Mode) color register loading and (AH) = 10H (Set Palette Registers).

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


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

Note: When enabled, the requested start/end value passed to (AH) = 01H (Set Cursor Type), is scaled to the current character height. The power-on default is to enable cursor emulation.

For Personal System/2 Model 25 and Model 30:

```
(BL) = 20H - Select alternate print screen routine

(BL) = 30H - Reserved

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

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


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


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


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


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

(BL) = 34H - Reserved
```

Note: When enabled, summing occurs during (AH) = 00H (Set Mode) color register loading, and on (AH) = 10H (Set Palette Registers).

For Personal System/2 products:

```
(BL) = 35H  Display switch

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


On Return for all:
  (AL) = 12H - Function supported
```

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

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

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 conflict between the adapter video and the system board video, the adapter video function is the primary video source. The system board video function remains disabled until display switching is enabled.

The following steps initiate display switching:

1. Initial adapter video off, (AL) = 00H
2. Initial system board video on, (AL) = 01H.

The initiate display switching steps are valid only the first time switching is initiated. After the initiation steps, switching between the system board and adapter displays is done through the switch-off active video request, (AL) = 02H and the switch-on inactive video request, (AL) = 03H.

> For a switch-off active video request, (AL) = 02H, the currently active video function and display are disabled. The switch state buffer saves the video state information. This state information is required when reactivation of this display is desired through a switch-on inactive video request, (AL) = 03H.
>
> For a switch-on inactive video request, (AL) = 03H, the currently inactive video function and display are enabled. The switch state buffer restores the video state information. This state information was saved on a previous switch-off active video request, (AL) = 02H, for this display.

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

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


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

For all others, no action is performed.

(AH) = 13H - Write String

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

```
(ES:BP) - Pointer to string to write
(CX) - Character-only count
(DX) - Position to begin string, in cursor terms
(BH) - Page number (0-based), see Figure 2-4 on page 2-13 for
       maximum pages


(AL) = 00H
     (BL) - Attribute
            String - (Char, char, char, ...); Cursor not moved

(AL) = 01H
     (BL) - Attribute
            String - (Char, char, char, ...); Cursor is moved

(AL) = 02H
       String - (Char, attr, char, attr, ...)
       Cursor not moved, valid for alpha modes only

(AL) = 03H
       String - (Char, attr, char, attr, ...)
       Cursor is moved, valid for alpha modes only
```

Note: Carriage Return, Line Feed, Backspace, and Bell are treated as commands rather than 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
     (ES:DI) - Point to character font within user table where
                  loading starts
     (CX) - Number of characters to store (1 to 256) value
            checked
     (DX) - Character offset into RAM font area
     (BL) = 00H - Load main font (block 0)
          = 01H - Load alternate font (block 1)
          = 02H to FFH - No operation
     (BH) - Number of bytes per character (1 to 255) value
            checked


(AL) = 01H - Load system ROM default font
     (BL) = 00H - Load main font (block 0)
          = 01H - Load alternate 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 alternate 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) - Alternate display adapter type
       = 0 - No alternate 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 PC Convertible has defined the following display types:

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 2-5. PC Convertible Display Types

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 supported (see display codes on page
               2-40)
       (BL) - Active display code
       (BH) - Alternate display code


(AL) = 01H - Write display combination code (see display codes
             on page 2-40)
     (BL) - Active display code
     (BH) - Alternate display code


On Return:
  (AL) = 1AH - Function supported
```

```
Display Codes:

00H - No display
01H - Monochrome with 5151 (monochrome)
02H - CGA with 5153/4 (color)
03H - Reserved
04H - EGA with 5153/4 (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
      analog monochrome
08H - Personal System/2 products except Model 25 and Model 30 with
      analog color
09H to 0AH - Reserved
0BH - Personal System/2 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
-1  - Unknown
```

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 supported


For implementation type 00H:

(BX) = 00H
(ES:DI) = Buffer of size 40H 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 2-12 for
                supported modes]
(DI+05H) word - Columns on screen (character columns on screen)

(DI+07H) word - Length of regenerator buffer (bytes)
(DI+09H) word - Starting address in regenerator 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 - CRT 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 (alternate)

(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 block specifier [see (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 block specifier [see (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 0
                          for Personal System/2 Model 25 and Model 30)
              Bit 3 = 1 - Mode set 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 0
                          for Personal System/2 Model 25 and Model 30)

(DI+2EH) byte - Reserved
(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 - Alpha 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 = Mode 07H
              Bit 6 = Mode 06H
              Bit 5 = Mode 05H
              Bit 4 = Mode 04H
              Bit 3 = Mode 03H
              Bit 2 = Mode 02H
              Bit 1 = Mode 01H
              Bit 0 = Mode 00H


   (01H) byte - Video modes
              Bit 7 = Mode 0FH
              Bit 6 = Mode 0EH
              Bit 5 = Mode 0DH
              Bit 4 = Mode 0CH
              Bit 3 = Mode 0BH
              Bit 2 = Mode 0AH
              Bit 1 = Mode 09H
              Bit 0 = Mode 08H


   (02H) byte - Video modes
              Bits 7 to 4 - Reserved
              Bit 3 = Mode 13H
              Bit 2 = Mode 12H
              Bit 1 = Mode 11H
              Bit 0 = Mode 10H

   See (AH) = 00H on page 2-12 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 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 for character block loading interfaces.
```

```
(0AH) byte - Miscellaneous functions
           Bit 7 = Color paging [see (AH) = 10H]
                   (Always 0 for Personal System/2 Model 25 and
                   Model 30)
           Bit 6 = Color palette [see (AH) = 10H]
           Bit 5 = EGA palette [see (AH) = 10H]
           Bit 4 = Cursor emulation [see (AH) = 01H]
           Bit 3 = Mode set default palette loading [see
                   (AH) = 12H]
           Bit 2 = Character font loading [see (AH) = 11H]
           Bit 1 = Summing [see (AH) = 10H and (AH) = 12H]
           Bit 0 = All modes on all displays (Always 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]
           Bit 2 = Background intensity/blinking control [see
                   (AH) = 10H]
           Bit 1 = Save/restore [see (AH) = 1CH]
                   (Always 0 for Personal System/2 Model 25 and
                   Model 30)
           Bit 0 = Light pen [see (AH) = 04H]


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


(0EH) byte - Save pointer functions
           Bits 7, 6 = Reserved
           Bit 5 = DCC extension (Always 0 for
                   Personal System/2 Model 25 and Model 30)
           Bit 4 = Palette override
           Bit 3 = Graphics font override
           Bit 2 = Alpha 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 2-45)


On Return:
  (AL) = 1CH - Function 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 2-45)
     (ES:BX) = Buffer pointer to save state


On Return:
  (AL) = 1CH - Function supported

       Requested states saved


(AL) = 02H - Restore state
     (CX) - Requested states (see supported save/restore states
            on page 2-45)
     (ES:BX) - Buffer pointer to restore state


On Return:
  (AL) = 1CH - Function supported

       Requested states 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 the optional devices that are attached to the system. BIOS data area hex 40:10 (installed hardware) is set during the POST as follows:

```
On Return:
  (AX) - Equipment flags
       Bits 15,14 - Number of printers attached
       Bit 13 - Internal modem installed
       Bit 12 - Not used
       Bits 11,10,9 - Number of RS-232C cards attached
       Bit 8 - Not used
       Bits 7,6 - Number of diskette drives, if bit 0 = 1
                  (values are binary)
                = 00 - 1 drive
                = 01 - 2 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 - Pointing device installed
       Bit 1 = Math coprocessor installed
       Bit 0 = IPL diskette installed
```

Interrupt 12H - Memory Size Determination

This routine returns the amount of RAM up to 640KB in the system as determined by the POST, minus the memory allocated to the Extended BIOS Data Area. See INT 15H, (AH) = C1H (Return Extended BIOS Data Area Segment Address) on page 2-94, and INT 15H, (AH) = 88H (Extended Memory Size Determine) on page 2-87 for additional information.

The following assumptions are made during memory size determination:

- All installed memory is functional
- All memory from 0 to 640KB is contiguous.

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

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 DASD Type
(AH) = 16H	–	Diskette Change Line Status
(AH) = 17H	–	Set DASD Type for Format
(AH) = 18H	–	Set Media Type for Format
(AH) = 19H to FFH	–	Reserved

Figure 2-6. INT 13H - Diskette Functions

Note: For the diskette drive parameters see "Diskette Drive Parameter Table" on page 3-26.

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

Prior to waiting for the diskette interrupt, BIOS calls INT 15H, (AH) = 90H (Device Busy) with (AL) = 01H (Type = Diskette). This informs the operating system of the wait. The complementary INT 15H, (AH) = 91H (Interrupt Complete) with (AL) = 01H (Type = Diskette) is called indicating the operation is complete. See "Multitasking Provisions" on page 4-16 for additional information.

(AH) = 00H - Reset Diskette System

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


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0

  (AH) - Status of operation
       = 80H - Diskette drive not ready
       = 40H - Seek operation failed
       = 20H - General controller failure
       = 10H - Cyclic redundancy check (CRC) error on diskette
               read
       = 0CH - Media type not found
       = 09H - Attempt to DMA across a 64KB boundary
       = 08H - DMA overrun on operation
       = 06H - Diskette change line active
       = 04H - Requested sector not found
       = 03H - Write protect error
       = 02H - Address mark not found
       = 01H - Invalid diskette parameter
       = 00H - No error

Diskette status at hex 40:41 - Status of operation
```

Notes:

1. If an error is reported by the diskette BIOS, reset the diskette system and retry the operation.
2. If (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 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:
  CF = 1 - Status is non 0
     = 0 - Status is 0
  (AH) - Status of operation (see values for the status of
         operation on page 2-49)
```

(AH) = 02H - Read Desired Sectors into Memory

```
(DL) - Drive number (0-based)
     Bit 7 = 0 - Diskette (value checked)
(DH) - Head number (not value checked, 0-based)
(CH) - Track number (not value checked, 0-based)
(CL) - Sector number (not value checked)
(AL) - Number of sectors (not value checked)
(ES:BX) - Address of buffer


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0
  (AL) - Number of sectors actually transferred
  (AH) - Status of operation (see values for the status of
         operation on page 2-49)

Diskette status at hex 40:41 - Status of operation
```

Note: If an error is reported by the diskette BIOS, reset the diskette system, then retry the operation.

(AH) = 03H - Write Desired Sectors from Memory

```
(DL) - Drive number (0-based)
     Bit 7 = 0 - Diskette (value checked)
(DH) - Head number (not value checked, 0-based)
(CH) - Track number (not value checked, 0-based)
(CL) - Sector number (not value checked)
(AL) - Number of sectors (not value checked)
(ES:BX) - Address of buffer


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0
  (AL) - Number of sectors actually transferred
  (AH) - Status of operation (see values for the status of
         operation on page 2-49)

Diskette status at hex 40:41 - Status of operation
```

Notes:

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

(AH) = 04H - Verify Desired Sectors

```
(DL) - Drive number (0-based)
     Bit 7 = 0 - Diskette (value checked)
(DH) - Head number (not value checked, 0-based)
(CH) - Track number (not value checked, 0-based)
(CL) - Sector number (not value checked)
(AL) - Number of sectors (not value checked)
(ES:BX) - Address of buffer


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0
  (AL) - Number of sectors verified
  (AH) - Status of operation (see values for the status of
         operation on page 2-49)

Diskette status at hex 40:41 - Status of operation
```

Notes:

1. If an error is reported by the diskette BIOS, reset the diskette system, then retry the operation.
2. ES:BX is not required for AT BIOS dated 11/15/85 and after, 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 four bytes:

```
Byte 0 - Track number
Byte 1 - Head number
Byte 2 - Sector number

Byte 3 - Number of bytes per sector
       = 00H - 128-bytes per sector
       = 01H - 256-bytes per sector
       = 02H - 512-bytes per sector
       = 03H - 1024-bytes per sector
```

There must be one entry for every sector on the track. This information is used to find the requested sector during read/write access. Prior to formatting a diskette, if there is more than one supported format for the drive in question, it is necessary to call (AH) = 17H (Set DASD Type for Format), or (AH) = 18H (Set Media Type for Format) to set the diskette type to be formatted.

```
(AL) - Number of sectors to format (not value checked)
(DL) - Drive number (0-based)
     Bit 7 = 0 - Diskette (value checked)
(DH) - Head number (not value checked, 0-based)
(CH) - Track number (not value checked, 0-based)
(ES:BX) - Address of buffer


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0
  (AH) - Status of operation (see values for the status of
         operation on page 2-49)

Diskette status at hex 40:41 - Status of operation
```

Notes:

1. If an error is reported by the diskette BIOS, reset the diskette system, then retry the operation.
2. The diskette parameter table is used to format the diskette. See "Diskette Drive Parameter Table" on page 3-26.
3. For PC XT Model 286, (AL) is not required.

(AH) = 06H to 07H - Reserved

(AH) = 08H - Read Drive Parameters

There is a parameter table for each supported media type.

For PC*jr*, PC, PC XT, and for AT BIOS dated 1/10/84:

```
On Return:
  CF = 1 - Error
  (AH) - Status of operation
       = 01H - Invalid command

Diskette status at hex 40:41 - Status of operation
```

For all others:

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


On Return:
(ES:DI) - Pointer to 11-byte parameter table
          associated with the maximum supported media type
          within the drive in question (see
          "Diskette Drive Parameter Table" on page 3-26.)
```

```
(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
(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

(AX) = 0

*KB = 1,024 bytes; MB = 1,048,576 bytes.
```

When the drive type is known but the CMOS type is invalid, CMOS is not present, CMOS battery is discharged or CMOS checksum is invalid, all registers are returned as above except (BL) = 0.

If the requested drive is not installed, then (AX), (BX), (CX), (DX), (DI), and (ES) = 0.

```
Diskette status hex 40:41 = 0 and CF = 0
```

For drive number 80H or above (indicating fixed disks):

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

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

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

```
Diskette status hex 40:41 = 0 and CF = 0
```

(AH) = 09H to 14H - Reserved

(AH) = 15H - Read DASD Type

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

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


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

Diskette status at hex 40:41 - Status of operation
```

For all others:

```
On Return:
  CF = 1 - Error
  (AH) - Status of operation
       = 01H - Invalid command

Diskette status at hex 40:41 - Status of operation
```

(AH) = 16H - Diskette Change Line Status

For AT, PC XT BIOS dated 1/10/86 and after, 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

Diskette status at hex 40:41 - (AH) on return
```

For all others:

```
On Return:
  (AH) - Status of operation
       = 01H - Invalid command
  CF = 1 - Error

Diskette status at hex 40:41 - Status of operation
```

(AH) = 17H - Set DASD Type for Format

The 'diskette change' signal is checked for all drives that support it. If found active, the logic attempts to reset 'diskette change' to the inactive state. If successful, the BIOS sets the data rate for format and returns the disk change error code. If the attempt fails, the logic returns the time-out error code and sets the DASD type to a predetermined state, indicating that the media type is unknown.

When the 'diskette change' signal is found active, as it is after a diskette is changed, this function is called again.

For PC XT BIOS dated 1/10/86 and after, AT, 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 through 0FFH - Invalid request


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0
  (AH) - Status of operation (see values for the status of
         operation on page 2-49)

Diskette status at hex 40:41 - Status of operation
```

For all others:

```
On Return:
  (AH) - Status of operation
       = 01H - Invalid command
  CF = 1 - Error

Diskette status at hex 40:41 - Status of operation
```

(AH) = 18H - Set Media Type for Format

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

There is one parameter table for each supported media type.

```
(DL) - Drive number (0-based)
     Bit 7 = 0 - Diskette (value checked)
(CH) - Number of tracks (low 8 bits, 0-based)
(CL) - Bits 7, 6 - Number of tracks (high two bits, 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
            "Diskette Drive Parameter Table" on page 3-26.)

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

  (AH) - Status of operation (see values for the status of
         operation on page 2-49)
```

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

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

For all others:

```
On Return:
  (AH) - Status of operation
       = 01H - Invalid command
  CF = 1 - Error

Diskette status at hex 40:41 - Status of operation
```

(AH) = 19H to FFH - Reserved

Interrupt 13H - Fixed Disk

This interface provides access to fixed disk drives. The following is a summary of the fixed disk functions of INT 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 Cylinder
(AH) = 06H	Format Desired Cylinder 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	Alternate 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	Format Unit
(AH) = 1BH to FFH	Reserved

Figure 2-7. INT 13H - Fixed Disk Functions

Notes:

1. If a fixed disk drive adapter is not installed, the code is not hooked into INT 13H. The returns are described in the diskette interface.

2. For the fixed disk interface, the drive number (DL) is value checked for all functions that use (DL).

3. For AT, PC XT Model 286, and Personal System/2 products, prior to waiting for interrupt, the BIOS calls INT 15H, (AH) = 90H (Device Busy) with (AL) = 00H (Type = Disk), informing the operating system of the wait. The complementary INT 15H, (AH) = 91H (Interrupt Complete) with (AL) = 00H (Type = Disk), is called indicating the operation is complete.

4. For Personal System/2 products, prior to waiting for the fixed disk reset the BIOS calls INT 15H, (AH) = 90H (Device Busy) with (AL)

= FCH (Type = Fixed Disk Reset). This is a time-out only function. There is no complementary Post operation. See "Multitasking Provisions" on page 4-16.

5. Bit 7 of the drive number must be set upon entry to the fixed disk BIOS.
6. For the drive parameters see "Fixed Disk Drive Parameter Table" on page 3-18.

(AH) = 00H - Reset Disk System

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


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0

  (AH) - Status of operation
       = 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 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
       = 20H - General controller failure
       = 40H - Seek operation failed
       = 80H - Time-out
       = BBH - Undefined error occurred
       = CCH - Write fault on selected drive
       = E0H - Status error/error register = 0
       = FFH - Sense operation failed
```

Notes:

1. Reset Disk System is issued only if the 7-bit drive number is less than or equal to the maximum number of fixed disk drives. The diskette system is also reset for all values of (DL).

2. For Personal System/2 products, prior to waiting for the fixed disk reset, the BIOS calls INT 15, (AH) = 90H (Device Busy) with (AL) = FCH (Type = Fixed Disk Reset) informing the operating system of the wait.

(AH) = 01H - Read Status of Last Operation

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


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0


  (AH) - Status of operation (see values for the status of
         operation on page 2-59)

Disk status is reset to 0
```

(AH) = 02H - Read Desired Sectors into Memory

```
(DL) - Drive number, bit 7 = 1 for fixed disk drive (0-based)
(DH) - Head number (0-based, not value checked)
(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)
(AL) - Number of sectors
(ES:BX) - Address of buffer


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0
  (AH) - Status of operation (see values for the status of
         operation on page 2-59)
```

Notes:

1. An 11H error indicates the data read had a recoverable error that was corrected by the ECC algorithm. The data may be good; however, the BIOS routine indicates an error to allow the controlling program to make this determination. The error may not recur if the data is rewritten.

2. If an error is reported by the fixed disk BIOS, reset the disk system, then retry the operation.

(AH) = 03H - Write Desired Sectors from Memory

```
(DL) - Drive number, bit 7 = 1 for fixed disk drive (0-based)
(DH) - Head number (0-based, not value checked)
(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)
(AL) - Number of sectors
(ES:BX) - Address of buffer


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0
  (AH) - Status of operation (see values for the status of
         operation on page 2-59)
```

Note: If an error is reported by the fixed disk BIOS, reset the disk system, then retry the operation.

(AH) = 04H - Verify Desired Sectors

```
(DL) - Drive number, bit 7 = 1 for fixed disk drive (0-based)
(DH) - Head number (0-based, not value checked)
(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)
(AL) - Number of sectors


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0
  (AH) - Status of operation (see values for the status of
         operation on page 2-59)
```

Note: If an error is reported by the fixed disk BIOS, reset the disk system, then retry the operation.

(AH) = 05H - Format Desired Cylinder

```
(DL) - Drive number, bit 7 = 1 for fixed disk drive (0-based)
(DH) - Head number (0-based, not value checked)
(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)
```

For PC XT:

```
(AL) - Contains interleave value

On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0

  (AH) - Status of operation (see values for the status of
         operation on page 2-59)
```

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 x (Sectors per cylinder) bytes contain F, N for each sector.
              F = 00H - Good sector
                = 80H - Bad sector
              N - Sector number


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0
  (AH) - Status of operation (see values for the status of
         operation on page 2-59)
```

For any device using ESDI-type commands:

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

Note: If an error is reported by the fixed disk BIOS, reset the disk system, then retry the operation.

(AH) = 06H - Format Desired Cylinder and Set Bad Sector Flags

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

For PC XT:

```
(DL) - Drive number, bit 7 = 1 for fixed disk drive (0-based)
(DH) - Head number (0-based, not value checked)
(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)
(AL) - Interleave value


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0
  (AH) - Status of operation (see values for the status of
         operation on page 2-59)
```

For AT, PC XT Model 286, Personal System/2 products, and any device using ESDI-type commands:

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

Note: If an error is reported by the fixed disk BIOS, reset the disk system, then retry the operation.

(AH) = 07H - Format Drive Starting at Desired Cylinder

For PC XT:

```
(DL) - Drive number, bit 7 = 1 for fixed disk drive (0-based)
(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)
(AL) - Interleave value


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0
  (AH) - Status of operation (see values for the status of
         operation on page 2-59)
```

For AT, PC XT Model 286, Personal System/2 products, and any device using ESDI-type commands:

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

Note: If an error is reported by the fixed disk BIOS, reset the disk system, then retry the operation.

(AH) = 08H - Read Drive Parameters

If the drive number is invalid then (AH) and hex 40:74 = 07H (last fixed disk drive operation status), (CX) and (DX) = 0, and CF is set. If no fixed disk drive is attached or no fixed disk drive adapter is installed, (AH) and hex 40:41 = 01H (last diskette drive operation status), and CF is set.

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


On Return:
  (DL) - Number of consecutive drives attached (1, 2; controller
         card 0 tally only)
  (DH) - Maximum value for head number (range 0-3FH)
  (CH) - Maximum value for cylinder number (range 0-3FFFH)
  (CL) - Maximum value for sector and high order 2 bits of
         cylinder numbers
```

(AH) = 09H - Initialize Drive Pair Characteristics

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


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0
  (AH) - Status of operation (see values for the status of
         operation on page 2-59)
```

For PC XT:

Interrupt 41H points to the parameter tables. Four entries in the PC XT table correspond to the switch settings 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 [80H ≤ (DL) ≤ 87H] then both drives 0 and 1 are initialized.

For all other values, an invalid command status is returned. If drive 0 initialization fails, drive 1 initialization is not attempted. If either attempt fails, hex 40:74 = 07H (last fixed disk drive operation status) and (AH) are updated with the appropriate error.

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

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, then drive 0 is initialized using interrupt 41H. If (DL) = 81H, then drive 1 is initialized using interrupt 46H. For all other values, an invalid command status is returned.

For any device using ESDI-type commands:

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

Note: If an error is reported by the fixed disk BIOS, reset the disk system, then retry the operation.

(AH) = 0AH to 0BH - Reserved

(AH) = 0CH - Seek

```
(DL) - Drive number, bit 7 = 1 for fixed disk drive (0-based)
(DH) - Head number (0-based, not value checked)
(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)


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0
   (AH) - Status of operation (see values for the status of
          operation on page 2-59)
```

Note: If an error is reported by the fixed disk BIOS, reset the disk system, then retry the operation.

(AH) = 0DH - Alternate Disk Reset

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


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0
  (AH) - Status of operation (see values for the status of
         operation on page 2-59)
```

Note: Alternate Disk Reset is issued only if the 7-bit drive number is less than or equal to the maximum number of fixed disk drives.

(AH) = 0EH to 0FH - Reserved

(AH) = 10H - Test Drive Ready

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


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0
  (AH) - Status of operation (see values for the status of
         operation on page 2-59)
```

(AH) = 11H - Recalibrate

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


On Return:
  CF = 1 - Status is non 0
     = 0 - Status is 0
  (AH) - Status of operation (see values for the status of
         operation on page 2-59)
```

Note: If an error is reported by the fixed disk BIOS, reset the disk system, then retry the operation.

(AH) = 12H to 14H - Reserved

(AH) = 15H - Read DASD Type

For PC XT:

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

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:
  (AH) = 00H - Drive not present or (DL) invalid
       = 01H - Reserved
       = 02H - Reserved
       = 03H - Fixed disk


  (CX,DX) - Number of 512-byte blocks
            If (AH) = 0 then (CX) and (DX) = 0

  CF = 0 - Operation successfully completed
```

(AH) = 16H to 18H - Reserved

(AH) = 19H - Park Heads

For PC XT, AT, and PC XT Model 286:

```
On Return:
  (AH) - Status of operation = 01H - invalid function request
  CF = 1 - Error
```

For Personal System/2 products:

```
(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 2-59)
  CF = 1 - Error
```

(AH) = 1AH - Format Unit

For any device using ESDI-type commands:

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

This command may take more than an hour to complete.

```
(AH) = 1AH - Format fixed disk drive

(AL) - Relative block address (RBA) defect table block count
     = 0 - No RBA table associated with this format request
     > 0 - RBA table used


(ES:BX) - Address of RBA table


(CL) - Modifier bits
     Bits 7, 6, 5 - Must be 0
     Bit 4 - Periodic interrupt.  The controller interrupts the
             host for every cylinder completed during each phase
             of the formatting operation.  This feature allows
             the host to display formatting progress.  The phase
             is defined as follows:

                0 - Reserved
                1 - Surface analysis
                2 - Formatting


             An INT 15H (AH) = 0FH, (AL) = Phase Code is executed
             by BIOS.  The return must clear CF to allow
             formatting to continue.  Set CF to end formatting.
             The host must keep a running count of interrupts
             for each phase.  This running count is the cylinder
             number.  The host may display formatting progress
             in granularities other than 1, although interrupts
             occur for every cylinder.

     Bit 3 - Perform extended surface analysis.  (A format with this
             bit set to 0 must have occurred before attempting to
             format with this bit set.)

     Bit 2 - Update secondary defect map.  This map is updated to
             reflect defects found during surface analysis and
             those passed with the format command.  If this bit
             is set with bit 1, the secondary defect map is
             replaced.

     Bit 1 - Ignore secondary defect map.  The secondary defect
             map is not processed.

     Bit 0 - Ignore primary defect map.  The primary defect map
             is not processed.

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

For all others, this function is reserved.

(AH) = 1BH to FFH - Reserved

Interrupt 14H - Asynchronous Communications

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

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

Figure 2-8. INT 14H - Asynchronous Communications Functions

(AH) = 00H - Initialize the Communications Port

```
(AL) - Parameters for initialization
     Bits 7, 6, 5 - Baud rate (values are binary)
                  = 000 - 110
                  = 001 - 150
                  = 010 - 300
                  = 011 - 600
                  = 100 - 1200
                  = 101 - 2400
                  = 110 - 4800
                  = 111 - 9600
                On Personal System/2 products, for baud rates above
                9600, see INT 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 - Word length (values are binary)
               = 10 - 7 Bits
               = 11 - 8 Bits

(DX) - RS-232C Communications line to use (0,1,2,3) corresponding
       to actual port base address at hex 40:00
```

```
On Return:
  (AL) - Modem status
       Bit 7 - Received line signal detect
       Bit 6 - Ring indicator
       Bit 5 - Data set ready
       Bit 4 - Clear to send
       Bit 3 - Delta receive line signal detect
       Bit 2 - Trailing edge ring detector
       Bit 1 - Delta data set ready
       Bit 0 - Delta clear to send


  (AH) - Line status
      Bit 7 - Time-out
      Bit 6 - Transmitter shift register empty
      Bit 5 - Transmitter holding register empty
      Bit 4 - Break detect
      Bit 3 - Framing error
      Bit 2 - Parity error
      Bit 1 - Overrun error
      Bit 0 - Data ready
```

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

(AH) = 01H - Send Character

```
(AL) - Character to send
(DX) - RS-232C communications line to use (0,1,2,3) corresponding
       to actual port base addresses at hex 40:00


On Return:
  (AL) is preserved
  (AH) - Line status (see values for the line status on page 2-70)
```

(AH) = 02H - Receive Character

```
(DX) - RS-232C communications line to use (0,1,2,3) corresponding
       to actual port base addresses at hex 40:00


On Return:
  (AL) - Character received
  (AH) - Line status (see values for the line status on page 2-70)
```

Note: The routine waits for the character.

(AH) = 03H - Read Status

```
(DX) - RS-232C communications line to use (0,1,2,3) corresponding
       to actual port base addresses at hex 40:00


On Return:
  (AL) - Modem status (see values for the modem status on page 2-70)
  (AH) - Line status (see values for the line status on page 2-70)
```

(AH) = 04H - Extended Initialize

For Personal System/2 products:

```
(DX) - RS-232C communications line to use (0,1,2,3) corresponding
       to actual port base addresses at hex 40:00

(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 - One
     = 01H - Two if 6-, 7-, or 8-bit word length
           - One-and-one-half if 5-bit word length

(CH) - Word 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


On Return:
  (AL) - Modem status (see values for the modem status on page 2-70)
  (AH) - Line status (see values for the line status on page 2-70)
```

For all others, no action is performed.

(AH) = 05H - Extended Communications Port Control

For Personal System/2 products:

```
(AL) = 00H - Read modem control register
     (DX) - RS-232C communications line to use (0,1,2,3)
            corresponding to actual port base addresses
            at hex 40:00


On Return:
  (BL) - Modem control register
       Bit 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
     (DX) - RS-232C communications line to use (0,1,2,3)
            corresponding to actual port base addresses
            at hex 40:00


     (BL) - Modem control register
          Bit 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

On Return:
  (AL) - Modem status (see values for the modem status on page 2-70)
  (AH) - Line status (see values for the line status on page 2-70)
```

For all others, no action is performed.

(AH) = 06H to FFH - Reserved

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 to 0EH – Reserved
(AH) = 0FH – Format Unit Periodic Interrupt
(AH) = 10H to 20H – Reserved
(AH) = 21H – Power-On Self-Test Error Log
(AH) = 22H 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 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 Determine
(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 FFH – Reserved

Figure 2-9. INT 15H - System Services Functions

(AH) = 00H - Turn Cassette Motor On

For PC*jr* and PC:

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

For all others:

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

(AH) = 01H - Turn Cassette Motor Off

For PC*jr* and PC:

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

For all others:

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

(AH) = 02H - Read Blocks from Cassette

For PC*jr* and PC:

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


On Return:
  (ES:BX) - Pointer to last byte read + 1
  (DX) - Count of bytes read
  CF = 0 - No error
     = 1 - Error
           For PCjr when CF = 1, (AH) contains:
             01H = CRC error
             02H = Lost data transitions
             04H = No data found
```

For all others:

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

(AH) = 03H - Write Blocks to Cassette

For PC*jr* and PC:

```
(ES:BX) - Pointer to data buffer
(CX) - Count of bytes to write


On Return:
  (ES:BX) - Pointer to last byte written + 1
  (CX) = 00H
  CF = 0 - No error
     = 1 - Error
           For PCjr when CF = 1, (AH) contains:
             01H = CRC error
             02H = Lost data transitions
             04H = No data found
```

For all others:

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

(AH) = 04H to 0EH - Reserved

(AH) = 0FH - Format Unit Periodic Interrupt

For any device using 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: Function (AH) = 0FH provides a hook to the caller upon completion of formatting or scanning each cylinder. If no handler is hooked, CF is set to 1 on return.

For PC*jr* and PC:

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

For all others:

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

(AH) = 10H to 20H - Reserved

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

For PC*jr* and PC:

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


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


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

For all others:

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

(AH) = 22H to 3FH - Reserved

(AH) = 40H - Read/Modify Profiles

For PC*jr* and PC:

```
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 - Profile execution failed
```

For all others:

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

(AH) = 41H - Wait for External Event

For PC*jr* and PC:

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

For PC Convertible:

```
(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 read for
          event determination (event type codes 11H
          through 14H)
(AL) - Event type code
     = 00H - Return after any event has occurred
     = 01H - Compare value, return if equal
     = 02H - Compare value, 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


On Return:
  CF = 1 - Time-out
```

Note: 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.

For all others:

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

(AH) = 42H - Request System Power-Off

For PC*jr* and PC:

```
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:

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

(AH) = 43H - Read System Status

For PC*jr* and PC:

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

For PC Convertible:

```
On Return:
  (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
  (AH) is modified
```

For all others:

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

(AH) = 44H - Activate/Deactivate Internal Modem Power

For PC*jr* and PC:

```
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 - Request failed
  CF = 0 - Operation successfully completed
     = 1 - Request failed
```

For all others:

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

(AH) = 45H to 4EH - Reserved

(AH) = 4FH - Keyboard Intercept

For PC*jr* and PC:

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

For PC XT BIOS dated 11/08/82 and AT BIOS dated 1/10/84:

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

For all others, the keyboard intercept (keyboard escape), is called by the INT 09H (keyboard) routine to allow the keystroke to be changed or absorbed. Normally, the system returns the scan code unchanged, but the operating system can point INT 15H to itself and do one of the following:

1. Replace (AL) with a different scan code and return with the carry flag set, effectively changing the keystroke.
2. Process the keystroke and return with the carry flag reset, causing the INT 09H routine to ignore the keystroke.

```
(AL) - Scan code
CF = 1

On Return:
  (AL) - New scan code
  CF = 1
       -or-
  (AL) - Unchanged scan code
  CF = 0
```

Note: To dynamically determine the products that support this feature, see INT 15H, (AH) = C0H (Return System Configuration Parameters) on page 2-92.

(AH) = 50H to 7FH - Reserved

(AH) = 80H - Device Open

For PC*jr* and PC:

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

For PC XT BIOS dated 11/08/82:

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

For all others:

```
(BX) - Device ID
(CX) - Process ID
```

(AH) = 81H - Device Close

For PC*jr* and PC:

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

For PC XT BIOS dated 11/08/82:

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

For all others:

```
(BX) - Device ID
(CX) - Process ID
```

(AH) = 82H - Program Termination

For PC*jr* and PC:

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

For PC XT BIOS dated 11/08/82:

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

For all others:

```
(BX) - Device ID
```

(AH) = 83H - Event Wait

For PC*jr* and PC:

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

For PC XT:

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

For AT BIOS dated 1/10/84:

```
(ES:BX) - Pointer to byte in caller's memory that has
          the high order bit set as soon as possible after
          interval expires.

(CX,DX) - Microseconds until posting
          (Granularity is 976 microseconds)


On Return:
  CF = 0 - Operation successfully completed
     = 1 - Operation unsuccessful, 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 as soon as possible after
          interval expires.

(CX,DX) - Microseconds until posting
          (Granularity is 976 microseconds)


On Return:
  CF = 0 - Operation successfully completed
     = 1 - Operation unsuccessful, function busy


(AL) = 01H - Cancel set interval


On Return:
  CF = 0 - Operation successfully completed
     = 1 - Operation unsuccessful, function busy

(Personal System/2 Model 25 and Model 30 always return with CF = 1)
```

(AH) = 84H - Joystick Support

For PC*jr*, PC, and PC Convertible:

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

For PC XT BIOS dated 11/08/82:

```
On Return:
  (AH) = 86H
  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) - A(x) value
  (BX) - A(y) value
  (CX) - B(x) value
  (DX) - B(y) value
  CF = 1 - Invalid call
```

(AH) = 85H - System Request Key Pressed

For PC*jr* and PC:

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

For PC XT BIOS dated 11/08/82:

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

For all others:

```
(AL) = 00H - Key make
(AL) = 01H - Key break
```

(AH) = 86H - Wait

For PC*jr* and PC:

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

For PC XT:

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

For all others:

```
(CX,DX) - Time before return to caller, in microseconds
          (Granularity is 976 microseconds)
CF = 0 - Successful wait
   = 1 - Wait function already in progress
```

(AH) = 87H - Move Block

For PC*jr* and PC:

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

For PC XT, PC Convertible, and PS/2™ Model 25 and Model 30:

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

For AT, PC XT Model 286, and PS/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.

```
(AH) = 87H - Block move.
(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)
          built by a routine using this function.
```

PS/2 is a trademark of the International Business Machines Corporation.

(ES:SI) points to a global descriptor table (GDT) built before interrupting to this function. The descriptors are used to perform the block move in the protected mode. The source and target descriptors 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 CPL0-R/W (93H). The 24-bit address (byte high, word low) must be set to the target/source.

Note: No interrupts are allowed during transfers. Large block moves may 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 are restored except (AH)


  If (AH) = 00H:
          CF = 0
          ZF = 1
  If (AH) = 01H to 03H:
          CF = 1
          ZF = 0
```

The following shows the organization of a block move GDT:

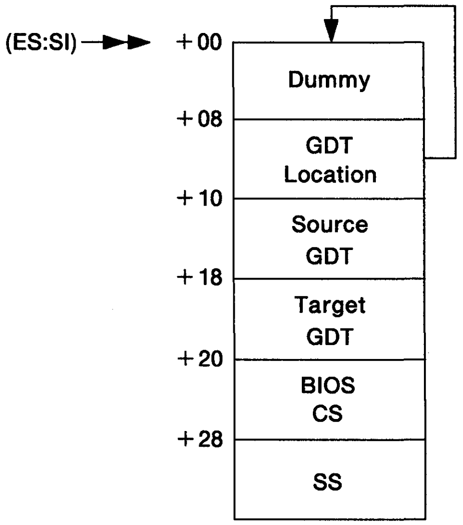


Figure 2-10. Block Move Global Descriptor Table

The descriptors are defined as follows:

- The first is the required dummy and is user initialized to 0.
- The second points to the GDT as a data segment. It is user initialized to 0 and can be modified by the BIOS.
- The third points to the source to be moved and is user initialized.
- The fourth points to the destination segment and is user initialized.
- The fifth is used by the BIOS to create the protected mode code segment. It is user initialized to 0 and can be modified by the BIOS.
- The sixth is used by the BIOS to create a protected mode stack segment. It is user initialized to 0, can be modified by the BIOS, and points to the user stack.

The following is a sample 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 (CPL0-R/W)
  Reserved          DW     0   ; Reserved word (must be 0)

SOURCE_TARGET_DEF     ENDS

The global descriptor table [actual location 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
```

(AH) = 88H - Extended Memory Size Determine

For PC*jr* and PC:

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

For PC XT, PC Convertible, and PS/2 Model 25 and Model 30:

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

For AT, PC XT Model 286, and PS/2 products except Model 25 and Model 30, this routine returns the amount of system memory beginning at address 100000H, as determined by the POST. The system may not be able to use I/O channel memory unless the system board is fully populated.

```
On Return:
  (AX) - Contiguous 1KB blocks of memory available beginning
         at address 100000H
```

(AH) = 89H - Switch Processor to Protected Mode

For PC*jr* and PC:

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

For PC XT, PC Convertible, and PS/2 Model 25 and Model 30:

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

For AT, PC XT Model 286, and PS/2 products except Model 25 and Model 30, this function allows the user to switch into the protected (virtual address) mode. Upon completion, the system microprocessor is in the protected mode and control is transferred to the code segment specified by the user.

Entry requirements:

> (ES:SI) points to a GDT built before calling this function. 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 built by the routine using this function.

(BH) contains an index into the interrupt descriptor table that states where the first eight hardware interrupts begin (interrupt level 1). (BL) contains an index into the interrupt descriptor table that states where the second eight hardware interrupts begin (interrupt level 2).

The following shows the organization of the selectors in this GDT; the actual location is pointed to by (ES:SI).

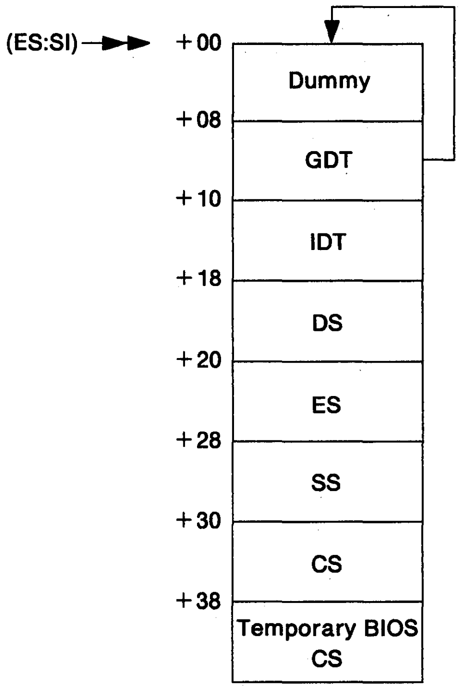


Figure 2-11. Global Descriptor Table

Each descriptor must contain the limit, the base address, and the access rights byte. The descriptors are defined as follows:

- The first is the required dummy and is user initialized to 0.
- The second points to the GDT as a data segment and is user initialized.
- The third points to the user-defined interrupt descriptor table (IDT) and is user initialized.

- The fourth points to the user data segment (DS) and is user initialized.
- The fifth points to the user extra segment (ES) and is user initialized.
- The sixth points to the user stack segment (SS) and is user initialized.
- The seventh points to the user code segment (CS) that this function returns to, and is user initialized.
- The eighth is used to establish a code segment for itself. This is necessary for this function to complete its operation while 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 using this
          function.

On Return:
  (AH) = 00H - Operation successfully completed

All segment registers are changed, (AX) and (BP) are modified.
```

Considerations:

1. BIOS functions are not available to the user. The user must handle all I/O commands.
2. Interrupt vector locations must be moved, due to the 80286 reserved areas.
3. The hardware interrupt controllers must be reinitialized to define locations that do not reside in the 80286 reserved areas.
4. An exception interrupt table and handler must be initialized by the user.
5. The interrupt descriptor table cannot overlap the real mode BIOS interrupt descriptor table.

The following is an example of a way to call the protected (virtual address) mode:

```
- User code -
MOV   AX,GDT SEGMENT
MOV   ES,AX
MOV   SI,GDT OFFSET
MOV   BH,HARDWARE INT LEVEL 1 OFFSET
MOV   BL,HARDWARE INT LEVEL 2 OFFSET
MOV   AH,89H
INT   15H
- User code -
(Protected mode established)
```

(AH) = 8AH to 8FH - Reserved

(AH) = 90H - Device Busy

For PC*jr* and PC:

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

For PC XT BIOS dated 11/08/82:

```
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 91H use the following general guidelines:

00H to 7FH:	Serially reusable devices (operating system must serialize access).
80H to BFH:	Reentrant devices; (ES:BX) is used to distinguish different calls (multiple I/O calls are allowed simultaneously).
C0H to FFH:	Wait only calls; there is no complementary Post 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 PC*jr* and PC:

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

For PC XT BIOS dated 11/08/82:

```
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 (AH) = 90H (Device Busy)]
```

(AH) = 92H to BFH - Reserved

(AH) = C0H - Return System Configuration Parameters

For PC*jr* and PC:

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

For PC XT BIOS dated 11/08/82 and AT BIOS dated 1/10/84:

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

For AT BIOS dated 6/10/85 and after, PC XT BIOS dated 1/10/86 and after, PC XT Model 286, PC Convertible, and Personal System/2 products:

```
(AH) = C0H

On Return:
  (ES:BX) - Pointer to system descriptor vector in ROM
  (AH) = 0
  CF = 0

  System Descriptor:

    DW      XXXX      Byte count of data that follows;
                      minimum length = 8

    DB      XX        Model byte
                      See "System Identification" on page 4-18

    DB      XX        Submodel byte
                      See "System Identification" on page 4-18

    DB      XX        BIOS revision level
                      See "System Identification" on page 4-18
                      00 = First release
                      Revision level is increased by one
                      for each subsequent release of code

    DB      XX        Feature information byte 1

                      Bit 7 = 1 - Fixed disk BIOS uses
                                  DMA channel 3
                            = 0 - DMA channel 3 not used by
                                  fixed disk BIOS or channel 3
                                  usage cannot be determined

                      Bit 6 = 1 - Slave interrupt controller
                                  present
                            = 0 - Slave interrupt controller not
                                  present

                      Bit 5 = 1 - Real-time clock present
                            = 0 - Real-time clock not present
```

```
                        Bit 4 = 1 - Keyboard intercept sequence
                                    (INT 15H) called in keyboard
                                    interrupt (INT 09H)
                              = 0 - Keyboard intercept sequence not
                                    called

                        Bit 3 = 1 - Wait for external event
                                    supported
                              = 0 - Wait for external event not
                                    supported

                        Bit 2 = 1 - Extended BIOS data area is
                                    allocated
                              = 0 - Extended BIOS data area is
                                    not allocated

                        Bit 1 = 1 - Micro Channel implemented
                              = 0 - PC type I/O channel implemented

                        Bit 0 - Reserved

    DB        XX        Feature information byte 2

                        Bit 7 - Reserved

                        Bit 6 = 1 - Keyboard functionality call
                                    supported
                              = 0 - Keyboard functionality call
                                    not supported

                        Bits 5 to 0 - Reserved
    DB        XX        Feature information byte 3 - Reserved

    DB        XX        Feature information byte 4 - Reserved

    DB        XX        Feature information byte 5 - 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, CF = 1, and (ES:BX) is not changed.

(AH) = C1H - Return Extended BIOS Data Area Segment Address

For PC*jr* and PC:

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

For PC XT, AT, PC XT Model 286, and PC Convertible:

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

For all others:

```
On Return:
  (ES) - Extended BIOS data area segment address
  CF = 0 - No error
     = 1 - Error
```

(AH) = C2H - Pointing Device BIOS Interface

For PC*jr* and PC:

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

For PC XT, AT, PC XT Model 286, and PC Convertible:

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

For all others:

```
(AL) = 00H - Enable/disable pointing device
     (BH) = 00H - Disable
          = 01H - Enable


On Return:
  CF = 0 - Operation successfully completed
  CF = 1 - Operation unsuccessful
  (AH) - Status
       = 00H - No error
       = 01H - Invalid function call
       = 02H - Invalid input
       = 03H - Interface error
       = 04H - Resend
       = 05H - No far call installed


(AL) = 01H - Reset pointing device
```

```
On Return:
  See Return for (AL) = 00H on page 2-95

If the operation successfully completed:
  (BH) - Device ID
       = 00H
  The pointing device state is as follows:
       -Disabled
       -Sample rate at 100 reports per second
       -Resolution at 4 counts per millimeter
       -Scaling at 1 to 1
       -Data package size remains the same as before this
        function was called

  (BL) is modified on return


(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:
  See Return for (AL) = 00H on page 2-95


(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:
  See Return for (AL) = 00H on page 2-95


(AL) = 04H - Read device type


On Return:
  See Return for (AL) = 00H on page 2-95

If the operation 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:
  See Return for (AL) = 00H on page 2-95

  The pointing device state is as follows:
       -Disabled
       -Sample rate at 100 reports per second
       -Resolution at 4 counts per millimeter
       -Scaling at 1 to 1
```

```
(AL) = 06H - Extended commands
     (BH) = 00H - Return status
```

```
On Return:
  See Return for (AL) = 00H on page 2-95
```

```
If the operation 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:
  See Return for (AL) = 00H on page 2-95


(BH) = 02H - Set scaling to 2:1


On Return:
  See Return for (AL) = 00H on page 2-95


(AL) = 07H - Device driver far call initialization
     (ES) - Segment
     (BX) - Offset


On Return:
  See Return for (AL) = 00H on page 2-95
```

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 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 1)
      Bit 2 - Reserved (must be 0)

      Bit 1 - Right button status
            = 1 - Pressed

      Bit 0 - Left button status
            = 1 - Pressed
   High byte = 0


 Word 2 on stack:
   Low byte - X data
      Bit 7 = Most significant bit
      Bit 0 - Least significant bit
   High byte = 0


 Word 3 on stack:
   Low byte - Y data
      Bit 7 = Most significant bit
      Bit 0 - Least significant bit
   High byte = 0


 Word 4 on stack:
   High byte = 0
   Low 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 PC*jr* and PC:

```
 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:

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

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

```
(AL) = 00H - Disable watchdog time-out
     = 01H - Enable watchdog time-out
     (BX) - Watchdog timer count
            (1 to 255 is valid for Personal System/2 products)

On Return:
  CF = 0 - Operation successfully completed
     = 1 - Operation unsuccessful
```

(AH) = C4H - Programmable Option Select (POS)

For PC*jr* and PC:

```
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:

```
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:
  (AL) = 00H
  (DX) - Base POS adapter register address


(AL) = 01H - Enable slot for setup
(BL) - Slot number

On Return:
  (AL) = 01H
  (BL) - Slot number
```

```
(AL) = 02H - Adapter enable

On Return:
  (AL) = 02H

On Return for all:
  CF = 0 - Operation successfully completed
     = 1 - Request failed
```

(AH) = C5H 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 0FH – Reserved
(AH) = 10H – Extended Keyboard Read
(AH) = 11H – Extended Keystroke Status
(AH) = 12H – Extended Shift Status
(AH) = 13H to FFH – Reserved
```

Figure 2-12. INT 16H - Keyboard Functions

The extended functions, (AH) = 10H, 11H, and 12H, have been added to the BIOS interface to support the 101/102-Key Keyboard. The extended-function keyboard scan codes fall into one of three categories:

1. When only one key produces an ASCII character, the scan code read from the keyboard port is the same as with the standard keyboards.
2. 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.
3. New scan codes are assigned to keys that did not exist on the standard keyboards.

The extended functions allow new programs to take advantage of all categories and avoid compatibility problems with existing programs.

If the extended functions are not supported by the system BIOS, the scan code/character code combination placed in the keyboard buffer by the keyboard interrupt handler are returned without change upon calling (AH) = 00H (Keyboard Read) and (AH) = 01 (Keystroke Status).

If the extended functions are supported by the system BIOS:

- The character code placed in the keyboard buffer by the keyboard interrupt handler differentiates between keys with identical nomenclature.
- The keyboard interrupt handler places the scan code/character code combination for new keys in the keyboard buffer.
- (AH) = 10H (Extended Keyboard Read) and (AH) = 11H (Extended Keystroke Status) 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. The scan code/character code combination is returned for like keys, with the character code used to differentiate between them. If the character code is equal to hex F0H and the scan code is not equal to hex 00H, the character code is set to hex 00H.
- (AH) = 00H (Keyboard Read) and (AH) = 01H (Keystroke Status) extract the scan code/character code combination and translate it, if necessary, to the the scan code/character code combination compatible with previous keyboards. The translation:
  1. Converts like codes to compatible codes
  2. Extracts the scan code/character code combination until a compatible combination is found.
- (AH) = 12H (Extended Shift Status) returns the existing keyboard shift state and the shift state of the separate Ctrl and Alt keys.

To determine if the extended functions, (AH) = 10H, 11H, and 12H, are supported by the system BIOS, the program must use INT 16H, (AH) = 05H (Keyboard Write) to write a scan code/character code combination of hex FFFF to the buffer. If on return (AL) = 00H, the function successfully inserted hex FFFF into the buffer. Next, INT 16H, (AH) = 10H (Extended Keyboard Read) is issued to read the scan code/character code combination from the keyboard buffer. If on return (AX) is hex FFFF, the extended keyboard functions are supported. If on return (AX) is not hex FFFF, INT 16H, (AH) = 10H (Extended Keyboard Read) is issued until (AX) is hex FFFF on return. If after 16 tries (the buffer size) or if each of the calls to the Extended Keyboard Read function yields an (AX) not equal to hex FFFF, the extended keyboard functions are not supported.

See “Scan Code/Character Code Combinations” on page 4-24 for scan code/character code combinations.

(AH) = 00H - Keyboard Read

The scan code/character code is extracted from the buffer. The keyboard buffer head pointer (hex 40:1A) is increased by 2 or, if the pointer is already at the end, is reinitialized to the start of the buffer.

```
On Return:
  (AL) - ASCII character code
  (AH) - Scan code
```

For AT, PC XT Model 286, PC Convertible, and Personal System/2 products, if no keystroke is available, INT 15H, (AH) = 90H (Device Busy) is called with (AL) = 02H (Type = Keyboard), to inform the operating system that a keyboard loop is about to take place, allowing the operating system to perform another task. When the keyboard operation is completed, INT 09H calls INT 15H, (AH) = 91H (Interrupt Complete) with (AL) = 02H (Type = Keyboard). See “Multitasking Provisions” on page 4-16 for additional information.

Note: Control is returned only when a keystroke is available. The keystroke is removed from the buffer.

(AH) = 01H - Keystroke Status

```
On Return:
  ZF = 1 - No code available
     = 0 - Code is available

If code is available:
  (AL) - ASCII character code
  (AH) - Scan code
```

Note: The keystroke is not removed from the buffer.

(AH) = 02H - Shift Status

```
On Return:
  (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) - Reserved
```

(AH) = 03H - Set Typematic Rate

For PC*jr*, and for Personal System/2 BIOS supporting Interrupt 16H, (AH) = 09H with bit 0 = 1:

```
(AL) = 00H - Returns to default, restores original state
             (typematic on, normal initial delay and normal
             typematic rate)
```

For PC*jr* only:

```
(AL) = 01H - Increases initial delay (this is the delay between
             first character and the burst of typematic
             characters)

(AL) = 02H - Slows typematic characters by one-half

(AL) = 03H - Increases initial delay and slows typematic
             characters by one-half
```

For PC*jr*, and for Personal System/2 BIOS supporting Interrupt 16H, (AH) = 09H with bit 1 = 1:

```
(AL) = 04H - Turns off typematic characters
```

For AT BIOS dated 11/15/85 and after, PC XT Model 286, and Personal System/2 products:

```
(AL) = 05H - Set typematic rate and delay
     (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

     (BH) - Delay value (in milliseconds)
      00H = 250
      01H = 500
      02H = 750
      03H = 1000
      04H to FFH - Reserved
```

For Personal System/2 BIOS supporting Interrupt 16H, (AH) = 09H with bit 3 = 1:

```
(AL) = 06H - Return current typematic rate and delay

On Return:
  (BL) - Typematic rate
  (BL) - Delay
```

For all others no action is performed.

(AH) = 04H - Keyboard Click Adjustment

For PC*jr* 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 after, PC XT dated 1/10/86 and after, PC XT Model 286, and Personal System/2 products, this function places scan code/character code combination in the keyboard buffer as if they came from the keyboard.

```
(CL) - ASCII character code
(CH) - Scan 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 if INT 16H, function (AH) = 09H is supported, the program must use INT 15H, (AH) = C0H (Return System Configuration Parameters), testing bit 6 of Feature Information Byte 2. If this bit is 0, (AH) = 09H is undefined. If this bit is 1, (AH) = 09H is defined as follows. This test can be made for all systems that support INT 15H, (AH) = C0H.

```
On Return:
  (AL) - Function code
       Bits 7 to 4 - Reserved
       Bit  3 = 1 - Get current typematic rate/delay supported
                0 - Get current typematic rate/delay
                    not supported
       Bit  2 = 1 - Set typematic rate/delay supported
                0 - Set typematic rate/delay not supported
       Bit  1 = 1 - Turn on/off typematic supported
                0 - Turn on/off typematic not supported
       Bit  0 = 1 - Return to default typematic rate/delay
                    supported
                0 - Return to default typematic rate/delay
                    not supported
```

(AH) = 0AH to 0FH - Reserved

(AH) = 10H - Extended Keyboard Read

For AT BIOS dated 11/15/85 and after, PC XT dated 1/10/86 and after, 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 (hex 40:1A) is increased by 2. If the pointer is already at the end, it is reinitialized to the start of the buffer .

```
On Return:
  (AL) - ASCII character code
  (AH) - Scan 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 after, PC XT dated 1/10/86 and after, PC XT Model 286, and Personal System/2 products:

```
On Return:
  ZF = 1 - No code available
     = 0 - Code is available

If 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 after, PC XT dated 1/10/86 and after, PC XT Model 286, and Personal System/2 products:

```
On Return:
  (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) - 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
```

For all others no action is performed.

(AH) = 13H 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 2-13. INT 17H - Printer Functions

(AH) = 00H - Print Character

```
(AL) - Character to print
(DX) - Printer to use (0,1,2); index into the port base address
       table at hex 40:08


On Return:
  (AH) - Status
       Bit 7 = 1 - Not busy
       Bit 6 = 1 - Acknowledge
       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 use (0,1,2); index into the port base address
       table at hex 40:08


On Return:
  (AH) - Status
       Bit 7 = 1 - Not busy
       Bit 6 = 1 - Acknowledge
       Bit 5 = 1 - Out of paper
       Bit 4 = 1 - Selected
       Bit 3 = 1 - I/O error
       Bits 2, 1 - Reserved
       Bit 0 - Time-out
```

(AH) = 02H - Read Status

```
(DX) - Printer to use (0,1,2); index into the port base address
       table at hex 40:08


On Return:
  (AH) - Status
       Bit 7 = 1 - Not busy
       Bit 6 = 1 - Acknowledge
       Bit 5 = 1 - Out of paper
       Bit 4 = 1 - Selected
       Bit 3 = 1 - I/O error
       Bits 2, 1 - Reserved
       Bit 0 - Time-out
```

(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, the BIOS calls INT 15H, (AH) = 90H (Device Busy) with (AL) = FEH (Type = Printer), informing the operating system that a time-out loop is about to begin. See “Multitasking Provisions” on page 4-16 for additional information.

2. For AT BIOS dated before 11/15/85, PC*jr*, PC, and PC XT BIOS dated 11/08/82, the printer port number associated with (DX) is tested for 0. If found to be 0, no action occurs. If it is non 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 hex 40:08.

3. For PC XT BIOS dated 1/10/86 and after, if (DX) is greater than 3 or the printer port associated with (DX) is 0, no action is performed and, on return (AH) = 29H.

4. For PC Convertible, if the printer port associated with (DX) is 0, the return is (AH) = 01H. If (DX) is non 0, the print operation is performed. No test is made on (DX) to see if a valid printer port number exists at the offset into the printer base address data area at hex 40:08.

5. For AT BIOS dated 11/15/85 and PC XT Model 286, if (DX) is greater than 3 or the printer port associated with (DX) is 0, no action is performed and (AH) is returned unchanged.
6. For Personal System/2 products, if (DX) is greater than 2 or the printer port associated with (DX) is 0, no action is performed and (AH) is returned unchanged.

Interrupt 19H - Bootstrap Loader

Track 0, sector 1 is read into segment 0, offset 7C00H. Control is then transferred as follows:

```
(CS) = 0000H
(IP) = 7C00H
(DL) - Drive where bootstrap sector was read
```

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 2-14. INT 1AH - System-Timer and Real-Time Clock Services

Note: For Personal System/2 Model 25 a real-time clock is not available. Therefore, (AH) = 02H through (AH) = 09H do not apply. The system-timer functions do apply.

(AH) = 00H - Read System-Timer Time Counter

```
On Return:
  (CX) - High portion of count
  (DX) - Low portion of count
  (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.
```

Note: Execution causes the timer overflow flag (hex 40:70) to be reset to 0. Counts occur at the rate of 1193180÷65536 counts per second (about 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 1193180÷65536 counts per second (about 18.2 per second).

(AH) = 02H - Read Real-Time Clock Time

For AT BIOS dated before 6/10/85:

```
On Return:
  (CH) - Hours in 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 after, 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 savings time option
       = 00H - No daylight savings 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 in BCD (19 or 20)
  (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 in BCD (19 or 20)
(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 passed in (CH), (CL), and (DH) respectively. When the alarm interrupt occurs, a software interrupt 4AH is issued. The user must point software interrupt 4AH to an alarm routine prior to setting the real-time clock alarm INT 1AH, (AH) = 06H. Only one alarm function may 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 after, PC XT Model 286, and Personal System/2 products:

```
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 system
       02H = Alarm enabled and will power-on system
```

Note: Personal System/2 Model 30 does not support the power-on system feature.

For AT BIOS dated 6/10/85 and after, PC XT Model 286, and Personal System/2 products except Model 30:

```
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:

```
On Return:
  CF = 1 - Invalid function request
```

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

```
On Return:
  (CX) - Count of days since 1-1-1980
```

Note: The count of days since 1/1/80 is initialized to 0 during the POST.

For all others no action is performed.

(AH) = 0BH - Set System-Timer Day Counter

For AT and PC XT Model 286:

```
On Return:
  CF = 1 - Invalid function request
```

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

```
(CX) - Count of days since 1-1-1980
```

Note: The count of days since 1/1/80 is initialized to 0 during the POST.

For all others no action is performed.

(AH) = 0CH to 7FH - Reserved

(AH) = 80H - Set Up Sound Multiplexer

For PC*jr*:

```
(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 after, PC XT Model 286, PC Convertible, and Personal System/2 products:

```
On Return:
  CF = 1 - Invalid function request
```

For all others no action is performed.

(AH) = 81H to FFH - Reserved

Interrupt 70H - Real-Time Clock Interrupt

For AT, PC XT Model 286, and Personal System/2 products except Model 25:

This interrupt handler controls the periodic and alarm interrupt functions from the real-time clock.

Periodic function — When activated, the interrupt occurs approximately 1024 times per second. The doubleword microsecond counter is decremented by a value of 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. For INT 15H, (AH) = 83H (Event Wait), the designated location is provided by the user. For INT 15H, (AH) = 86H (Wait), 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 prior to setting INT 1AH, (AH) = 06H (Real-Time Clock Alarm).

For all others, the Real-Time Clock Interrupt is not supported.

Notes:

1. The PC Convertible provides the above functions, but the Real-Time Clock Interrupt generates a nonmaskable interrupt rather than INT 70H. Additionally, PC Convertible uses the real-time clock update ended interrupt function (interrupts once per second) when certain system profiles are enabled.
2. For Personal System/2 Model 30, the periodic function is not supported.

Notes:

Section 3. Data Areas and ROM Tables

Notes:

BIOS Data Area

The BIOS Data Area is allocated specifically as a work area 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 area follows:

Address (Hex)	**Function**	**Size**
40:00	RS-232C Communications Line 1 Port Base Address	Word
40:02	RS-232C Communications Line 2 Port Base Address	Word
40:04	RS-232C Communications Line 3 Port Base Address	Word
40:06	RS-232C Communications Line 4 Port Base Address	Word

Note: The RS-232C communications line port base address fields may be initialized to 0 by the POST if the system configuration contains less than four serial ports. The POST never places 0 in the RS-232C communications line port base address table between two valid RS-232C communications 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 may be initialized to 0 by the POST if the system configuration contains less than four parallel ports. The POST never places 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 11,10,9	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 (PC Convertible Only)	
Bit 2	Reserved (PC, PC XT, AT, and PC Convertible)	

Note: Refer to INT 11H for equipment return information .

Figure 3-3. System Equipment Data Area

Address (Hex)	**Function**	**Size**
40:12	Reserved	Byte
Exception		
40:12	Power-On Self-Test Status (PC Convertible Only)	Byte

Figure 3-4. Miscellaneous Data Area 1

Address (Hex)	Function	Size
40:13	Memory Size in KB (Range 0 to 640)	Word
40:15 to 40:16	Reserved	Byte

Figure 3-5. Memory Size Data Area

Address (Hex)	Function	Size
40:17	Keyboard Control	Byte
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	
40:18	Keyboard Control	Byte
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	System Request Key Pressed	
Bit 1	Left Alt Key Pressed	
Bit 0	Left Ctrl Key Pressed	
40:19	Alternate Keypad Entry	Byte
40:1A	Keyboard Buffer Head Pointer	Word
40:1C	Keyboard Buffer Tail Pointer	Word
40:1E	Keyboard Buffer	32 Bytes

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	
	40H = Seek Operation Failed	
	80H = Diskette Drive not Ready	
40:42	Diskette Drive Controller Status Bytes	7 Bytes

Figure 3-7. Diskette Drive Data Area

Address (Hex)	Function	Size
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	CRT 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

Address (Hex)	Function	Size
40:67	Reserved	DWord
40:6B	Reserved	Byte
Exception		
40:67	Pointer to reset code upon system reset with memory preserved (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

Address (Hex)	Function	Size
40:6C	Timer Counter	DWord
40:70	Timer Overflow (If non 0, timer has counted past 24 hours.)	Byte

Figure 3-10. System-Timer Data Area

Address (Hex)	Function	Size
40:71	Break Key State	Byte
40:72	Reset Flag 1234H = Bypass Memory Test 4321H = Preserve Memory (Personal System/2 products except Model 25 and Model 30) 5678H = System Suspended (PC Convertible) 9ABCH = Manufacturing Test Mode (PC Convertible) ABCDH = System POST Loop Mode (PC Convertible)	Word

Figure 3-11. System Data Area 2

Address (Hex)	Function	Size
40:74	Last Fixed Disk Drive Operation Status 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	Byte

Figure 3-12 (Part 1 of 2). Fixed Disk Drive Data Area

Address (Hex)	Function	Size
40:75	Number of Fixed Disk Drives Attached	Byte
40:76	Reserved	Byte
40:77	Reserved	Byte
Exceptions		
40:74	Reserved (devices using ESDI-type commands)	Byte
40:76	Fixed Disk Drive Control (PC XT)	Byte
40:77	Fixed Disk Drive Controller Port (PC XT)	Byte

Figure 3-12 (Part 2 of 2). 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	Reserved	
Exception		
40:7B	Printer 4 Time-out Value (PC, PC XT, and AT)	Byte

Figure 3-13. Printer Time-Out Value Data Area

Address (Hex)	Function	Size
40:7C	RS-232C Communications Line 1 Time-out Value	Byte
40:7D	RS-232C Communications Line 2 Time-out Value	Byte
40:7E	RS-232C Communications Line 3 Time-out Value	Byte
40:7F	RS-232C Communications Line 4 Time-out Value	Byte

Figure 3-14. RS-232C Time-Out Value Data Area

Address (Hex)	Function	Size
40:80	Keyboard Buffer Start Offset Pointer	Word
40:82	Keyboard Buffer End Offset Pointer	Word

Figure 3-15. Keyboard Data Area 2

Address (Hex)	Function	Size
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 = Reserved	
Bits 5,4	Last Diskette Drive Step Rate Selected	
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 2). Diskette Drive/Fixed Disk Drive Control Data Area

Address (Hex)	Function	Size
40:90	Drive 0 Media State	Byte
40:91	Drive 1 Media State	Byte
Bits 7,6	Diskette Drive Data Rate (Values in Binary)	
	00 = 500KB Per Second	
	01 = 300KB Per Second	
	10 = 250KB Per Second	
	11 = Reserved	
Bit 5	Double Stepping Required	
Bit 4	Media Established	
Bit 3	Reserved	
Bits 2,1,0	Drive/Media State (Values in Binary)	
	000 = 360KB Diskette/360KB Drive not Established	
	001 = 360KB Diskette/1.2MB Drive not Established	
	010 = 1.2MB Diskette/1.2MB Drive not Established	
	011 = 360KB Diskette/360KB Drive Established	
	100 = 360KB Diskette/1.2MB Drive Established	
	101 = 1.2MB Diskette/1.2MB Drive Established	
	110 = Reserved	
	111 = None of the Above	
40:92	Reserved	Byte
40:93	Reserved	Byte
40:94	Drive 0 Current Cylinder	Byte
40:95	Drive 1 Current Cylinder	Byte
Exception		
40:8B to 40:95	Reserved (PC, PC*jr*, PC XT BIOS Dated 11/8/82, and PC Convertible)	Byte

Figure 3-17 (Part 2 of 2). Diskette Drive/Fixed Disk Drive Control Data Area

Address (Hex)	Function	Size
40:96	Keyboard Mode State and Type Flags	Byte
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	
40:97	Keyboard LED Flags	Byte
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 0)	
Bits 2,1,0	Keyboard LED State Bits	

Figure 3-18. Keyboard Data Area 3

Address (Hex)	Function	Size
40:98	Offset Address to User Wait Complete Flag	Word
40:9A	Segment Address to User Wait Complete Flag	Word
40:9C	User Wait Count - Low Word (Microseconds)	Word
40:9E	User Wait Count - High Word (Microseconds)	Word
40:A0	Wait Active Flag	Byte
Bit 7	Wait Time Elapsed and Post	
Bits 6 to 1	Reserved	
Bit 0	INT 15H, AH = 86H (Wait) has Occurred	
40:A1 to 40:A7	Reserved	Byte

Figure 3-19. Real-Time Clock Data Area

For systems with EGA capability and Personal System/2 products, the save pointer table contains pointers that define specific dynamic overrides for the video mode set function, INT 10H, (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 (except Personal System/2 Model 25 and Model 30)	
	Initialized to hex 00:00, this value is optional. When non 0, 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	Alpha Mode Auxiliary Character Generator Pointer	
	Initialized to hex 00:00, this value is optional. When non 0, this value points to a table that is described as follows:	
	Bytes/Character	Byte
	Block to Load, 0 = Normal Operation	Byte
	Count to Store, 256 = Normal Operation	Word
	Character Offset, 0 = Normal Operation	Word

Figure 3-20 (Part 1 of 2). Save Pointer Data Area

Address (Hex)	Function	Size
	Pointer to a Font Table	DWord
	Displayable Rows If 0FFH, the maximum calculated value is used, otherwise this value is used.	Byte
	Consecutive bytes of mode values for this font description. The end of this stream is indicated by a byte code of 0FFH.	Byte
Note: Use of the DWord 3 pointer may cause unexpected cursor type operation. For an explanation of cursor type, see INT 10H, (AH) = 01H.		
DWord 4	Graphics Mode Auxiliary Character Generator Pointer Initialized to hex 00:00, this value is optional. When non 0, this value points to a table that is described as follows:	
	Displayable Rows	Byte
	Bytes Per Character	Word
	Pointer to a Font Table	DWord
	Consecutive bytes of mode values for this font description. The end of this stream is indicated by a byte code of 0FFH.	Byte
DWord 5	Secondary Save Pointer (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	Function	Size
Word 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 MDPA	
	00,02 Entry 2 CGA	
	02,01 Entry 3 MDPA + CGA	
	00,04 Entry 4 EGA	
	04,01 Entry 5 EGA + MDPA	
	00,05 Entry 6 MEGA	
	02,05 Entry 7 MEGA + CGA	
	00,06 Entry 8 PGC	
	01,06 Entry 9 PGC + MDPA	
	05,06 Entry 10 PGC + MEGA	
	00,08 Entry 11 CVGA	
	01,08 Entry 12 CVGA + MDPA	
	00,07 Entry 13 MVGA	
	02,07 Entry 14 MVGA + CGA	
	02,06 Entry 15 MVGA + PGC	

	Abbreviation Meanings:
	MDPA = Monochrome Display and Printer Adapter
	CGA = Color/Graphics Monitor Adapter
	EGA = Enhanced Graphics Adapter
	MEGA = EGA with monochrome display
	PGC = Professional Graphics Controller
	VGA = Video Graphics Array
	MVGA = VGA based with monochrome display
	CVGA = VGA based with color display	

Figure 3-21 (Part 1 of 3). Secondary Save Pointer Data Area

Address	Function	Size
DWord 3	Second Alpha Mode Auxiliary Character Generator Pointer	
	Initialized to hex 00:00, this value is optional. When non 0, this value points to a table that is described as follows:	
	Bytes/Character	Byte
	Block to load, should be non 0 for normal operation.	Byte
	Reserved	Byte
	Pointer to a Font Table	DWord
	Consecutive bytes of mode values for this font description. The end of this stream is indicated by a byte code of 0FFH.	Byte
Note: Attribute bit 3 is used to switch between primary and secondary fonts. It may be desirable to use the user palette profile to define a palette of consistent colors independent of attribute bit 3.		
DWord 4	User Palette Profile Table Pointer	
	Initialized to hex 00:00, this value is optional. When non 0, this value points to a table that is described as follows:	
	Underlining flag (1 = On, 0 = Ignore, -1 = Off; 0 = Normal Operation)	Byte
	Reserved	Byte
	Reserved	Word
	Internal Palette Count (0-17; 17 = Normal Operation)	Word
	Internal Palette Index (0-16; 0 = Normal Operation)	Word
	Pointer to Internal Palette	DWord

Figure 3-21 (Part 2 of 3). Secondary Save Pointer Data Area

Address	Function	Size
	External Palette Count (0-256; 256 = Normal Operation)	Word
	External Palette Index (0-255; 0 = Normal Operation)	Word
	Pointer to External Palette	DWord
	Consecutive bytes of mode values for this font description. The end of this stream is indicated by a byte code of 0FFH.	Byte
DWord 5 to DWord 7	Reserved and set to hex 00:00.	

Figure 3-21 (Part 3 of 3). Secondary Save Pointer Data Area

Address (Hex)	Function	Size
40:AC to 40:FF	Reserved	Byte
50:00	Print Screen Status Byte (INT 05H Status)	Word

Figure 3-22. Miscellaneous Data Area 2

Extended BIOS Data Area

The Extended BIOS Data Area is supported on Personal System/2 products only. The POST allocates the highest possible (n) KB of memory below 640KB to be used as this data area. The word value at hex 40:13 (memory size), indicating the number of KB below the 640KB limit, is decremented by (n). The first byte in the Extended BIOS Data Area is initialized to the length in KB of the allocated area.

To access the Extended BIOS Data Area segment, issue an INT 15H, (AH) = C1H (Return Extended BIOS Data Area Segment Address). To determine if an Extended BIOS Data Area is allocated, use INT 15H, (AH) = C0H (Return System Configuration Parameters).

ROM Tables

The following ROM tables are used by the BIOS to define the characteristics of the hardware devices 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	1 Word	Maximum Number of Cylinders
2	1 Byte	Maximum Number of Heads
3	1 Word	For PC XT: Starting Reduced Write Current Cylinder All Others: Not Used
5	1 Word	Starting Write Precompensation Cylinder
7	1 Byte	For PC XT: Maximum ECC Data Burst Length All Others: Not Used
8	1 Byte	Control Byte For PC XT: Bit 7 - Disable Disk-Access Retries Bit 6 - Disable ECC Retries Bits 5 to 3 = 0 Bits 2,1,0 - Drive Option 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 2,1,0 - Reserved
9	1 Byte	For PC XT: Standard Time-out Value All Others: Not Used
10	1 Byte	For PC XT: Time-out Value For Format Drive All Others: Not Used
11	1 Byte	For PC XT: Time-out Value For Check Drive All Others: Not Used
12	1 Word	For PC XT: Reserved All Others: Landing Zone
14	1 Byte	For PC XT: Reserved All others: Number of Sectors Per Track
15	1 Byte	Reserved

Figure 3-23. Fixed Disk Drive Parameter Table Definition

For AT and Personal System/2 products, the following 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 through 255 are reserved.

Figure 3-24. Fixed Disk Drive Parameters (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 contains entries 0 through 14.
3. AT BIOS dated 6/10/85 or 11/15/85 contains entries 0 through 23.
4. PC XT Model 286 contains entries 0 through 24.
5. Personal System/2 products except Model 25 and Model 30 contain entries 0 through 32.
6. Personal System/2 Model 30 contains entries 0 through 26.
7. For Personal System/2 Model 70 and Personal System/2 Model 80 BIOS dated 10/07/87 and after, the fixed disk drive parameters in Figure 3-24 on page 3-19 and Figure 3-25 on page 3-21 do not apply. Also, software interrupts 41H and 46H are reserved.

For Personal System/2 products except Model 25 and Model 30 the following fixed disk parameter table applies:

Offset	Length	Value	Description
0	2	41	Length of Fixed Disk Drive Table
2	22	(ID)	ASCII string 'IBM HARDFILE TYPE xxx', where xxx is the type number in ASCII.
24	1	yyy	Type Number (Values are Binary)
25	2	*	Maximum Number of Cylinders
27	1	*	Maximum Number of Heads
28	2	0	Reserved
30	2	*	Start Write Precompensation Cylinder
32	1	0	Reserved
33	1	*	Control Byte Bit 7 or 6 - Disable Retries Bit 5 - Defect Map Installed Bit 3 - More Than 8 Heads (AT Only)
34	3	0	Reserved
37	2	*	Landing Zone
39	1	*	Number of Sectors Per Track
40	1	0	Reserved

Figure 3-25. Fixed Disk Drive Parameter Table (Personal System/2 products except Model 25 and Model 30)

Note: This information is located at head 0, track 0, sector 2 and applies only to ST412 and ST506 type drives.

For PC XT BIOS dated 11/10/82 the following fixed disk drive parameter tables apply:

Size	Value	Description
DW	0306	Maximum Cylinders
DB	02	Maximum Heads
DW	0306	Start Reduced Write Current Cylinder
DW	0000	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 3-26. Fixed Disk Drive Parameter Table 00 (PC XT BIOS Dated 11/10/82)

Size	Value	Description
DW	0375	Maximum Cylinders
DB	08	Maximum Heads
DW	0375	Start Reduced Write Current Cylinder
DW	0000	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 3-27. Fixed Disk Drive Parameter Table 01 (PC XT BIOS Dated 11/10/82)

Size	Value	Description
DW	0306	Maximum Cylinders
DB	06	Maximum Heads
DW	0128	Start Reduced Write Current Cylinder
DW	0256	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 3-28. Fixed Disk Drive Parameter Table 02 (PC XT BIOS Dated 11/10/82)

Size	Value	Description
DW	0306	Maximum Cylinders
DB	04	Maximum Heads
DW	0306	Start Reduced Write Current Cylinder
DW	0000	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 3-29. Fixed Disk Drive Parameter Table 03 (PC XT BIOS Dated 11/10/82)

Note: INT 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 1/08/86 and after the following fixed disk drive parameter tables 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 3-30. Fixed Disk Drive Parameter Table 00 - Type 1 (PC XT BIOS Dated 1/08/86)

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 3-31. Fixed Disk Drive Parameter Table 01 - Type 16 (PC XT BIOS Dated 1/08/86)

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 3-32. Fixed Disk Drive Parameter Table 02 - Type 2 (PC XT BIOS Dated 1/08/86)

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 3-33. Fixed Disk Drive Parameter Table 03 - Type 13 (PC XT BIOS Dated 1/08/86)

Note: INT 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	1 Byte	First Specify Byte
1	1 Byte	Second Specify Byte
2	1 Byte	Number of Timer Ticks to Wait Prior to Turning Diskette Drive Motor Off
3	1 Byte	Number of Bytes Per Sector 00H = 128 Bytes Per Sector 01H = 256 Bytes Per Sector 02H = 512 Bytes Per Sector 03H = 1024 Bytes Per Sector
4	1 Byte	Sectors Per Track
5	1 Byte	Gap Length
6	1 Byte	Dtl (Data Length)
7	1 Byte	Gap Length for Format
8	1 Byte	Fill Byte for Format
9	1 Byte	Head Settle Time (Milliseconds)
10	1 Byte	Motor Startup Time (1/8 Second) For Example: 8 = 1 Second Wait

Figure 3-34. Diskette Drive Parameter Table

Note: The diskette drive parameter table is pointed to by INT 1EH.

Section 4. Additional Information

Notes:

Interrupt Sharing

This section defines an interrupt sharing protocol that allows multiple hardware adapters on the PC type I/O channel and Micro Channel™ to share a single interrupt request line.

Precautions

Take the following precautions before implementing interrupt sharing:

- This 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.
- This interrupt sharing protocol does not apply to the sharing of an interrupt level between an interrupt handler running in the real mode and an interrupt handler running in the protected mode.
- This 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. The disabling of the interrupts allows the next handler to receive control as if a hardware interrupt had caused it to receive control.
- Interrupts must be disabled before the non-specific End of Interrupt (EOI) is issued and not reenabled in the interrupt handler to ensure that the Return from Interrupt (IRET) is executed. The flags are restored and the interrupts reenabled before another interrupt is serviced, protecting the stack from excessive build-up.
- All interrupt handlers must have a routine that can be executed after power-on to disable their adapters' interrupts. Executing this routine and resetting the interrupt sharing hardware ensures that adapters are deactivated if the user resets the system.

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

- Interrupt handler implementations must store data in memory using Intel format; that is, word hex 424B is stored as 4BH,42H in memory.

Interrupt Request (IRQn) Reset

The Micro Channel interrupt mechanism is level sensitive as opposed to the edge sensitive mechanism of the PC type I/O channel. The level sensitive Micro Channel mechanism simplifies the interrupt hardware needed for the adapters.

An interrupt request in the PC type I/O channel is implicitly reset due to the edge sensitive characteristic of the signal. In the Micro Channel, due to 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 if the bus slave hardware implicitly resets the interrupt request. An example of a bus slave device that implicitly resets an interrupt request is the system timer.

Interrupt-Sharing Software Requirements

All interrupt sharing software 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 stored in the interrupt-sharing chaining structure must be stored in Intel data format (see "Interrupt-Sharing Chaining Structure and Signature" on page 4-6). These requirements are specified to support the portability of the interrupt handlers across hardware operating environments.

The interrupt handling software for all adapters sharing an interrupt request line must implement this interrupt sharing software standard. Interrupt sharing software operating 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 once the task is complete.

To link an interrupt handler, the newly activated task's interrupt handler replaces the interrupt vector in low memory with a pointer to its own interrupt handler. (See "ROM Considerations" on page 4-7 for interrupt handlers stored in ROM.) The interrupt handler must preserve the interrupt vector it is replacing and use it as a forward pointer to the next interrupt handler in the chain. This 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 presenting the interrupt request. This is accomplished by the handler reading the contents of the interrupt status register of the adapter.

If the handler's device caused the interrupt, the handler must service the interrupt, reset the interrupt status bit, clear the interrupts, issue a non-specific EOI to the interrupt controller, then execute an IRET.

If the handler's device did not cause the interrupt, the handler passes control to the next interrupt handler in the chain using the previously stored forward pointer.

An interrupt handler is unlinked from a chain by the task first locating its handler's position within the chain. The chain can be searched by starting at the interrupt vector in low memory and using the offset of each handler's forward pointer to locate the entry point of each handler. This is done until the task finds its own handler. Each interrupt handler's signature (424BH) must be checked to ensure that a valid forward pointer exists. The task's forward pointer replaces the forward pointer of the previous handler in the chain, thus removing the handler from the chain.

Note: If the interrupt handler cannot locate its position in the chain, the interrupt handler cannot unlink.

An application-dependent unlinking error-recovery procedure must be incorporated into the unlinking routine for those situations where the unlinking routine discovers that the interrupt chain has been corrupted (an interrupt handler is linked but does not have a valid signature). All interrupt sharing handlers, except those in ROM (see "ROM Considerations" on page 4-7), must use 424BH as the signature to avoid corrupting the chain.

During a system-reset condition, a routine for each interrupt handler must be executed after power-on to disable interrupts from their responsible devices.

Operating system environments that support dynamic relocation of software must manage the entire interrupt sharing process. Interrupt handler software written exclusively for dynamic-relocation operating-system environments does not have to provide the interrupt-sharing chaining structure. These interrupt handlers do not have to provide linking and unlinking support. They must provide support for disabling the interrupting capability of the bus slave they support.

Interrupt-Sharing Chaining Structure and Signature

The interrupt-sharing software chaining structure is in a 16-byte format containing a 4-byte forward pointer (FPTR), a 2-byte signature, and 8 reserved bytes (RES_BYTES), as depicted 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 interrupt handler's entry point. The first instruction of each handler is a short jump around the structure, placing the structure at a known offset from the beginning of the handler routine. Since the position of each interrupt handler's chaining structure is known (except for the handlers on adapter ROM), the FPTRs can be updated when linking and unlinking.

The FIRST flag is used to determine the handler's position in the chain when linking and unlinking for shared interrupt levels. The contents of the FLAGS byte is changed to the value of the FIRST flag (80H) to indicate that the handler is the first handler linked in the chain. All interrupt handlers not stored in ROM must store the FIRST flag (80H) in the FLAGS byte when they are the first 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

Adapters with interrupt handlers in ROM must implement chaining by storing the FPTR in latches or ports on the adapter. If the adapter is sharing interrupt levels 7 or 15, it must also store the FIRST flag that indicates whether it is positioned first in the chain of interrupt handlers. Storage of this information is required because it cannot be guaranteed that handlers in ROM will always link first and never unlink. The ROM handler must contain the signature 0000H beginning at the seventh byte from the handler entry point since the forward pointer in ROM handlers is not stored at the third byte from the handler entry point.

Implementation Information

The Interrupt Mask register is located at I/O port 21H. Specific End of Interrupt (EOI) values for the various interrupt levels are listed (67H for level 7). The specific EOI is accomplished by issuing an OUT to the programmable interrupt controller operation command register (port 20H), using Operation Command Byte 2. A non-specific EOI is accomplished by issuing an OUT value of hex 20 to the programmable interrupt controller operation command register (port 20H).

The following are examples of code used to implement 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 the SEG:OFF points to a valid interrupt handler or to an IRET (0CFH) for levels 7 and 15.
2. ROM interrupt handlers during ROMSCAN (before the operating system is loaded) and handlers on other than IRQ 7, must test the SEG:OFF as shown in the "else" clause in this listing to determine if they are the first handler in the chain. Checking the SEG:OFF to see if 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
                                        ; interrupt control/status register
REARM      EQU      2F7H                ;Global Rearm location for
                                        ; interrupt level 7
SPC_EOI    EQU      67H                 ;Specific EOI for programmable
                                        ; interrupt controller interrupt level 7
EOI        EQU      20H                 ;Non-specific 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 non-specific 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],4B42H    ;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:FTPR[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

The BIOS provides a method for integrating adapters with on-board ROM code into the system. During the POST, interrupt vectors are established for BIOS calls. After 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 the POST the absolute addresses hex C0000 through hex C7FFF are scanned in 2KB blocks in search of adapter ROM modules that need to be initialized (for example, valid video adapter ROM).

Later in the POST, the absolute addresses hex C8000 through hex DFFFF are scanned in 2KB blocks in search of 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 (limit hex 7F) in the ROM (length/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 the 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 the POST by executing a far return.

For PC Convertible, if the adapter ROM diagnoses a self-test error, the following should be done before returning:

- Set bit 4 of hex 40:12 (POST status) to 1
- Set the device number for the supported adapter into (AH)
- Set a two-digit error code into (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 except Model 25 and Model 30, video adapters in the channel have a ROM signature code that identifies the video adapter. During the 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 0CH in the ROM address space and consists of:

```
77H, CCH, 'VIDEO '
```

The POS parameters are accessed from offset 30H 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 C0000H to C7FFFH.

For Personal System/2 BIOS dated 10/07/87 and after, the adapter ROM integration method is as follows:

> Early in the POST, the absolute addresses hex C0000 through DFFFF are scanned in 2KB blocks in search of adapter ROM modules that have video adapter signature code, as previously described. Only adapters with a video signature code are initialized early in the POST.
>
> Later in the POST, the absolute addresses hex C0000 through DFFFF are again scanned in 2KB blocks to initialize other adapter ROM modules not initialized in the early scan.

For PC Convertible, during early ROM scan the following protocol is established to determine the video support:

> Upon 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 to application software.

Video Presence Test

Use this video presence test to determine which IBM video functions are present.

1. Issue an INT 10H, with (AH) = 1AH and (AL) = 00H (Read Display Combination Code).

   If on return (AL) is not equal to 1AH, the Read/Write Display Combination Code function is not supported, and step 2 should be followed to determine video presence.

   If on return (AL) = 1AH, the information returned in (BX) defines the video environment. The active display code is returned in (BL). The alternate display code, if any, is returned in (BH). Refer to INT 10H, (AH) = 1AH on page 2-39 for display code definitions.

2. To determine the presence of an IBM Enhanced Graphics Adapter (EGA) when the Display Combination Code function is not supported, issue an INT 10H with (AH) = 12H and (BL) = 10H (Return EGA Information).

   If on return, (BL) = 10H, an EGA is not present and step 3 should be followed.

   If on return (BL) is not equal to 10H then an EGA is present. Note that an IBM Color/Graphics Monitor Adapter or an IBM Monochrome Display and Printer Adapter may also be present, depending on the EGA switch settings.

3. Complete steps 1 and 2 before performing this step. The video functions that may 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 0B8000H, 0B0000H 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 function 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 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 should change the system equipment video mode type bits (see data area hex 40:10, bits 5, 4 on page 3-4) to monochrome or color and issue a INT 10H, (AH) = 00H (Set Mode).

3. A single video function that supports both monochrome and color video modes. To determine if a single video function supports both monochrome and color video modes, the application program should issue an INT 10H, (AH) = 1BH (Return Functionality/State Information).

   If on return (AL) is not equal to 1BH, the Return Functionality/State function is not supported. Support for both monochrome and color video modes on a single video function is not available.

   If on return (AL) = 1BH, use the returned information to determine if the All Modes on All Displays function is active. If active, color and monochrome modes are available and the application program should change the system equipment video mode type bits to monochrome or color and issue a INT 10H, (AH) = 00H (Set Mode). If inactive, only color modes or monochrome modes are available, depending on the results of the video presence test.

Multitasking Provisions

The BIOS provides hooks to assist in multitasking implementation. Whenever a busy (Wait) loop occurs in the BIOS, a hook is provided for the program to break out of the loop. Also, when the BIOS services an interrupt, a corresponding Wait loop is exited, and another hook is provided. A program may be written that employs the bulk of the device driver code. The following is valid only in the microprocessor real address mode, and the following steps must be taken by the code to allow 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 following four interfaces are used by the multitasking dispatcher:

Startup: The startup code hooks INT 15H. The dispatcher is responsible to check for function codes of (AH) = 90H or (AH) = 91H (see the following descriptions of Wait and Post). The dispatcher must pass all other functions to the previous user of INT 15H (use a JMP or a CALL). If (AH) = 90H or (AH) = 91H, the dispatcher must do 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): Whenever the BIOS is about to enter a Wait loop, it first issues an INT 15H, (AH) = 90H. This signals a wait condition. At this point, the dispatcher should save the task status and dispatch another task. This allows overlapped execution of tasks when the hardware is busy. The following is an outline of the code that has been added to the 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): Whenever the BIOS has set an interrupt flag for a corresponding busy loop, an INT 15H, (AH) = 91H occurs. 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 added to the BIOS that performs 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 7F) is serially reusable. This means that for the devices that use these codes, access to the BIOS must be restricted to one task at a time and the operating system must serialize access.
- The second (hex 80 to BF) is for reentrant devices. There is no restriction on the number of tasks that may access the device. ES:BX is used to distinguish different calls.
- The third (hex C0 to FF) is non interrupt (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 Waits. 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, it generally should remain there for a specific amount of time before indicating an error. The dispatcher must return to the BIOS Wait loop with the carry bit set if a time-out occurs.

System Identification

Each BIOS ROM module has a model byte located at hex F000:FFFE in ROM. In some cases, a submodel byte and a BIOS revision level byte are used to further distinguish the various BIOS ROM modules. To gain access to this information, see INT 15H, (AH) = C0H (Return System Configuration Parameters) on page 2-92.

Product	BIOS Date	Model Byte	Submodel Byte	Revision
PC	04/24/81	FF	—	—
PC	10/19/81	FF	—	—
PC	10/27/82	FF	—	—
PC XT	11/08/82	FE	—	—
PC XT	01/10/86	FB	00	01
PC XT	05/09/86	FB	00	02
PC*jr*	06/01/83	FD	—	—
AT	01/10/84	FC	—	—
AT	06/10/85	FC	00	01
AT	11/15/85	FC	01	00
PC XT Model 286	04/21/86	FC	02	00
PC Convertible	09/13/85	F9	00	00
PS/2 Model 25	06/26/87	FA	01	00
PS/2 Model 30	09/02/86	FA	00	00
PS/2 Model 50 (Type 1)	02/13/87	FC	04	00
PS/2 Model 50 (Type 2)	01/28/88	FC	04	03
PS/2 Model 60	02/13/87	FC	05	00
PS/2 Model 70 (Type 1)	01/29/88	F8	09	00
PS/2 Model 70 (Type 2)	01/29/88	F8	04	00
PS/2 Model 80 (Type 1)	03/30/87	F8	00	00
PS/2 Model 80 (Type 2)	10/07/87	F8	01	00

The "Supplements" section of this manual may contain additional system identification information.

Figure 4-1. System Identification

Note: Specific information about system board types can be found in the technical reference manual for that model.

Application Guidelines

Use the following information to develop application programs for the IBM Personal System/2 and Personal Computer products. Whenever possible, BIOS should be used as an interface to hardware in order to provide maximum compatibility and portability of applications across systems.

Math Coprocessor Testing

The BIOS Equipment Function 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 using the Micro Channel architecture while systems using the PC type I/O channel have edge-triggered hardware interrupts. On 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 is sent whether the incoming interrupt request to the controller is active or inactive.

In level-sensitive systems, the interrupt-in-progress latch is readable at an I/O address bit position. This latch is read during the interrupt service routine and may be reset by the read operation or may require an explicit reset.

Note: Designers may want to limit the number of devices sharing an interrupt level for performance and latency considerations.

The interrupt controller on level-sensitive systems requires the interrupt request to be inactive at the time the EOI is sent; otherwise, a "new" interrupt request will be detected and another microprocessor interrupt caused.

To avoid this problem, a level-sensitive interrupt handler must clear the interrupt condition (usually by a Read or Write to an I/O port on the device causing the interrupt). After clearing the interrupt condition, a JMP $+2 should be executed prior to sending the EOI to the interrupt controller. This ensures that the interrupt request is removed prior to reenabling the interrupt controller. Another JMP $+2 should be executed after sending the EOI, but prior to enabling the interrupt through the Set Interrupt Flag (STI) instruction.

I/O commands followed immediately by an STI instruction do not permit enough recovery time for some system board and channel operations. To ensure enough time, a JMP SHORT $+2 must be inserted between the I/O command and the STI instruction.

Notes:

1. MOV AL,AH type instructions do not allow enough recovery time. An example of the correct procedure follows:

```
OUT  IO_ADD,AL
JMP  SHORT $+2
MOV  AL,AH
STI
```

2. Prior to programming the interrupt controllers, interrupts should be disabled by issuing a Clear Interrupt Flag (CLI) instruction. This includes programming the Mask register and issuing EOIs, initialization command bytes, and operation command bytes.

In the level-sensitive systems, hardware prevents the interrupt controllers from being set to the edge-triggered mode.

Hardware interrupt IRQ9 is defined as the replacement interrupt level for the cascade level IRQ2. Program interrupt sharing should be implemented on IRQ2, INT 0AH. The following processing occurs to maintain compatibility with the IRQ2 used by IBM Personal Computer products:

1. A device drives the interrupt request active on IRQ2 of the channel.
2. This interrupt request is mapped in hardware to IRQ9 input on the slave interrupt controller.
3. When the interrupt occurs, the system microprocessor passes control to the IRQ9 (INT 71H) interrupt handler.
4. This interrupt handler performs an EOI to the slave interrupt controller and passes control to IRQ2 (INT 0AH) interrupt handler.
5. When handling the interrupt, the IRQ2 interrupt handler causes the device to reset the interrupt request prior to performing an

EOI to the master interrupt controller that finishes servicing the IRQ2 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, that program may 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 may 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 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 the BIOS to accept the same programming interface on the new device.
- In cases where 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 pointed to by INT 1EH consists of 11 parameters required for diskette operation. It is

recommended that the values supplied in ROM be used. If it becomes necessary to modify any of the parameters, build another parameter block and modify the address at INT 1EH (0:78) 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 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 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, the following minimum head settle times are enforced.

Drive Type	**Head Settle Time**
5.25-Inch Diskette Drives:	
Double Sided (48 TPI)	20 milliseconds
High Capacity (96 TPI)	15 milliseconds
3.5-Inch Diskette Drives:	
720KB	20 milliseconds
1.44MB	15 milliseconds

Figure 4-2. Head Settle Time

Read and verify operations use the head settle time provided by the parameter block.

For any function that requires a parameter block containing 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 may be designed to reside on both 5.25-inch and 3.5-inch diskettes. Since not all programs are operating-system dependent, the following procedure can be used to determine the type of media inserted into a diskette drive:

  1. Verify track 0, head 0, sector 1 (1 sector): This allows diskette BIOS to determine if the format of the media is a recognizable type.

     If the verify operation fails, issue the reset function (AH=0) to diskette BIOS and try the operation again. If another failure occurs, 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 3.5-inch 720KB diskette is installed. The type can be determined by verifying track 78, head 1, sector 1 (1 sector). A successful verification of track 78 indicates a 3.5-inch 720KB diskette is installed; a verification failure indicates 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.

Scan Code/Character Code Combinations

Note: Refer to the *Personal System/2 Model 25 Technical Reference* for scan code/character code combinations for the PC Space Saving (84/85-key) Keyboard.

The following lists the keyboard keystrokes and the scan code/character code combinations that are returned through INT 16H:

Keystroke	83- and 84-Key Standard Function	101/102-Key Standard Function	101/102-Key Extended Function
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

Figure 4-3 (Part 1 of 3). Keyboard Keystrokes

Keystroke	83- and 84-Key Standard Function	101/102-Key Standard Function	101/102-Key Extended Function
f	21/66	21/66	21/66
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
‘	29/60	29/60	29/60
Shift	**	**	**
\	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
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

Figure 4-3 (Part 2 of 3). Keyboard Keystrokes

Keystroke	83- and 84-Key Standard Function	101/102-Key Standard Function	101/102-Key Extended Function
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
SysReq	**	(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/E0

** These combinations do not provide a keystroke for the application but perform some other action. They are not put in the INT 16H queue.
-- These combinations have no function and are ignored.

Figure 4-3 (Part 3 of 3). Keyboard Keystrokes

The following lists the Shift keyboard keystrokes and the scan code/character code combinations that are returned through INT 16H:

Keystroke	83- and 84-Key Standard Function	101/102-Key Standard Function	101/102-Key Extended Function
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 4-4 (Part 1 of 3). Shift Keyboard Keystrokes

Keystroke	83- and 84-Key Standard Function	101/102-Key Standard Function	101/102-Key Extended Function
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 SysReq	**	(no key)	(no key)
Shift Key 45	(no key)	56/7C	56/7C

Figure 4-4 (Part 2 of 3). Shift Keyboard Keystrokes

Keystroke	83- and 84-Key Standard Function	101/102-Key Standard Function	101/102-Key Extended Function
Shift Enter	(no key)	1C/0D	E0/0D
Shift /	(no key)	35/2F	E0/2F
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 presently running but perform some other action. They are not put in the INT 16H queue.
-- These combinations have no function and are ignored.

Figure 4-4 (Part 3 of 3). Shift Keyboard Keystrokes

The following lists the Ctrl keyboard keystrokes and the scan code/character code combinations that are returned through INT 16H:

Keystroke	83- and 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 4-5 (Part 1 of 3). Ctrl Keyboard Keystrokes

Keystroke	83- and 84-Key Standard Function	101/102-Key Standard Function	101/102-Key Extended Function
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

Figure 4-5 (Part 2 of 3). Ctrl Keyboard Keystrokes

Keystroke	83- and 84-Key Standard Function	101/102-Key Standard Function	101/102-Key Extended Function
Ctrl F4	61/00	61/00	61/00
Ctrl F5	62/00	62/00	62/00
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 SysReq	**	(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 presently running but perform some other action. They are not put on the INT 16H queue.
-- These combinations have no function and are ignored.

Figure 4-5 (Part 3 of 3). Ctrl Keyboard Keystrokes

The following lists the Alt keyboard keystrokes and the scan code/character code combinations that are returned through INT 16H:

Keystroke	83- and 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 4-6 (Part 1 of 3). Alt Keyboard Keystrokes

Keystroke	83- and 84-Key Standard Function	101/102-Key Standard Function	101/102-Key Extended Function
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 Nos.	#	#	#
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

Figure 4-6 (Part 2 of 3). Alt Keyboard Keystrokes

Keystroke	83- and 84-Key Standard Function	101/102-Key Standard Function	101/102-Key Extended Function
Alt End	(no key)	--	9F/00
Alt Down	(no key)	--	A0/00
Alt PageDown	(no key)	--	A1/00
Alt Insert	(no key)	--	A2/00
Alt Delete	(no key)	--	A3/00

# See the following page for use of Alt key with number keys.
** These combinations do not provide a keystroke for the application presently running but perform some other action. They are not put on the INT 16H queue.
-- These combinations have no function and are ignored.

Figure 4-6 (Part 3 of 3). Alt Keyboard Keystrokes

For all keyboards, the numeric keypad can be used in combination with the Alt key to input any ASCII character. The scan code (always 00) and character code are returned after the Alt key is released. For example, pressing Alt and Keypad 1, then releasing Alt returns scan code/character code combination hex 00/01; pressing Alt and Keypad 255, then releasing Alt returns scan code/character code combination hex 00/FF.

Index

Notes:

Contents Advanced BIOS

Figures Advanced BIOS

Section 1. Introduction to Advanced BIOS

Notes:

Introduction

Advanced BIOS (ABIOS) is firmware that isolates an operating system from the low-level system hardware interface on IBM Personal System/2 systems that use 80286 or 80386 microprocessors. The operating system makes functional requests of ABIOS (read, write) rather than directly manipulating the I/O ports and control words of the system hardware. This allows 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.

The ROM BIOS operates as a single-tasking component whose addressing capabilities are limited to less than 1 megabyte of memory and only in the real address mode (real mode) of the Intel[1] 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 be executed in virtual 8086 mode.

Requests to ABIOS 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 the processing time may be used. An interrupt from an I/O device usually indicates completion of a stage of the operation.

1 Intel is a trademark of Intel Corporation.

The following figure shows the three categories of ABIOS requests.

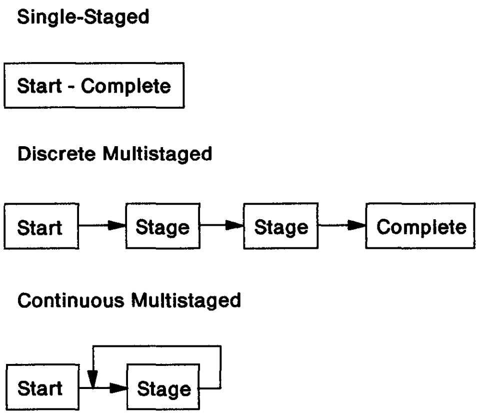


Figure 1-1. Types of Requests

Data Structures

Requests to ABIOS made by an operating system are through transfer conventions provided by the ABIOS structure. These conventions require data structures that link the operating system to the device function routines of each supported device. These data structures include the Common Data Area, Function Transfer Tables, and Device Blocks. They reside in system memory and are initialized during ABIOS initialization.

The transfer conventions provided by ABIOS are defined to allow operations that use the real mode and/or the protected mode of an Intel microprocessor. To provide flexibility in implementing a real mode, protected mode, or bimodal operating system, the Common Data Area links all ABIOS 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 ABIOS data pointers.

ABIOS entry points are stored in vector tables called Function Transfer Tables (see "Function Transfer Table" on page 2-7). Each supported ABIOS device has an associated Function Transfer Table. The first three entries of the Function Transfer Table are structured entry routines called the Start Routine, the Interrupt Routine, and the Time-out Routine.

ABIOS routines require a permanent work area for each device called the Device Block (see “Device Block” on page 2-9). Hardware port addresses, interrupt levels, and device state information are the types of information stored in the Device Block.

Initialization

Initialization is a defined protocol between ABIOS 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, ABIOS cannot be used (see Section 3, “Initialization”). This initialization process must occur in the real mode of the microprocessor, and consists of three steps:

1. The operating system calls BIOS to build the System Parameters Table. This table describes the number of devices available in the system, ABIOS 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 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 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. The operating system then calls ABIOS to build the Device Blocks and Function Transfer Tables for each device.

The flow of the initialization process is shown below.

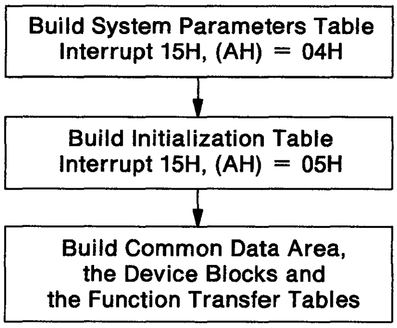

Build System Parameters Table
Interrupt 15H, (AH) = 04H
Build Initialization Table
Interrupt 15H, (AH) = 05H
Build Common Data Area,
the Device Blocks and
the Function Transfer Tables

Figure 1-2. Flow of the Initialization Process

Transfer Conventions

After ABIOS is initialized, requests are presented through a parameter block called the Request Block. The Request Block has fields that identify the target device, requested operation, details of the request, memory locations involved in a data transfer, and the status of the staged/completed request. The Request Block is described in detail in Section 4, "Transfer Conventions."

ABIOS is implemented as a call-return programming model using either the ABIOS Transfer Convention or the Operating System Transfer Convention. These two calling conventions allow an operating system flexibility in calling ABIOS. Both calling conventions use the stack to pass request information to the target ABIOS device routine.

The ABIOS 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, or 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 ABIOS routine whose pointer is in the Function

Transfer Table. The flow of the ABIOS Transfer Convention is shown below.

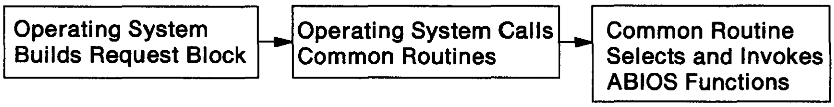

Operating System Builds Request Block
Operating System Calls Common Routines
Common Routine Selects and Invokes ABIOS Functions

Figure 1-3. Flow of ABIOS Transfer Convention

The Operating System Transfer Convention requires the operating system to determine the address for the requested ABIOS routine. This allows the operating system flexibility in maintaining ABIOS 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, Request Block, Function Transfer Table, and Device Block pointers are required on entry to the ABIOS routine. The flow of the Operating System Transfer Convention is shown below.

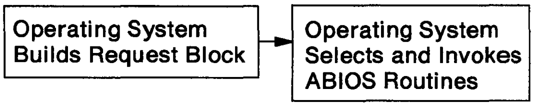

Operating System Builds Request Block
Operating System Selects and Invokes ABIOS Routines

Figure 1-4. Flow of Operating System Transfer Convention

The ABIOS Transfer Convention and 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 ABIOS expects the operating system to receive control . ABIOS provides interrupt routines for the processing of ABIOS interrupts. Interrupt processing is described in “Interrupt Processing” on page 5-3.

Extending ABIOS

The ability to add, patch, extend, and replace ABIOS routines is necessary for supporting new devices or device features on the system. For more information see Section 5, "Additional Information."

Important: Information added to the **Supplements** area of this technical reference may have new information about subjects covered in other parts of this technical reference. Refer to **Supplements** for information that could affect your hardware or software development decisions.

Section 2. Data Structures

Notes:

Introduction

ABIOS uses data structures to link the operating system to the device function routines for each ABIOS device. These data structures are called the Common Data Area, the Function Transfer Table, and the Device Block. They reside in system memory and are initialized during ABIOS initialization.

The transfer conventions used by ABIOS are defined to allow operations that use the real mode and/or protected mode of an Intel microprocessor. ABIOS provides the Common Data Area for implementing a real mode, a protected mode, or a bimodal operating system. This structure contains Function Transfer Table pointers, Device Block pointers, and the ABIOS data pointers. The Common Data Area links all ABIOS pointers in a single structure to allow 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 initialization and contain information for each device the system supports. The Common Data Area is required in all operating modes. These are the protected mode only, real mode only, and bimodal implementations.

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 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 used by ABIOS services.

On each request to ABIOS, a segment or selector with an assumed offset of 0 is passed to ABIOS, which points to the Common Data Area. 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 with the other ABIOS data structures.

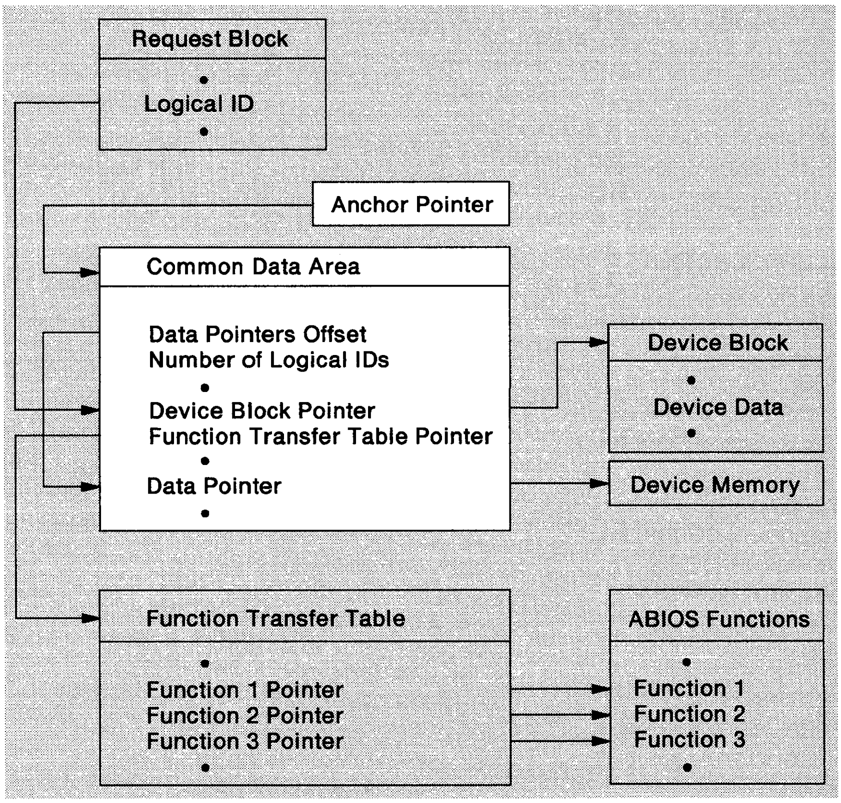


Figure 2-1. Flow of Common Data Area

The following figure shows a detailed representation of the Common Data Area.

Field	Offset	Length
Offset to Data Pointer 0	+00H	2
Count of Logical IDs	+02H	2
Reserved	+04H	4
Device Block Pointer Logical ID 1	+08H	4
Function Transfer Table Pointer Logical ID 1	+0CH	4
Device Block Pointer Logical ID 2	+10H	4
Function Transfer Table Pointer Logical ID 2	+14H	4
.	.	.
.	.	.
Device Block Pointer Logical ID n	+(08H*n)	4
Function Transfer Table Pointer Logical ID n	+(08H*n)+04H	4
Data Pointer p Length	+(08H*n)+08H	2
Data Pointer p Offset	+(08H*n)+0AH	2
Data Pointer p Segment	+(08H*n)+0CH	2
Data Pointer p - 1 Length	+(08H*n)+0EH	2
Data Pointer p - 1 Offset	+(08H*n)+10H	2
Data Pointer p - 1 Segment	+(08H*n)+12H	2
.	.	.
.	.	.
Data Pointer 0 Length	+(08H*n)+(06H*p)+08H	2
Data Pointer 0 Offset	+(08H*n)+(06H*p)+0AH	2
Data Pointer 0 Segment	+(08H*n)+(06H*p)+0CH	2
Data Pointer Count	+(08H*n)+(06H*p)+0EH	2

n - is the number of Logical IDs
p - is 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 Table pointer pairs.

Device Block Pointers: These fields contain the pointers to the Device Blocks for the given Logical IDs.

Function Transfer Table Pointers: These fields contain the pointers to the Function Transfer Tables for the given Logical IDs.

Data Pointer Lengths: These fields contain the lengths of the data areas 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 0:0 after 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 ABIOS extendibility. 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 pointers for each ABIOS function. Reserved function pointers are initialized to 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 with the Common Data Area.

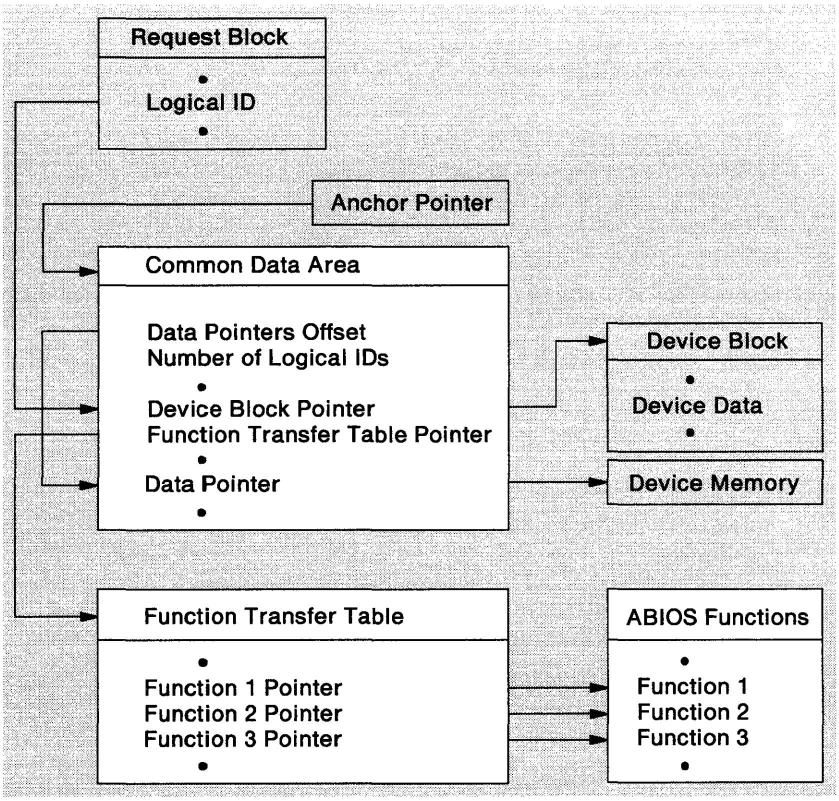


Figure 2-3. Flow of Function Transfer Table

The operating system builds a Request Block, including the Logical ID (defines which device) and Function (defines which function). Based on the information contained 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 Function Transfer Table.

Function	**Offset**	**Length**
Start Routine Pointer	+00	4
Interrupt Routine Pointer	+04	4
Time-out Routine Pointer	+08	4
Function Count	+0C	2
Reserved	+0E	2
Function 1 Routine Pointer	+10	4
Function 2 Routine Pointer	+14	4
.	.	.
.	.	.
Function n Routine Pointer	+0C+(4*n)	4

n - is 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, and 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, and 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 0:0.

Time-out Routine Pointer: The Time-out Routine pointer is a doubleword pointer, and is called (using Call Far Indirect) to terminate a request that fails to receive a hardware interrupt in a specified time. This routine aborts 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 a device that interrupts but never times out, the Time-out Routine Pointer field is initialized to 0:0.

Function Count: This is a word count of the number of functions supported by a device.

Reserved: This is a reserved word (allocated even if the Function Count field is equal to zero).

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, per device, called the Device Block. Hardware port addresses, interrupt levels, and device status information are the types of information stored in the 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. This area should not be altered by the operating system. Private data in the Device Block is used internally by ABIOS and its format and content may not be identical in all implementations of ABIOS. The operating system should 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 with the Common Data Area.

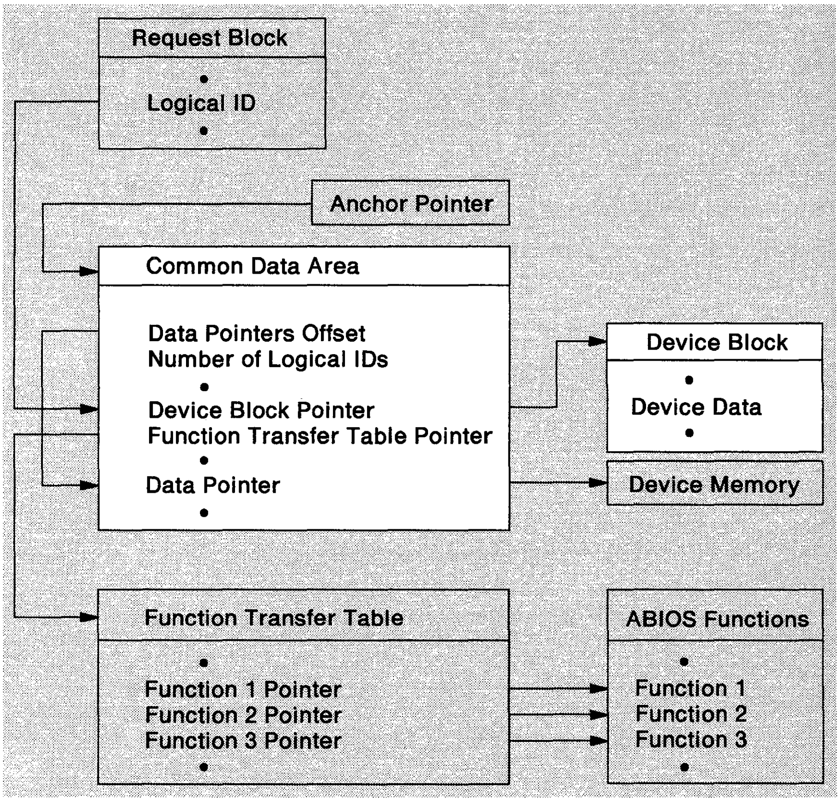


Figure 2-5. Flow of Device Block

Every ABIOS device has an associated Device Block. The Device Block is shown in the following figure.

Field	Offset	Length	Access
Device Block Length	+00H	2	Public Read
Revision	+02H	1	Public Read
Secondary Device ID	+03H	1	Public Read
Logical ID	+04H	2	Public Read
Device ID	+06H	2	Public Read
Count of Logical ID Exclusive Port Pairs	+08H	2	Public Read
Count of Logical ID Common Port Pairs	+0AH	2	Public Read
Logical ID Exclusive Port Pairs 0	?	4	Public Read
Logical ID Exclusive Port Pairs 1	?	4	Public Read
.			
.			
Logical ID Exclusive Port Pairs n	?	4	Public Read
Logical ID Common Port Pairs 0	?	4	Public Read
Logical ID Common Port Pairs 1	?	4	Public Read
.			
.			
Logical ID Common Port Pairs n	?	4	Public Read
Device Unique Data Area Length	?	2	Private
Device Unique Data Area	?	?	Private
Count of Units	?	2	Private
Unit Unique Data Area Length	?	2	Private
Unit Unique Data Area	?	?	Private

? - is a placeholder for variable values
n - is the count of port pairs

Figure 2-6. Device Block

The Device Block entries are:

Device Block Length: This field is a word containing the number of bytes in the Device Block, including the Device Block Length field. The maximum specifiable length is 64KB minus 1. The required size of the Device Block for a particular device is returned during ABIOS initialization.

Revision: This byte is used to indicate the level of the supporting code for a device. The initial value of the base level is 0. For each succeeding version of ABIOS code for a particular Device ID and Secondary Device ID, the Revision field is increased by 1 (that is, the Revision field is increased by 1 if a new level of ABIOS code is developed for existing hardware).

Secondary Device ID: This byte is used to determine the level of hardware that an ABIOS implementation supports. The initial value of the base level is 0. 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 Secondary Device ID field is increased, the Revision field reverts to 0.

Logical ID: Logical ID indicates the logical name of the device associated with a Device Block. It is analogous to the software interrupt number used by BIOS to access different device types. Logical ID values are determined dynamically during initialization and 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 ABIOS functions, the operating system is required to reserve a number of Logical IDs to allow ABIOS to call these common internal ABIOS functions. They are identified to the operating system during initialization. The Logical ID values are shown in the following figure.

Usage	**Logical ID**
Reserved	00H
Reserved	01H
ABIOS Internal Calls	02H through n
System and Adapter Devices	> n

n - is the last Logical ID reserved for ABIOS internal calls

Figure 2-7. Logical ID Values

Device ID: Device ID indicates the type of device addressed by a function request and the level of ABIOS function that is supported. The assigned values of this field are shown below.

Device	**Device ID Value**
ABIOS Internal Calls	00H
Diskette	01H
Disk	02H
Video	03H
Keyboard	04H
Parallel Port	05H
Asynchronous Communications	06H
System Timer	07H
Real-Time Clock Timer	08H
System Services	09H
Nonmaskable Interrupt	0AH
Pointing Device	0BH
Reserved	0CH
Reserved	0DH
Nonvolatile Random Access Memory (NVRAM)	0EH
Direct Memory Access (DMA)	0FH
Programmable Option Select (POS)	10H
Reserved	11H - 15H
Keyboard Security	16H
Reserved	17H - FFFFH

Figure 2-8. Device ID Values

The value hex 00 of the Device ID field is reserved for ABIOS internal calls; that is, an ABIOS function calling another ABIOS function. All other Device ID values denote a type of device as well as a level of ABIOS support, as described in Section 6, "Interfaces."

Count of Logical ID Exclusive Port Pairs: This is the count of Logical ID exclusive port pairs. Logical ID exclusive ports are ports used exclusively by a particular Logical ID. Examples are the diskette ports, disk ports, asynchronous communication ports, parallel ports, and video ports. If the Count of Logical ID Exclusive Port Pairs field equals 0, no space is allocated for the Logical ID Exclusive Port Pairs fields.

Count of Logical ID Common Port Pairs: This is the count of Logical ID common port pairs. The 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 this field equals 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 ABIOS 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 this device.

Device Unique Data Area: This field is reserved for data unique to a device. Parameters describing the device and working data that span the Device ID are kept in this area. This area contains private data for ABIOS and its content or format may change. Examples of the data kept in this area are interrupt level, arbitration level, and device status.

Count of Units: This field contains the count of the unit unique data areas in the Device Block. If this field equals 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 if the Count of Units field is greater than 0.

Unit Unique Data Area: This field is a private repeatable area reserved for each unit of the Device ID. For example, if the diskette ID value is hex 01 (Diskette), a particular diskette drive is a unit. Parameters describing the unit and working data that span individual requests are kept in this area. This area is private data for ABIOS and its content or format may change. This field exists only if the Count of Units field is greater than 0.

Section 3. Initialization

Notes:

Introduction

ABIOS is initialized in an “on demand” fashion. The operating system makes specific calls to BIOS and ABIOS to achieve the startup. 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, and 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.

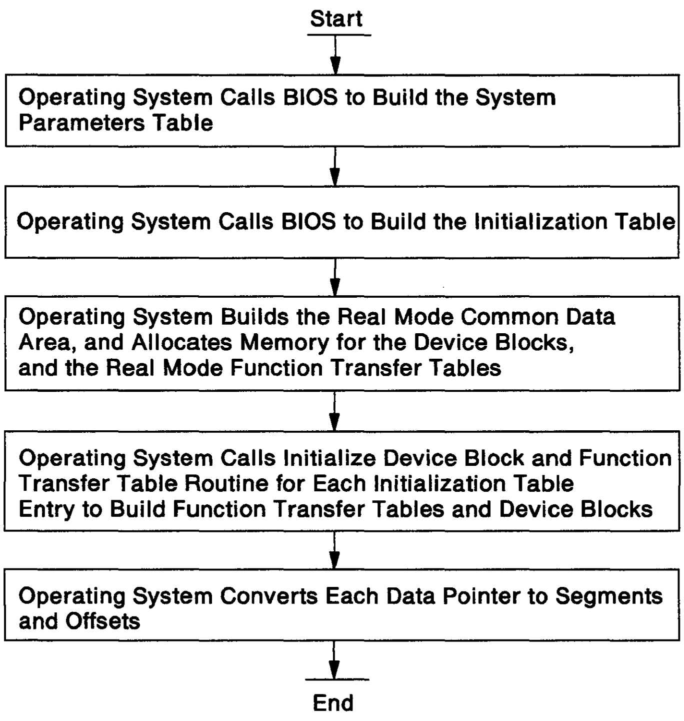

Start
Operating System Calls BIOS to Build the System Parameters Table
Operating System Calls BIOS to Build the Initialization Table
Operating System Builds the Real Mode Common Data Area, and Allocates Memory for the Device Blocks, and the Real Mode Function Transfer Tables
Operating System Calls Initialize Device Block and Function Transfer Table Routine for Each Initialization Table Entry to Build Function Transfer Tables and Device Blocks
Operating System Converts Each Data Pointer to Segments and Offsets
End

Figure 3-1. Flow of Real Mode Common Data Area Initialization

Build System Parameters Table - Operating System

The operating system allocates a hex 20-byte area and calls BIOS to build the System Parameters Table. This table describes the number of devices available in the system, the ABIOS common entry points, and system stack requirements.

```
INT 15H, (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 0 Offset
                         to RAM extension area (points to
                         a RAM extension with length=0 for
                         no RAM extensions).

On Return:        (CY) = 1 Indicates an exception error

                  (AH) =  0 for no errors

               (All registers except AX and FLAGS are restored.)
```

Figure 3-2. Build System Parameters Table BIOS Function

Build System Parameters Table - BIOS

When 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 with 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 with 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).

Once the System Parameters Table information is obtained, the memory allocated for this table may be deallocated and reused by the operating system. The System Parameters Table is shown below.

Field	**Offset**	**Length**
Common Start Routine Pointer	+00H	4
Common Interrupt Routine Pointer	+04H	4
Common Time-out Routine Pointer	+08H	4
Stack Required	+0CH	2
Reserved	+0EH	4
Reserved	+12H	4
Reserved	+16H	4
Reserved	+1AH	4
Number of Entries	+1EH	2

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 containing the amount of stack memory, in bytes, that is required for a particular ABIOS implementation.

Number of Entries: This field is a word containing the number of entries required in the Initialization Table.

Build Initialization Table - Operating System

The Initialization Table defines the initialization information for each device 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 required for the Initialization Table in bytes is hex 18 times the number of entries in the Initialization Table. The Number of Entries field in the System Parameters Table is used for this calculation. When the initialization process is complete the memory allocated for the Initialization Table can be deallocated and reused by the operating system.

INT 15H, (AH) = 05H BUILD INITIALIZATION TABLE

Invocation: Software interrupt, operating system calls BIOS.

(ES:DI) = Pointer to caller's memory where the Initialization Table is to be built.

(DS) = Segment with assumed 0 offset to RAM extension area (points to a RAM extension with length = 0 for no RAM extensions).

On Return: (CY) = 1 Indicates exception error

(AH) = 0 for no errors

(All registers except AX and 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 whose code resides 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 whose code resides 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.

Field	Offset	Length
Device ID	+00H	2
Number of Logical IDs	+02H	2
Device Block Length	+04H	2
Initialize Device Block and Function Transfer Table Routine Pointer	+06H	4
Request Block Length	+0AH	2
Function Transfer Table Length	+0CH	2
Data Pointers Length	+0EH	2
Secondary Device ID	+10H	1
Revision	+11H	1
Reserved	+12H	2
Reserved	+14H	2
Reserved	+16H	2

Figure 3-5. Initialization Table

The Initialization Table entries are:

Device ID: For a list of the values of the Device ID fields see Figure 2-8 on page 2-13. There may be more than one entry in the Initialization Table with the same Device ID.

Number of Logical IDs: This is a word containing 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 containing the length, in bytes, of the storage allocation required for the Device Block for this device. A Device Block Length of 0 indicates that this Initialization Table entry is for an ABIOS patch or extension, and no Device Block is required 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 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 containing the length, in bytes, of the storage allocation required for the Request Block for this device. When making a request to ABIOS, any Request Block size greater than the size returned is valid.

Function Transfer Table Length: This is a word containing 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 ABIOS patch and no Function Transfer Table data area is to be allocated. When the Function Transfer Table Length field 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 containing the length, in bytes, of the storage allocation required for the data pointer fields in the Common Data Area.

Secondary Device ID: This is a byte used to determine the level of hardware that an ABIOS 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 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 0 within a segment, and allocates memory for each Device Block and Function Transfer Table. Memory is allocated within 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 within 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 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 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 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 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, corresponding 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 following parameters: the Anchor Pointer, the Starting Logical ID, and the Number of Logical IDs to Initialize.

The Initialize Device Block and Function Transfer Table Routines are called in the order their pointers appear in the Initialization Table. This results in system board ROM devices being initialized prior to 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 system board services that may 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 needed. Once the Device ID is found, the Logical ID number contained within the public portion of the Device Block is used for all subsequent requests to the system board ABIOS service.

The operating system needs only to call the Initialize Device Block and Function Transfer Table routines for the devices that are to be made operational. The operating system can make the determination of whether or not to initialize an ABIOS device based upon the values within 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 equal to 0 must be initialized to ensure that internal ABIOS calls are supported. There are also Device IDs that may be required to be initialized to support other Device IDs. For example, the DMA ABIOS is required to be initialized if the Disk ABIOS is initialized. These requirements are defined in Section 6, "Interfaces."

When 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 Number of Logical IDs field.

INITIALIZE DEVICE BLOCK AND FUNCTION TRANSFER TABLE ROUTINE

Invocation:	Call FAR, operating system calls ABIOS on system board ROM, on adapter ROM, or RAM extension, depending on device.
	(CX) = Number of Logical IDs to Initialize (up to the Number of Logical IDs field from the Initialization Table)
	(DX) = Starting Logical ID
	(DS) = Anchor pointer to the Common Data Area
On Return:	(AL) = Exception condition, 0 for no errors 00H Successful completion 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 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 placed in the Function Transfer Table must equal the segment of its corresponding ROM header or RAM extension header, respectively. This allows an operating system to access the Length in 512-Byte Blocks field of the ROM header or the RAM extension header to determine the segment limit in a bimodal or protected mode environment. When building the protected mode Common Data Area, if offset 0 of the ROM header segment or RAM extension header segment contains the ROM/RAM signature, offset 2 contains the length in 512-byte increments (limit hex 7F). This value is used to calculate the segment limit.

After filling in the Function Transfer Table, 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 Number of Logical IDs to Initialize parameter (CX).

Upon return from the Initialize Device Block and Function Transfer Table routine, if the Exception Condition parameter (AL) is nonzero indicating an error, the appropriate action is to deallocate the associated Device Blocks and Function Transfer Table areas and replace the associated Device Block pointers and Function Transfer Table pointers with 0:0, making those entries null Common Data Area entries.

Data Pointers

The Initialize Device Block and Function Transfer Table routine stores all necessary ABIOS data pointers in the data pointer portion of the Common Data Area. As the data pointers are stored, the Data Pointer Count field is incremented. The offset to the stored data pointer within the Common Data Area may be stored in the Device Block as a handle to the data pointer.

Data pointers are initialized by ABIOS as 32-bit physical addresses stored in Intel format of low word, high word at the Data Pointer Offset field. Preceding this 32-bit physical address is the Data Pointer Length field, that indicates the segment limit for a protected mode or bimodal implementation. In bimodal implementations, a 0:0 data pointer value 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 prior to making any requests to ABIOS.

Logical ID 2 Initialization

Reserved data pointers are initialized by the call to the Initialize Device Block and Function Transfer Table routine for Logical ID 2. Logical ID 2 is reserved for ABIOS internal calls (Device ID = 0). 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	1st 64KB of System Board ROM
2	F0000H	FFFFH	2nd 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 ABIOS 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, Common Interrupt Routine, and Common Time-out Routine pointers at the Start, Interrupt, and Time-out pointers within the Function Transfer Table for Logical ID 2. The Initialization Table entry for the first Device ID equal to 0 must have a Function Transfer Table length of at least hex 10 (greater if ABIOS internal functions exist) to allow for the three doubleword pointers, a word count of functions, and a reserved field.

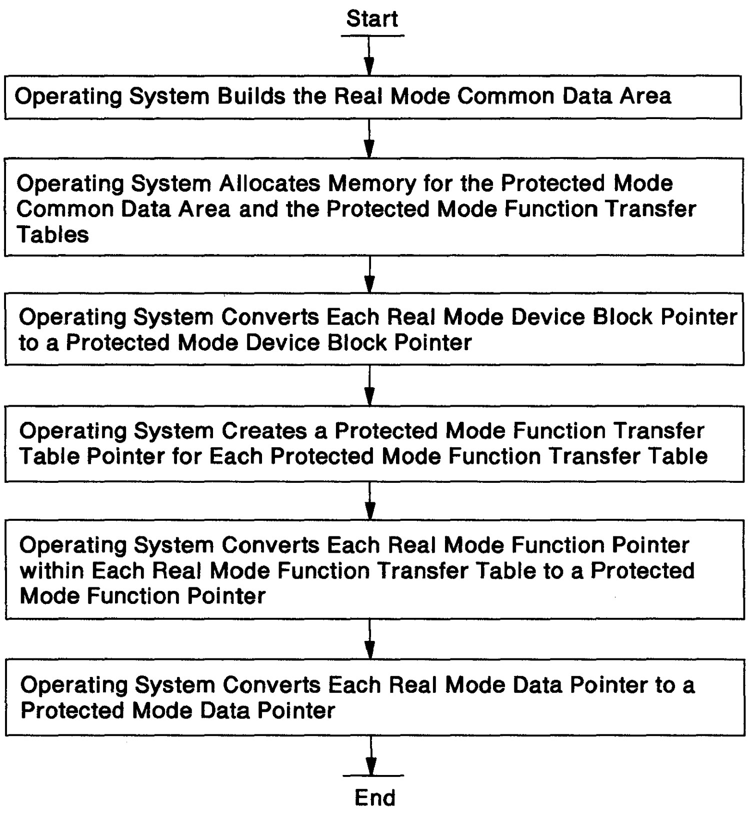
The Function Transfer Table for Logical ID 2 with a value of the Function Count field equal to 0 is shown below.

Field	Offset	Length
Common Start Routine Pointer	+00H	4
Common Interrupt Routine Pointer	+04H	4
Common Time-out Routine Pointer	+08H	4
Function Count (equals 0)	+0CH	2
Reserved	+0EH	2

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 built in 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 address is identical to their corresponding segments in the real mode Common Data Area and Function Transfer Tables. The following diagram describes the steps necessary to build the protected mode Common Data Area.



Start
Operating System Builds the Real Mode Common Data Area
Operating System Allocates Memory for the Protected Mode Common Data Area and the Protected Mode Function Transfer Tables
Operating System Converts Each Real Mode Device Block Pointer to a Protected Mode Device Block Pointer
Operating System Creates a Protected Mode Function Transfer Table Pointer for Each Protected Mode Function Transfer Table
Operating System Converts Each Real Mode Function Pointer within Each Real Mode Function Transfer Table to a Protected Mode Function Pointer
Operating System Converts Each Real Mode Data Pointer to a Protected Mode Data Pointer
End

Figure 3-9. Flow of Protected Mode Common Data Area Initialization

To build descriptors 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 in 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 Pointer Length field located in each data pointer entry in the Common Data Area.

The pointers to ABIOS functions located in the Function Transfer Table must be readable code segment descriptors whose conforming bit is determined by the operating system. If offset 0 of the ROM header segment or the RAM extension header segment contains the ROM/RAM signature, offset 2 contains the length in 512-byte increments (limit hex 7F). This value is to be used as the segment limit. If the ROM/RAM signature does not exist the segment limit is to be hex FFFF.

If ABIOS is called as a conforming code segment and is called by multiple privilege levels, the operating system is responsible for ensuring that ABIOS has I/O privilege at all times.

When the protected mode version of the Common Data Area is built, each Common Data Area entry in the protected mode version, whose corresponding entry in the real mode version is a null Common Data Area entry, must be 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, with a corresponding entry in the real mode version of 0:0, must have a 0:0 value 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 and the protected mode Common Data Areas must point to the same Device Block.

Notes:

Section 4. Transfer Conventions

Notes:

Introduction

ABIOS can be implemented in three environments: protected mode-only, real mode-only, and bimodal. ABIOS requires a method of transferring control from the caller of ABIOS to ABIOS without sacrificing performance. The two methods provided for this transfer are the ABIOS 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 or passes parameters to ABIOS.

Request Block

The Request Block is a parameter block used to communicate information bidirectionally between the caller and an ABIOS service. Parameters are passed by the caller (IN) and returned by ABIOS (OUT).

The following diagram shows the Request Block and its relationship with a Common Data Area.

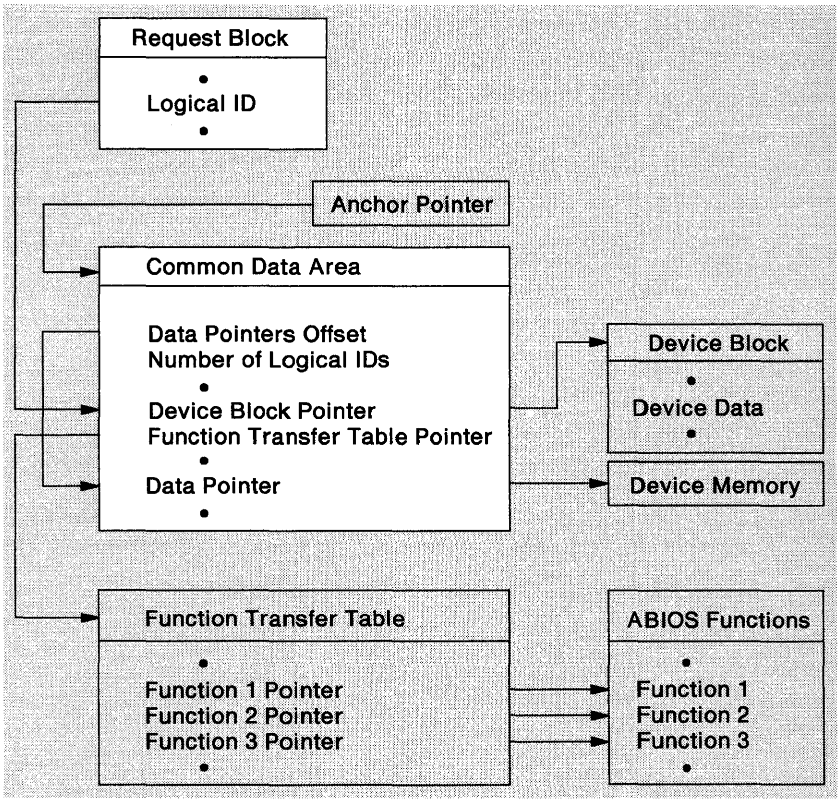


Figure 4-1. Flow of Request Block

All input parameters (IN) are unaltered by ABIOS throughout the duration of a request. All output parameters (OUT) and work areas need not be set to any predefined value before ABIOS is called. This allows Request Blocks to be reused after requests are completed. This requires that any Work Area fields containing request state information be initialized by the ABIOS Start routines to the predefined values. Only input (IN) or input/output (IN/OUT) parameters that change between requests are required to be initialized before the Request Block is reused. All reserved input fields must be set to 0 by the caller of ABIOS. The parameters are divided into two categories: functional parameters and service specific parameters.

Functional Parameters

Functional parameters are common to all ABIOS service requests. They convey information to ABIOS about which service should be invoked on which device. Each input parameter is initialized by the caller and, once initialized, must remain unaltered until the requested operation is complete. Functional parameters include the Request Block Length field through the Time-out field, as shown in Figure 4-2 on page 4-6.

Service Specific Parameters

Service specific parameters are specific to an ABIOS request. The details of parameters passed by the caller and parameters returned by ABIOS depend on the service requested. The service specific parameters include the Data Pointer 1 field through the Work Area field, as shown in Figure 4-2 on page 4-6.

Request Block Structure

The structure of a Request Block containing functional parameters and service specific parameters is shown below.

Field	Offset	Length
Functional Parameters		
Request Block Length (IN)	+00H	2
Logical ID (IN)	+02H	2
Unit (IN)	+04H	2
Function (IN)	+06H	2
Reserved	+08H	2
Reserved	+0AH	2
Return Code (IN/OUT)	+0CH	2
Time-out (OUT)	+0EH	2
Service Specific Parameters		
Reserved	+10H	2
Data Pointer 1 (IN)	+12H	4
Reserved	+16H	2
Reserved	+18H	2
Data Pointer 2 (IN)	+1AH	4
Parameters (IN/OUT)	+1EH	?
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. The Request Block Length field is a fixed value initialized by the caller for the specific Logical ID. The size of the Request Block for a Logical ID is returned at ABIOS initialization time and by the Return Logical ID Parameters function (hex 01). However, the Request Block may be larger than the returned size.

Logical ID (IN): The Logical ID field indicates the particular device addressed by a function request. It is analogous to a software interrupt number used by BIOS to access different device types.

Unit (IN): The Unit field is a parameter that addresses a particular unit of a device within a Logical ID. The range of valid values is limited by the number of units attached to a single controller. The maximum unit number is n-1, where n is the count of units attached to the controller. The minimum number of units is one, resulting in a Unit field equal to 0.

Function (IN): The Function field is a parameter used to request a particular category of operation. The assignment of functions is:

Function	*Function Performed*
00H	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 for Operation Completed Successfully or hex 0005 for Not My Interrupt. For more information on the Default Interrupt Handler, see "Default Interrupt Handler" on page 5-5.
01H	Return Logical ID Parameters - This function is a standard, single-staged function common to all ABIOS Device IDs. It returns information pertaining to the Logical ID, and 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

Byte    10H       Hardware interrupt level
                    FFH = Noninterrupting Logical ID
                    FEH = Special case for NMI
Byte    11H       Arbitration level
                    FFH = Not applicable
Word    12H       Device ID
Word    14H       Count of Units
Word    16H       Logical ID flags
                  Bit 15 to 4 = Reserved
                  Bit 3       = Overlapped I/O across units
                                0 - Not supported
                                1 - Supported
                  Bit 2       = Reserved
                  Bits 1, 0   = Function Read/Write/Additional Data
                                  Transfer Data Pointer mode
                                  00 - No Read/Write/Additional Data Transfer
                                       Functions supported
                                  01 - Data Pointer 1, Logical
                                       Data Pointer 2, Reserved
                                  10 - Data Pointer 1, Reserved
                                       Data Pointer 2, Physical
                                  11 - Data Pointer 1, Logical
                                       Data Pointer 2, Physical
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       Reserved
Word    1EH       Reserved
```

Logical ID flags contain 2 bits that indicate the mode (physical vs. logical) of the data pointer for the Read (hex 08), the Write (hex 09), and the Additional Data Transfer (hex 0A) functions. 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 the logical pointer and Data Pointer 2 is a physical pointer. If the parameter indicates neither, this Logical ID does not support functions hex 08, 09, and 0A, or these functions require no address pointers. There is no space reserved for data pointers in the Request Block in this event.
02H	Reserved.
03H	Read Device Parameters - Device specific parameters are returned.
04H	Set Device Parameters - Device specific parameters are set.
05H	Reset/Initialize - Device is placed in a known state.
06H	Enable - Device is enabled for interrupts (not at interrupt controller).
07H	Disable - Device is disabled for interrupts (not at interrupt controller).
08H	Read - Data is transferred from device to memory. Data Pointer mode is determined by the function Return Logical ID Parameters.
09H	Write - Data is transferred from memory to device. Data Pointer mode is determined by the function Return Logical ID Parameters.
0AH	Additional Data Transfer Function - Data Pointer mode is determined by the function Return Logical ID Parameters.
0BH and Up	Additional Functions - as necessary.

Device specific functions are specified in detail in Section 6, "Interfaces."

Return Code (IN/OUT): Return Code is a field that contains the results of the current stage of the requested operation. For those operations that are single-staged or those that are on 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 Values	Definition
0000H	Operation Completed Successfully
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
8001-8FFFH	Service Specific Unsuccessful Operation
9000-90FFH	Device Error
9100-91FFH	Retryable Device Error
9200-9FFFH	Device Error
A000-A0FFH	Time-out Error
A100-A1FFH	Retryable Time-out Error
A200-AFFFH	Time-out Error
B000-B0FFH	Device Error with Time-out
B100-B1FFH	Retryable Device Error with Time-out
B200-BFFFH	Device Error with Time-out
C000H	Invalid Logical ID
C001H	Invalid Function
C002H	Reserved
C003H	Invalid Unit Number
C004H	Invalid Request Block Length
C005-C01FH	Invalid Service Specific Parameter
C020-FFFEH	Service Specific Unsuccessful Operation
FFFFH	Return Code Field Not Valid

Figure 4-3. Return Codes

The bits within 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 - 9	Reserved
8	Retryable Error
7	Unexpected Interrupt Reset
6 - 4	Reserved
3	Attention
2	Not My Interrupt
1	Stage On Time
0	Stage On Interrupt

Notes:

1. Bits 14 to 8 are defined as above only when Bit 15 equals 1.
2. Bits 7 to 0 are defined as above only when Bit 15 equals 0.
3. If all bits equal 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 Return Code Field Not Valid (hex FFFF) before calling any ABIOS Start routine. If the operating system has an outstanding Request Block at interrupt time, it first checks for a Return Code field equal to Return Code Field Not Valid (hex FFFF), and if it is, the operating system considers the Return Code field as 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 of writing to the interrupt enable port and changing the Return Code field from a value of Return Code Field Not Valid (hex FFFF) to a value of the Return Code field with the Stage On Interrupt bit (bit 0) set. After changing the Return Code field, the interrupt flag is restored to the value contained prior to disabling it.

When the hardware interrupt occurs, the caller only responds to those requests that have a value of the Return Code field with the Stage On

Interrupt bit (bit 0) set. The outstanding requests with a Return Code field equal to Return Code Not Valid (hex FFFF) are not called.

The caller should also maintain a flag that indicates whether or not a request has completed the Start routine to the point at which the Return Code field is interrogated. This allows for the situation when the interrupt occurs after the Return Code field is valid (not hex FFFF) but before the Return Code field is interrogated by the caller. At this point there could be a Start routine and an Interrupt routine operating on the same Request Block within different stack frames, necessitating the caller's flag.

Attention (hex 0009) and Stage on Time (hex 0002) are values of the Return Code field that need only be tested by services that require them. Attention (hex 0009) indicates that there is data available in a service specific output parameter although the function is not complete. Stage on Time (hex 0002) indicates that the operation is incomplete and must be resumed when a certain amount of time has elapsed. This amount of time is contained in a service specific output parameter depending on the service. In addition, the values of the Return Code field with bit 15 equal to 1 are service specific. These values are documented in Section 6, "Interfaces."

The return code value Device In Use, Request Refused (hex 8000) is used for device serialization. If a Logical ID/Unit combination is a serially reusable device, ABIOS returns this return code value when there is an outstanding request on this device.

Time-out (OUT): The Time-out field contains the expected duration of the requested stage. This is used to detect 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 through 3. Bits 2 through 0 of this field are reserved. A value of 0 indicates the operation has no time-out value. The Time-out field is valid for the value of the Return Code field with the Stage on Interrupt bit (bit 0) set.

Data Pointer 1, Data Pointer 2 (IN): Data pointers if required, are doubleword pointers to I/O buffer areas for this request. The effective address must be addressable in the current mode of the microprocessor in a bimodal environment. The address may be a 32-bit physical address for DMA, or segmented for programmed I/O. The Return Logical ID Parameters function returns a parameter that indicates the mode (physical or logical) of the data pointer for the functions Read (hex 08), Write (hex 09), and Additional Data Transfer (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 the logical pointer and Data Pointer 2 is a physical pointer. If the parameter indicates neither, this Logical ID does not support functions hex 08, 09 and 0A, or these functions require no address pointers. No space is reserved for data pointers in the Request Block in this event.

Parameters (IN/OUT): Parameters communicate operands and, in some cases, results of ABIOS functions. Parameter requirements vary by device and function requested. Detailed parameter requirements are documented in Section 6, "Interfaces."

Work Area: Work Area fields are an optional data area reserved for ABIOS. No user data may be stored here. Their content varies by the type of request and the particular device routine involved. These fields are not required to be initialized to any value. Their content must not be altered by the caller of ABIOS across multistaged requests. Work Area fields are those fields that are not defined as service specific input or service specific output parameters in Section 6, "Interfaces."

ABIOS Transfer Convention

The ABIOS Transfer Convention places the requirement on ABIOS to determine the effective address of a particular ABIOS function. ABIOS indexes into the Common Data Area based upon the Logical ID field in the Request Block to access the necessary pointers including the effective Routine pointer (Start, Interrupt, or Time-out). The ABIOS Transfer Convention is the simplest calling sequence for the operating system. The flow of an ABIOS Transfer Convention request is shown below.

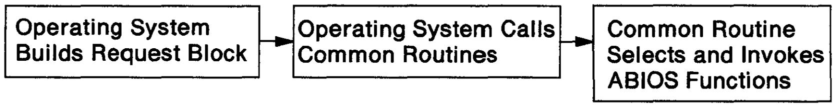

Operating System Builds Request Block
Operating System Calls Common Routines
Common Routine Selects and Invokes ABIOS Functions

Figure 4-5. Flow of ABIOS Transfer Convention

For this transfer convention, there are only three routines by which the caller can transfer control to ABIOS. The pointers to these three routines are returned in the System Parameters Table at initialization time. 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 within 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 Invalid Logical ID (hex C000).

- **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 in a specified time. The Time-out routine aborts 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 assumed 0 offset. 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 ABIOS common routines expect the addresses 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. ABIOS Transfer Convention Stack Frame

The following pseudo code 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
```

Pseudo Code - ABIOS 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 is to be used. These routines take this pair of pointers and place them in the stack placeholder positions 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 ABIOS

Transfer Convention, it is the responsibility of the caller to remove the parameters from the stack upon return.

The layout of the Function Transfer Table is contained in Figure 2-4 on page 2-8.

Operating System Transfer Convention

This convention places the requirement on the operating system to determine the effective address of a particular routine. This method is most useful for handling interrupts from character and programmed I/O devices that repeatedly call a single routine.

There are two methods to accomplish operating system transfers. In the first method, the operating system indexes into the Common Data Area based upon the Logical ID to access the necessary pointers including the effective Routine pointer (Start, Interrupt, or Time-out). The advantage of this approach over the ABIOS Transfer Convention is one of performance.

In the second method, the operating system stores the necessary pointers as it sees fit and accesses them without indexing into the Common Data Area. An operating system might want to use this method if that operating system 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. The flow of an Operating System Transfer Convention request is shown below.

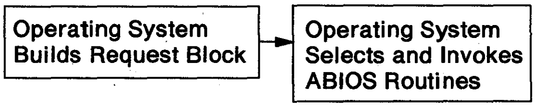

Operating System Builds Request Block
Operating System Selects and Invokes ABIOS Routines

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 is the Anchor pointer segment or selector of the Common Data Area with an assumed 0 offset. The second is a doubleword pointer to the Request Block. The third is a doubleword pointer to the Function Transfer Table, and the fourth is a doubleword pointer to the Device Block.

The Start, Interrupt, and Time-out routines for each Logical ID expect the addresses from high to low (the order of pushing) as shown in the following figure.

Contents	**Displacement (from Stack Pointer)**
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 pseudo code 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
```

Pseudo Code - Operating System Transfer Convention

For the Operating System Transfer Convention, it is the responsibility of the caller to remove the parameters from the stack upon return.

Notes:

Section 5. Additional Information

Notes:

Interrupt Processing

Interrupt Flow

The operating system that interfaces with ABIOS provides interrupt handlers that receive control through the hardware interrupt vector. The operating system interrupt handler is required to retain the Logical IDs of the devices that operate on the given 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 may have one or more Request Blocks active when the 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. The Return Code field is set by ABIOS to indicate if 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 that interrupts are required to be disabled. If no Request Blocks have the Stage On Interrupt bit (bit 0) of the Return Code field set, and an interrupt occurs, the Default Interrupt Handler is provided to remove the interrupt at the device.

Interrupt Sharing

Where 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 the end of interrupt (EOI) processing at the interrupt controller. The method used for EOI processing is entirely up to the operating system. ABIOS does not reset the interrupt controller. The operating system can choose its desired 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 responds that the interrupt was serviced. A serviced interrupt request returns from the Interrupt routine with the Return Code field having any value other than Not My Interrupt, Stage On Interrupt (hex 0005).

Rules for Interrupt Processing

One Interrupt Level per Logical ID: Every unit within 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, ABIOS is returned to the microprocessor mode (real or protected) that it was running at interrupt time. That is, after being preempted in the middle of a request stage, it will be returned to the microprocessor mode that it was running at the time of preemption.

Microprocessor Mode Changes Hidden from ABIOS: While ABIOS 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 mode X, ABIOS can be called through the Start routine in mode Y.

ABIOS Preserves Microprocessor Interrupt Flag State: ABIOS does not change the state of the interrupt flag. ABIOS may temporarily disable the interrupt flag but will restore it to its original state. ABIOS never enables the interrupt flag if it is disabled upon entry to ABIOS.

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 (those pointers in the Request Block in the form segment:offset or selector:offset) can be changed or moved across stages of a request. In the real mode, the segment and/or offset can be changed. In the protected mode, the selector and/or offset can be changed as well as the physical address located in the descriptor.

Operating System Performs EOI: ABIOS does not perform end of interrupt processing (EOI) on its own behalf. In a level-sensitive interrupt environment, the device condition causing the interrupt is reset by ABIOS when it processes the Request Block at the Interrupt routine.

Return Code Indicates Reset of Interrupt Condition: The caller of ABIOS can perform end of interrupt (EOI) processing when ABIOS returns with a successful Return Code during processing of the interrupt as long as all outstanding Request Blocks for that Logical ID have been processed. If the Return Code field is any field other than Not My Interrupt, Stage On Interrupt (hex 0005), and all Request Blocks are 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 both an actual request or the Default Interrupt Handler resets the interrupting condition at the hardware if the Return Code field is any other value than Not My Interrupt, Stage on Interrupt (hex 0005).

Exhaustive Calling: The caller is required to call ABIOS with each outstanding request per Logical ID at interrupt time until the first Logical ID with an interrupt is completely processed. Completely processed means the act of calling each request that has a value of the Return Code field with the Stage On Interrupt bit (bit 0) set for a given Logical ID.

If multiple outstanding requests per Logical ID are waiting for an interrupt, regardless of whether any single request returns indicating that the interrupt was serviced, it is necessary to call each of the requests. This is because resetting the interrupting condition for the first request may reset the interrupt of the second request, causing a loss of interrupt. Exceptions to this rule are defined in Section 6, "Interfaces." An example 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, it is necessary to handle unexpected hardware interrupts by not only resetting the interrupt at the interrupt controller (EOI), but also resetting it at the device. ABIOS provides this capability through the use of the Default Interrupt Handler.

Each interrupting ABIOS service provides a Default Interrupt Handler that resets the interrupt at the device and returns with a Return Code field equal to Operation Completed Successfully (hex 0000) or Not My Interrupt, Stage On Interrupt (hex 0005). The Default Interrupt Handler is passed a Request Block with no service specific parameters, and control is transferred to the Default Interrupt Handler through the Interrupt routine. The Default Interrupt Handler is only called if a given Logical ID has no outstanding Request Blocks waiting on interrupt.

To determine whether or not a Logical ID interrupts, a call to return Logical ID Parameters function is invoked. If the Interrupt Level field pertains to a device that interrupts, it contains the hardware interrupt level. If the device does not interrupt, the Interrupt Level field contains a value of hex FF, indicating a noninterrupting Logical ID. The nonmaskable interrupt (NMI) device is a special case and returns with a value of hex FE for the interrupt level. If the value of hex FE or FF is returned for the interrupt level, the Logical ID does not provide a Default Interrupt Handler.

Adding, Patching, Extending, and Replacing

ABIOS provides a mechanism to add, patch, extend and replace the system board ROM or adapter ROM ABIOS using an adapter ROM as well as using RAM. Definitions for adding, patching, extending, and replacing ABIOS, followed by the mechanisms for accomplishing each, are shown below.

Adding This adds a previously unsupported ABIOS 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 ABIOS support. Adding involves a new or old interface, new ABIOS, and new hardware.

Patching This revectors an existing ABIOS function to a patched routine. Patching involves an existing interface, new ABIOS, and existing hardware.

Extending This adds a previously unsupported function to a particular ABIOS interface that operates on the same device and uses the same Device Block. Extending involves a new interface, new ABIOS, 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 possibly new interface, new ABIOS, 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 ABIOS

Adapter ROM Structure

ABIOS provides a facility to integrate adapters with on-board ROM code into the system. During ABIOS initialization, the absolute addresses hex C0000 through DF800 are scanned in 2K blocks in search of a valid adapter ROM.

Adapters that support ROMs can participate in the following convention.

Field	**Offset**	**Length**
Signature=AA55H (Word Value)	+00H	2
Length in 512-Byte Blocks	+02H	1
BIOS Initialization Entry Point	+03H	3
Signature = BB66H (Word Value)	+06H	2
Number of Initialization Table Entries	+08H	1
Build Initialization Table Entry Point	+09H	-

Figure 5-2. ROM Module Header ABIOS

The ROM Module Header entries are described in more detail below.

Signature = AA55H (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 associated with the ROM module header.

BIOS Initialization Entry Point: This field is the location in the ROM which is called by the power-on self-test (POST).

Signature = BB66H (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 adapter's ROM that the Build Initialization Table BIOS function (INT 15H, (AH) = 05H, see Figure 3-4 on page 3-6) calls to build the Initialization Table entry for the adapter.

The ABIOS structure is very similar to the BIOS structure and does not preclude the support of existing adapters using 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 the power-on self-test (POST).

When the operating system invokes the Build System Parameters Table BIOS function (INT 15H, (AH) = 04H, see Figure 3-2 on page 3-4) 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 required for the system board ROM.

When the operating system invokes the Build Initialization Table BIOS function (INT 15H, (AH) = 05H, see Figure 3-4 on page 3-6) a ROM scan is invoked, searching the ROM address space in 2K increments until a valid ABIOS ROM is detected. The Build Initialization Table Entry routine is called for each valid ROM to fill in the Initialization Table for devices 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 of the corresponding ROM module header.

RAM Extension Structure

ABIOS provides a facility to integrate adapters with RAM-loadable code into the system. It is the responsibility of the operating system to load the RAM extensions from permanent media to RAM before ABIOS initialization. After ABIOS initialization, RAM extensions may be relocated, but they are always required to be in memory. During ABIOS initialization, when the Build System Parameters Table BIOS function (see Figure 3-2 on page 3-4) and the Build Initialization Table BIOS function (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.

Field	Offset	Length
Signature = AA55H (Word Value)	+00H	2
Length in 512-Byte Blocks	+02H	1
Model Byte	+03H	1
Submodel Byte	+04H	1
ROM Revision Level	+05H	1
Device ID	+06H	2
Number of Initialization Table Entries	+08H	1
Build Initialization Table Entry Point	+09H	3
Secondary Device ID	+0CH	1
Revision	+0DH	1
Reserved	+0EH	2

Figure 5-3. RAM Extension Header

The RAM Extension Header entries are:

Signature = AA55H (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 7H) of the RAM extension associated with the RAM extension header.

Model Byte, Submodel Byte, ROM Revision Level: These fields describe the system board ROM to which the RAM extension is associated.

Device ID, Secondary Device ID, Revision: These fields describe the ABIOS service to 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 is the location in the RAM extension that the Build Initialization Table BIOS function 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 linked by way of the Length in 512-Byte Blocks 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 in 512-Byte Blocks field is n, meaning the extension was n/2 KB in length, the header for RAM extension 1 is at location (XXXX + (20H * n)):0000. The last RAM extension in the RAM extension area points to a RAM extension with the Length in 512-Byte Block field set to 0.

When the operating system invokes the Build System Parameters Table BIOS function (INT 15H, (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 required for the system board and adapter ROMs.

When the operating system invokes the Build Initialization Table BIOS function (INT 15H, (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) 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 used by adapter ROMs and RAM extensions.

BUILD INITIALIZATION TABLE ENTRY ROUTINE

```
Invocation:  Call FAR, ABIOS 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 Completed Successfully
                   ≠ 00H, Indicates no entries added
                   = 80H, No units found

             (CX)  = Count of entries added to the Initialization Table
                   = 0 if AL ≠ 0

             (All registers except AX, CX, and Flags are restored)
```

Figure 5-4. Build Initialization Table Entry Routine

Adding

To add a previously unsupported ABIOS 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. Once the Initialization Table is 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 additional ABIOS interface.

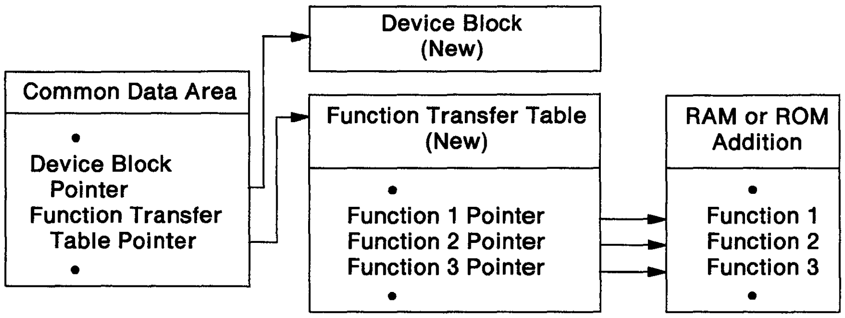


Figure 5-5. Adding ABIOS

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. To patch an ABIOS service, the new Initialization Table entry is built the same as the old Initialization Table entry with the following exceptions:

- The Device Block Length field is set to 0, indicating the existing Device Block suffices for the adapter ROM or RAM extension. Thus, the Device Block Pointer field within the Common Data Area associated with this Initialization Table entry should be set to 0:0 by the operating system.

- Likewise, the Function Transfer Table Length field is set to 0, indicating the existing Function Transfer Table suffices for the adapter ROM or RAM extension. Thus, the Function Transfer Table Pointer field in the Common Data Area associated with this Initialization Table entry should be set to 0:0 by the operating system.
- The Number of Logical IDs field is set to 1, indicating 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, the Secondary Device ID, and the Revision fields in the Device Block of the service to be patched. The search is accomplished by accessing the Device Block Pointer field associated with each Logical ID and interrogating the public portion of the Device Block containing the Device ID, the Secondary Device ID, and the Revision fields, until the Logical ID (entry in the Common Data Area) to be patched is found (see the ABIOS device block in Figure 2-6 on page 2-11). When scanning the Common Data Area, 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 corresponding to the Starting Logical ID parameter passed to the Initialize Device Block and Function Transfer Table routine are already set to 0:0, indicating 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.

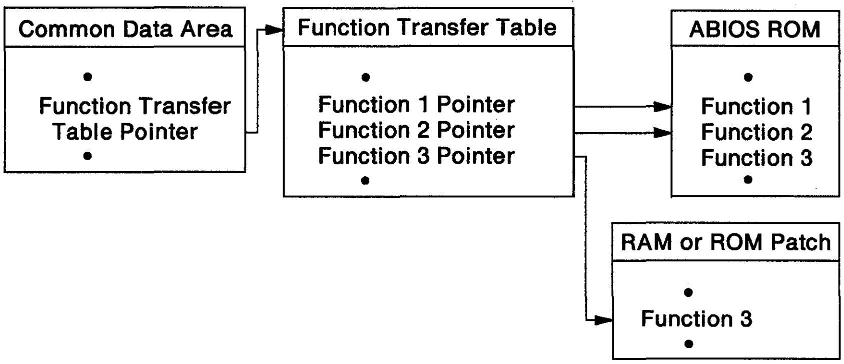


Figure 5-6. Patching ABIOS

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. To extend an ABIOS service, the new Initialization Table entry is built the same as the old Initialization Table with the following exceptions:

- The Device Block Length field is set to 0, indicating the existing Device Block suffices for the adapter ROM or RAM extension. Thus, the Common Data Area Device Block Pointer field associated with this Initialization Table entry should be set to 0:0.
- The Function Transfer Table Length field is set to the value of the old Function Transfer Table Length field plus the length of the extensions.
- The Count of Logical IDs field is set to 1, indicating 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, the Secondary Device ID, and the Revision fields in the device block of the service to be extended. The search is accomplished by accessing the Device Block Pointer field associated with each Logical ID and interrogating the public portion of the Device Block containing the Device ID, the Secondary Device ID, and the Revision fields, until the Logical ID (entry in the Common Data Area) to be extended is found (see the Device Block in Figure 2-6 on page 2-11). When the Common Data Area is scanned, any null entries should be disregarded. The old function pointers for the service 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 new Function Transfer Table Function Count field is updated to reflect the count of the old functions plus the count of the 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 corresponding to the Starting Logical ID parameter that is passed to the Initialize Device Block and Function Transfer Table routine is already set to 0:0. The Initialize Device Block and Function Transfer Table routine must set the associated Function Transfer Table Pointer field to 0:0, indicating a null Common Data Area entry.

The following diagram shows the effect of extending an ABIOS interface.

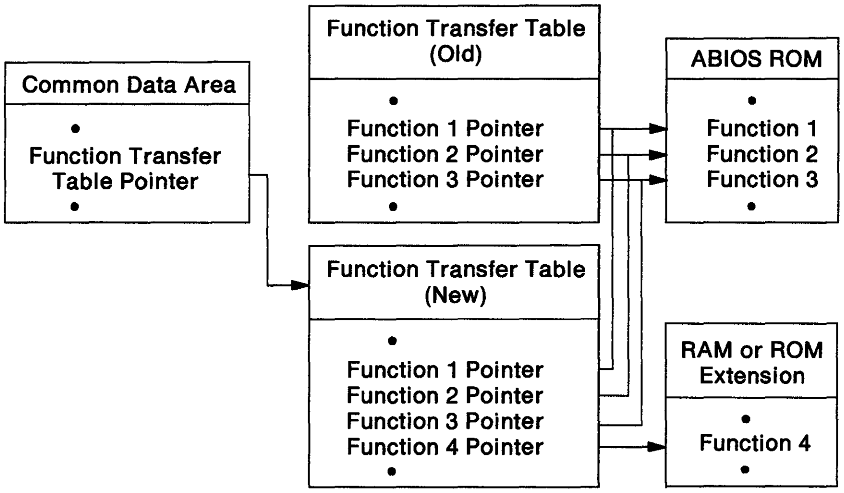


Figure 5-7. Extending ABIOS

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 new values in each of the fields 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, the Secondary Device ID, and the Revision fields in the Device Block of the service to be replaced. The search is accomplished by accessing the Device Block Pointer field associated with each Logical ID and interrogating the public portion of the Device Block containing the Device ID, the Secondary Device ID, and the Revision fields, until the Logical ID (entry in the Common Data Area) to be replaced is found (see the Device Block in Figure 2-6 on page 2-11). When 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 corresponding to the Starting Logical ID parameter. The Function Transfer Table Pointer field in the Common Data Area that previously pointed to the old Function Transfer Table is replaced with the pointer to the new Function Transfer Table. Next the Device Block is built for the Starting Logical ID parameter. Once the Device Block is built, the Device Block Pointer field in the Common Data Area that previously pointed to the old Device Block, is replaced with 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 corresponding to the Starting Logical ID parameter to 0:0 indicating a null Common Data Area entry.

The following diagram shows the effect of replacing an ABIOS interface.

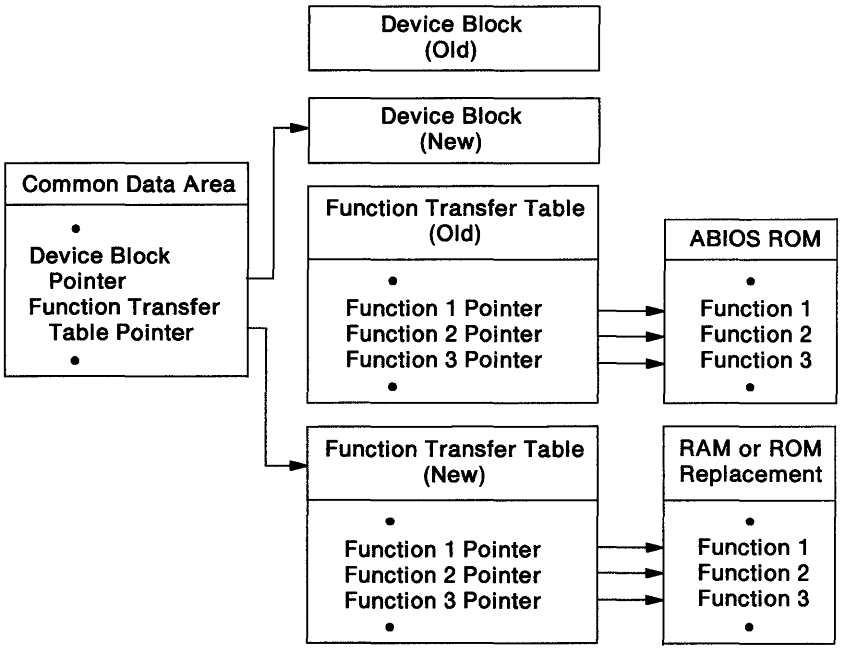


Figure 5-8. Replacing ABIOS

Considerations for RAM Extensions

The Model byte, Submodel byte and the ROM Revision Level fields within the RAM extension header are called system board identifiers because they describe the system board ROM associated with the RAM extension. The Device ID, Secondary Device ID, and Revision fields are called service identifiers because they describe the specific ABIOS service that is associated with the RAM extension.

Two tests determine if the RAM extension is required for a particular system.

1. The first test determines if the RAM extension should remain resident in memory and must be performed at RAM extension load time. This test determines if the system board identifiers in the RAM extension match the system board identifiers that are returned by Return System Configuration function (INT 15H, (AH) = C0H). If the system board identifiers match, the RAM extension remains resident in memory for ABIOS initialization. If the RAM extension header has the system board identifier combination that indicates that it is a system board identifier wild card (Model Byte equals 0, Submodel Byte equals 0, and ROM Revision Level equals 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 but ensures the test is accurate for all RAM extension headers in the file.

2. The second test determines if the service identifiers in the RAM extension header match a service that exists in the system board ROM or an adapter ROM. This test is performed by the Initialize Device Block and Function Transfer Table routine. If the 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 is 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, it must have the service identifiers in the RAM extension header set to the service identifier wild card (Device ID equals hex 00FF, Secondary Device ID equals hex FF, and Revision equals hex FF).

Each new version of a particular ABIOS service that patches, extends or replaces an existing version must have at least one of the service identifiers different in the new 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, since a new Device Block is not built, the Revision field in the existing Device Block is updated and the Device ID and Secondary Device ID fields remain the same. For replacing, since a new Device Block is built due to the addition of hardware, the Device Block is built with the same Device ID field, but the value of the Secondary Device ID field is increased by 1 and the value of the Revision field reverts to 0. For adding, no test is made on an existing Device Block, therefore the 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 and/or new lines.
- The files associated with the filespecs in ABIOS.SYS (as well as ABIOS.SYS itself) are assumed to reside on the root directory of the IPL volume.
- If the system identifier test passes, the files associated with the filespecs in ABIOS.SYS are loaded into memory and appended to one another in the order that 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/2 is a trademark of International Business Machines Corporation.

Operating Systems Implementation Considerations

ABIOS Rules

The following rules are presented for programmers writing operating systems and device drivers.

Rule 1	The Device Block Pointer field, the Function Transfer Table Pointer field and all Data Pointer fields for a given Logical ID within the ABIOS Common Data Area must not be altered by the operating system during a particular stage of a request to that Logical ID.
Rule 2	ABIOS, after being interrupted within a given stage of a request, returns to that stage in the mode that it was running at the time of interrupt.
Rule 3	ABIOS Device Blocks are owned by ABIOS and only the public portions are accessed by the operating system. There is no guarantee of compatibility of the Device Block private area contents across ABIOS implementations.
Rule 4	ABIOS Request Blocks are shared by ABIOS and the operating system.
Rule 5	ABIOS must traverse the Common Data Area to retrieve necessary pointers. It does not store pointers in one request or stage of a request to be used on another request or stage of a request.
Rule 6	ABIOS function X, running in protected mode, 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 mode X, ABIOS can be called through the START routine in mode Y.
Rule 7	ABIOS does not change the state of the interrupt flag. ABIOS can temporarily disable the interrupt flag, but restores it 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 within memory it is assumed that an external process is performing the read or write to memory, therefore across stages this address cannot change.

Rule 10 The effective memory address of a logical address pointer (those pointers in the Request Block in the form segment:offset or selector:offset) can be changed or moved across stages of a request. In real mode, the segment and/or offset can be changed. In protected mode, the selector and/or offset can be changed as well as the physical address located in the descriptor.

Rule 11 ABIOS does not do end of interrupt processing. In a level-sensitive interrupt environment, the device condition causing the interrupt is reset by ABIOS.

Rule 12 The caller of ABIOS can perform EOI processing when the Return Code field is any value other than Not My Interrupt, Stage On Interrupt (hex 0005) and all Request Blocks are serviced on the Logical ID. The caller can assume that the interrupt was serviced and process the EOI.

Rule 13 The caller of ABIOS must call each outstanding request per Logical ID at interrupt time until the first Logical ID with an interrupting condition is completely processed. Exceptions are defined in Section 6, "Interfaces."

Rule 14 Operating System device numbers are allocated by the operating system based on increasing units within increasing Logical IDs. For example; the first Logical ID with Device ID = printer, unit 0 is lpt1:, unit 1 is lpt2:. If unit 1 does not exist, the second Logical ID with Device ID = printer, unit 0 is named lpt2:, and so on.

Rule 15 ABIOS in a protected mode or bimodal implementation must have I/O privilege when operating in protected mode.

Considerations for Bimodal Implementations

ABIOS is written to be independent of the mode of the microprocessor. Because segmented address pointers have different meanings in the two modes, and memory above 1MB is not generally addressable in real mode, an operating system with a bimodal implementation must conform to the following requirements:

Addressability of Tables: The operating system is to ensure that the Request Blocks, Device Blocks, Function Transfer Tables, and the Common Data Area are addressable at all times to ABIOS in the mode in which it is called.

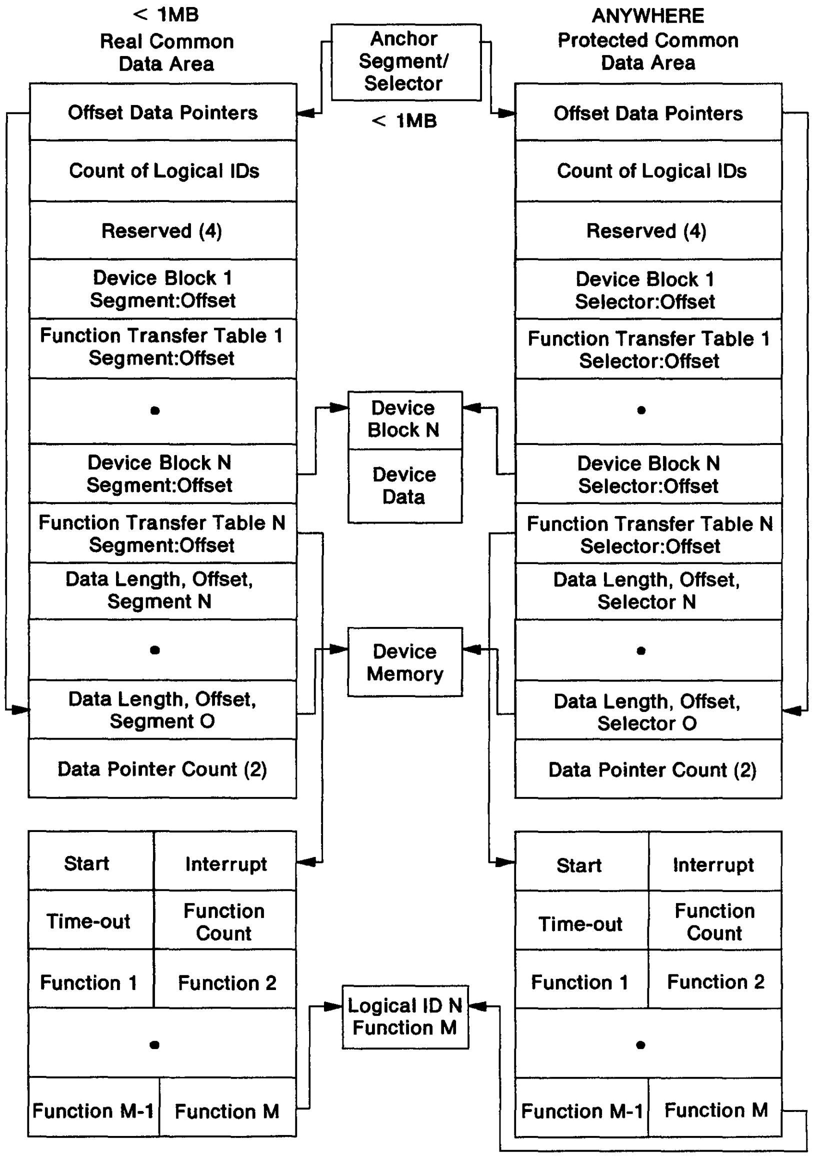
Addressability of Data for Programmed I/O: Non-DMA devices cannot readily use memory above 1MB in real mode. The operating system should allocate the I/O buffers for these devices below 1MB if ABIOS is called in real mode.

Reentrantcy and Mode Change: ABIOS operating in one mode can be interrupted and invoked in the other mode. The Interrupt and the Time-out routines are fully reentrant. The Start routines are reentrant with respect to the Device Block. That is, ABIOS can support multiple requests to common code operating on different Device Blocks at the same time within different stack frames. If the Start routine cannot begin a request, it sets a Return Code field of Device In Use, Request Refused (hex 8000).

Two Copies of Tables Recommended: Since segmented memory pointers have ambiguous meaning in a bimodal environment, the operating system should keep a real mode and a protected mode version of the Common Data Area and Function Transfer Tables to avoid the overhead of converting all of the pointers in the tables after switching modes. One restriction is that the offset fields in the Function Transfer Table must be the same for the corresponding entry in both tables. When there are two copies of the table, a 0:0 value in the Data Pointer field in the real mode Common Data Area indicates that the address is above 1MB.

ABIOS is oblivious to the fact there is more than one table. ABIOS does not require that the protected mode table be initialized before calling ABIOS in real mode. But, the protected mode table must be built before calling ABIOS in protected mode.

The following figure shows a view of the ABIOS Common Data Area, Function Transfer Tables, and Device Blocks in a bimodal environment.



< 1MB
Real Common Data Area
Anchor Segment/ Selector
< 1MB
ANYWHERE
Protected Common Data Area
Offset Data Pointers
Count of Logical IDs
Reserved (4)
Device Block 1 Segment:Offset
Function Transfer Table 1 Segment:Offset
•
Device Block N Segment:Offset
Function Transfer Table N Segment:Offset
Data Length, Offset, Segment N
•
Data Length, Offset, Segment O
Data Pointer Count (2)
Offset Data Pointers
Count of Logical IDs
Reserved (4)
Device Block 1 Selector:Offset
Function Transfer Table 1 Selector:Offset
•
Device Block N Selector:Offset
Function Transfer Table N Selector:Offset
Data Length, Offset, Selector N
•
Data Length, Offset, Selector O
Data Pointer Count (2)
Device Block N
Device Data
Device Memory
Start
Interrupt
Time-out
Function Count
Function 1
Function 2
•
Function M-1
Function M
Logical ID N Function M

Figure 5-9. Bimodal Data Areas

Description of Fields of Bimodal Common Data Area

The following are descriptions of the fields shown in Figure 5-9 on page 5-23.

Anchor Segment/Selector: Referred to as the Anchor pointer, this is a word segment or selector with an assumed 0 offset pointing to the Common Data Area. There is no requirement that the segment value passed in real mode equals the selector value passed in protected mode.

Real Common Data Area: This is the Common Data Area used by ABIOS operating in the real mode.

Protected Common Data Area: This is the Common Data Area used by ABIOS operating in protected mode.

Offset Data Pointers: This entry is an offset, which in conjunction with the Anchor pointer, points to the ABIOS 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 count 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 ABIOS Device Block n.

Start: This is a doubleword pointer to the Start routine for this Logical ID available to the caller using the Operating System Transfer Convention.

Interrupt: This is a doubleword pointer to the Interrupt routine for this Logical ID, and is available to the caller using the Operating System Transfer Convention.

Time-out: This is a doubleword pointer to the Time-out routine for this Logical ID, and is available to the caller using the Operating System Transfer Convention.

Function Count: This is a count of functions supported for this Logical ID.

Function M: This is a doubleword pointer to the Mth Function routine for this Logical ID.

Notes:

Section 6. Interfaces

Introduction

This section describes the interfaces supported by ABIOS. Each interface description includes the interface functions and the values of the Return Code field. Programming considerations are also included where appropriate.

Parameters are passed to ABIOS functions in the request block. All input parameters are set by the caller and all output parameters are returned by ABIOS functions.

This section describes only the service specific parameters. The functional parameters are described in Figure 4-2 on page 4-6.

The following notes apply to each ABIOS device interface in this section:

- The Default Interrupt Handler function (hex 00) and the Return Logical ID Parameters function (hex 01) are described on page 4-7.
- For the functions Read (hex 08), Write (hex 09) and Additional Data Transfer (hex 0A), the Data Pointer Mode (physical vs. logical) should be determined through the Return Logical ID Parameters function (hex 01).
- All input fields marked as reserved must be initialized to 0 by the caller of ABIOS.
- All input fields are unaltered by ABIOS 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. These fields must not be altered by the caller across the stages of a request.

- The following values of the Return Code field are returned for parameter errors although they are not indicated as possible Return Code field values within each function description:
  - Hex C000 - Invalid Logical ID, ABIOS Transfer Convention only
  - Hex C001 - Invalid Function Number
  - Hex C003 - Invalid Unit Number
  - Hex C004 - Invalid Request Block Length.
- The value of the Return Code field Device in Use, Request Refused (hex 8000) 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 this device.
- The caller should generically handle the error ranges of the Return Code field as defined for the Request Block (see Section 4, "Transfer Conventions"). 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 (in microseconds) is returned when the Return Code field is set to Wait On Time (hex 0002).

Diskette

Functions

The following are the diskette functions. The Default Interrupt Handler function and the Return Logical ID Parameters function are described in “Request Block” on page 4-3.

00H - Default Interrupt Handler

01H - Return Logical ID Parameters

02H - Reserved

03H - Read Device Parameters

- This function returns the default parameters used in diskette operations and device control information.
- 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 based upon the Number of Tracks to Format and the Number of Sectors per Track fields passed on the Set Media Type for Format (hex 0D) function. If this bit is 0, the user must provide the Gap Length for Format parameter for the media that is being formatted.
- The possible value of the Return Code field is equal to 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 supported by the drive
Word	+12H	Size of sector in bytes
		00H - Reserved
		01H - 256 bytes/sector
		02H - 512 bytes/sector
		03H to FFFFH - Reserved
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 - ABIOS defines Gap Length for Format parameter based on the Number of Tracks to Format and Number of Sectors per Track fields as input to the Set Media Type for Format function (hex 0D)
		Bits 5, 4 - Reserved
		Bit 3 - Recalibrate status
		0 - Recalibrate is not required
		1 - Recalibrate is required
		Bit 2 - Concurrent operations
		0 - Not supported
		1 - Supported
		Bit 1 - Format unit information
		0 - Format unit is not supported
		1 - Format unit is supported
		Bit 0 - Change signal availability
		0 - Change signal is not available
		1 - Change signal is available
Word	+16H	Diskette drive type
		00H - Drive not present/invalid NVRAM
		01H - 5.25 inch, 40-Track, 2-Head, 360KB
		02H - Reserved
		03H - Reserved
		04H - 3.5 inch, 80-Track, 2-Head, 1.44MB
		05H to FFFFH - Reserved
DWord	+1CH	Delay before turning off motor (in microseconds)
DWord	+20H	Motor startup time (in microseconds)
Word	+26H	Number of cylinders on the maximum media density supported by the drive
Byte	+2AH	Number of heads
Byte	+2BH	Recommended software retry count
Byte	+2CH	Fill byte for format
DWord	+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.
- If the media was formatted with a sector size other than the default sector size of 512-bytes per sector, this function should be issued once before executing the functions Read (hex 08), Write (hex 09), Verify (hex 0B), or Format (hex 0A) to set the correct sector size.
- ABIOS uses the 512-bytes per sector value until this function is called to set a different sector size.
- The possible values of the Return Code field are equal to hex 0000, 8000, and C005.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	+10H	Reserved
Word	+12H	Sector size in bytes 00H - Reserved 01H - 256 bytes per sector 02H - 512 bytes per sector 03H to FFFFH - Reserved
Byte	+31H	Gap length for read/write/verify
Byte	+33H	Data length

Service Specific Output

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.
- The possible values of the Return Code field are equal to 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 equal to 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 number on the diskette to the specified memory location.
- If the diskette was formatted with a sector size other than the default sector size of 512-bytes per sector, the Set Device Parameters function (hex 04) should be issued to set the proper sector size before issuing this function.
- 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 Media Changed (hex 8006). However, if the 'diskette change' signal is active and ABIOS is unable to reset the 'diskette change' signal to the inactive state, the Return Code field is set to Media Not Present (hex 800D) and no data is transferred.
- If the Number of Sectors to Read field is 0, no action is performed and the Return Code field is set to Operation Completed Successfully (hex 0000).
- The possible values of the Return Code field are equal to hex 0000, 0001, 0002, 8000, 8006, 800D, 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 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 read

09H - Write

- This function transfers data from the specified memory location to the diskette at the specified cylinder, head, and sector number.
- If the diskette was formatted with a different sector size other than the default sector size of 512-bytes per sector, the Set Device Parameters function (hex 04) should be issued to set the proper sector size before issuing this function.
- 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 Media Changed (hex 8006). However, if the ‘diskette change’ signal is active and ABIOS is unable to reset the ‘diskette change’ signal to the inactive state, the Return Code field is set to Media Not Present (hex 800D) and no data is transferred.
- If the Number of Sectors to Write field is 0, no action is performed, and the Return Code field is set to Operation Completed Successfully (hex 0000).
- The possible values of the Return Code field are equal to hex 0000, 0001, 0002, 8000, 8003, 8006, 800D, 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 write
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 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.
- The Set Media Type For Format function (hex 0D) must be issued once before issuing this function 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 Media Changed (hex 8006) or Media Not Present (hex 800D).
- 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 Media Changed (hex 8006). However, if the 'diskette change' signal is active and ABIOS is unable to reset the 'diskette change' signal to the inactive state, the Return Code field is set to Media Not Present (hex 800D) and the field ID is not written.

0AH - Additional Data Transfer (Subfunction 00H - Format) (continued)

- The possible values of the Return Code field are equal to 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 data is not transferred.
- If the diskette was formatted with a different sector size other than the default sector size of 512-bytes per sector, the Set Device Parameters function (hex 04) should be issued to set the proper sector size before issuing this function.
- 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 Media Changed (hex 8006). However, if the 'diskette change' signal is active and ABIOS is unable to reset the 'diskette change' signal to the inactive state, the Return Code field is set to Media Not Present (hex 800D).
- If the Number of Sectors to Verify field is 0, no action is performed and the Return Code field is set to Operation Completed Successfully (hex 0000).
- The possible values of the Return Code field are equal to 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
Word	+1EH	Reserved
Word	+24H	Number of sectors to verify
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 verified

0CH - Read Media Parameters

- This function returns the media parameters used for the previous operation.
- Because multiple media types may be supported for a single drive type, the functions Read (hex 08), Write (hex 09), Verify (hex 0B), or Format (hex 0A) should be issued prior to this function to ensure the proper media parameter values are returned.
- The possible values of the Return Code field are equal to hex 0000 and 8000 only.

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 - 256 bytes per sector 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 to be used for the format operation based on the given information of number of tracks and number of sectors per track.
- Media present is checked.
  - If the diskette has been removed or the drive door is left open, ABIOS sets the Return Code field to Media Not Present (hex 800D) 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 Format field and the Number of Sectors per Track field are valid for the supported diskette drive types, ABIOS sets the correct parameters as requested. Otherwise, a Return Code value of Unsupported Media Type/Unestablished Media (hex C00C) is returned.
- 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 this function. If bit 6 is set to 1, the Gap Length for Format parameter is determined based upon the Number of Tracks to Format and the Number of Sectors per Track fields passed to this function. If bit 6 is 0, the user must provide the Gap Length for Format parameter for the media that is being formatted.
- This function must be invoked once before issuing the Format function (hex 0B) to insure the proper diskette format information.
- ABIOS uses these parameters until they are changed by issuing the Set Device Parameters function (hex 04), or until the drive door is opened.
- The caller is responsible for restoring the sector size to its initial value if it was changed by issuing the Set Device Parameters function (hex 04).
- The possible values of the Return Code field are equal to 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 - 256 bytes per sector 02H - 512 bytes per sector 03H to FFFFH - Reserved
Word	+16H	Reserved
Word	+26H	Number of tracks to format
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 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 Change Signal Status field is valid only when the specified drive supports the 'diskette change' signal. The 'diskette change' signal availability is returned in the Read Device Parameters function (hex 03).
- The active status indicates that the diskette has been changed or the diskette drive door is open and the diskette type information is invalid. Data is not transferred when the 'diskette change' signal is active.
- The possible values of the Return Code field are equal to hex 0000, 8000, and 800E.

Service Specific Input

```
SIZE    OFFSET      DESCRIPTION

Word    +16H        Reserved
```

Service Specific Output

```
SIZE    OFFSET      DESCRIPTION

Byte    +10H        Change signal status
                      00H - Change signal inactive
                      01H to 05H - Reserved
                      06H - Change signal active
                      07H to FFH - Reserved
```

0FH - Turn Off Motor

- This function turns the diskette drive motor off for the requested drive.
- The motor may be turned off by the caller when the Return Code field is set to Operation Completed Successfully (hex 0000).
- This function is required for the functions Reset (hex 05), Read (hex 08), Write (hex 09), Additional Data Transfer (hex 0A), Verify (hex 0B), Read Media Parameters (hex 0C), Set Media Type For Format (hex 0D), and Read Change Signal (hex 0E).
- The delay before turning off the motor is returned in Read Device Parameters function (hex 03).
- The possible values of the Return Code field are equal to 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 equal to 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 01H - Interrupt pending

Return Codes

Value	Description
0000H	Operation Completed Successfully
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
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 Failed
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 if an unsuccessful operation needs to be retried. The recommended retry count is returned in the Read Device Parameters function (hex 03).

- The motor may be turned off by the caller when the Return Code field equals Operation Completed Successfully (hex 0000) by using the Turn Off Motor function (hex 0F). This function is required for the functions Reset (hex 05), Read (hex 08), Write (hex 09), Additional Data Transfer (hex 0A), Verify (hex 0B), Read Media Parameters (hex 0C), Set Media Type for Format (hex 0D), or Read Change Signal Status (hex 0E). The caller is responsible for turning off the motor when the request is completed by using the Turn Motor Off function (hex 0F).

- Diskette ABIOS supports crossing track boundaries, but only switching from head 0 to head 1 on the same cylinder. It does not support switching from head 1 of a cylinder to head 0 of the next cylinder.
- When issuing Diskette ABIOS and Diskette BIOS requests, the following rules should 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 following a BIOS request.
  - The Reset Diskette System BIOS function (INT 13H, (AH) = 00H) must be the first BIOS request following an ABIOS request. Also, set bit 4 to 0 in BIOS Data Area hex 40:90 for drive A and hex 40:91 for drive B before issuing any diskette function.
  - The Reset/Initialize function (hex 05) must be issued after ABIOS initialization has been completed.
- In the event of an error, ABIOS resets the diskette system.
- A request for the function Read (hex 08), Write (hex 09), Verify (hex 0B), Additional Data Transfer (hex 0A), or Set Media Type for Format (hex 0D) resets the 'diskette change' signal to the inactive state before proceeding with the requested function.

Notes:

Disk

Functions

The following are the disk functions. The Default Interrupt Handler function and the Return Logical ID Parameters function are described in "Request Block" on page 4-3.

00H - Default Interrupt Handler

01H - Return Logical ID Parameters

02H - Reserved

03H - Read Device Parameters

- This function returns disk drive information based on the unit requested and the disk device control information.
- The possible value of the Return Code field is equal to hex 0000.

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
Word    14H     Device control flags
                Bits 15 to 13 - Reserved
                Bits 12, 11   - Format support (values in binary)
                                00 - Format not supported
                                01 - Format track supported
                                10 - Format unit supported
                                11 - Format track and format unit supported
                Bit 10 - ST506 Drive
                         0 - Not ST506
                         1 - ST506
                Bit 9 - Concurrent unit requests per Logical ID
                        0 - Not concurrent
                        1 - Concurrent
```

Service Specific Output (continued)

SIZE	OFFSET	DESCRIPTION
		Bit 8 - Ejecting capability 0 - Not ejectable 1 - Ejectable Bit 7 - Media organization 0 - Random 1 - Sequential Bit 6 - Locking capability 0 - Not lockable 1 - Lockable Bit 5 - Read capability 0 - Not readable 1 - Readable Bit 4 - Caching support 0 - No caching 1 - Caching Bit 3 - Write frequency 0 - Write once 1 - Write many Bit 2 - Change signal support 0 - No change signal supported 1 - Change signal supported Bits 1, 0 - Reserved
DWord	18H	Physical number of cylinders associated with unit in Request Block
Byte	1CH	Physical number of heads associated with unit in Request Block
Byte	1DH	Suggested number of software retries for retryable operations
DWord	20H	Physical number of relative block addresses associated with unit in Request Block
DWord	24H	Reserved
Word	28H	Reserved
Word	2CH	Maximum number of blocks to transfer 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 listed in the Disk Return Codes table are possible for this function.

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 Read field contains the amount of data to transfer.
- If the Number of Blocks to Read field is 0, no action is performed.
- If the Number of Blocks to Read field is greater than the maximum number of blocks, then no action is performed, and the Return Code field is set to Invalid Parameter (hex C005).
- The Number of Blocks Read field contains the amount of data transferred.
- When a parameter error is returned, the Number of Blocks Read field is not updated. Also, when a hex 8000 or 800F error is returned, the Number of Blocks Read field is not updated.
- All Return Code values listed in the Disk Return Codes table are possible for this function.

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 read
Byte	2EH	
	Bits 7 to 1 - Reserved (set to 0)	
	Bit 0 - Caching	
		0 - Caching is OK for this request
		1 - Don't 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	Indicates if a soft error occurred
		= 0 - Soft error did not occur
		≠ 0 - 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 Write field contains the amount of data to transfer.
- If the Number of Blocks to Write field is 0, no action is performed.
- If the Number of Blocks to Write field is greater than the maximum number of blocks, no action is performed, and the Return Code field is set to Invalid Parameter (hex C005).
- The Number of Blocks Written field contains the amount of data transferred.
- When a parameter error is returned, the Number of Blocks Written field is not updated. Also, when a hex 8000 or 800F error is returned, the Number of Blocks Written field is not updated.
- All Return Code values listed in the Disk Return Codes table are possible for this function.

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 write
Byte    2EH
        Bits 7 to 1 - Reserved (set to 0)
        Bit 0 - Caching
                0 - Caching is OK for this request
                1 - Don't 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     Indicates if a soft error occurred
                = 0 - Soft error did not occur
                ≠ 0 - Soft error occurred
```

0AH - Write Verify

- The Write Verify function operates similar to the Write function (hex 09) with the addition of a Verify function (hex 0B).
- If the Number of Blocks to Write/Verify field is 0, no action is performed.
- If the Number of Blocks to Write/Verify field is greater than the maximum number of blocks, no action is performed, and the Return Code field is set to Invalid Parameter (hex C005).
- The Number of Blocks Written field contains the amount of data transferred.
- When a parameter error is returned, the Number of Blocks Written field is not updated. Also, when a hex 8000 or 800F error is returned, the Number of Blocks Written field is not updated.
- All Return Code values listed in the Disk Return Codes table are possible for this function.

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 write/verify
Byte     2EH
         Bits 7 to 1 - Reserved (set to 0)
         Bit 0 - Caching
                 0 - Caching is OK for this request
                 1 - Don't 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     Indicates if a soft error occurred
                 = 0 - Soft error did not occur
                 ≠ 0 - 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 Verify field is 0, no action is performed.
- If the Number of Blocks to Verify field is greater than the maximum number of blocks, no action is performed, and the Return Code field is set to Invalid Parameter (hex C005).
- All Return Code values listed in the Disk Return Codes table are possible for this function.

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 verify

Service Specific Output

SIZE	OFFSET	DESCRIPTION
DWord	28H	Time to wait before resuming request in microseconds
Word	2FH	Indicates if a soft error occurred = 0 - Soft error did not occur ≠ 0 - Soft error occurred

0CH - Interrupt Status

- This function returns the Disk Interrupt Pending status. It does not reset the interrupt condition.
- The Interrupt Status field is associated with the Logical ID as opposed to the Unit field. This field represents the current interrupt pending state of the disk controller. The Unit field is tested for validity.
- If there is a parameter error, the Interrupt Status field is undefined.
- The possible value of the Return Code field is equal to hex 0000.

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
```

Return Codes

Value	Description
0000H	Operation Completed Successfully
0001H	Stage on Interrupt
0002H	Stage on Time
0005H	Not My Interrupt, Stage on Interrupt
8000H	Device Busy, Request Refused
800FH	DMA Arbitration Level Out of Range
9001H	Bad Command
9002H	Address Mark Not Found
9004H	Record Not Found
9005H	Reset Failed
9007H	Controller Parameter Activity Failed
900AH	Defective Sector
900BH	Bad Track
900DH	Invalid Sector on Format
900EH	CAM Detected During Read or Verify
9010H	Uncorrectable ECC or CRC Error
9020H	Bad Controller
9021H	Equipment Check
9040H	Bad Seek
9080H	Device Did Not Respond
90AAH	Drive Not Ready
90BBH	Undefined Error
90CCH	Write Fault
90FFH	Incomplete Sense Operation
9101H	Bad Command
9105H	Reset Failed
9107H	Controller Parameter Activity Failed
9120H	Bad Controller
9121H	Equipment Check
9140H	Bad Seek
9180H	Device Did Not Respond
91AAH	Drive Not Ready
91BBH	Undefined Error
91CCH	Write Fault
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

Figure 6-2 (Part 1 of 2). Disk Return Codes

Value	Description
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
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 Parameter

Figure 6-2 (Part 2 of 2). Disk Return Codes

Programming Considerations

- The Disk ABIOS interface requires the use of the DMA ABIOS interface, therefore, if the disk ABIOS is initialized and used, the DMA ABIOS must be initialized.
- Read Device Parameters returns the number of software retries to attempt for any one operation when an error occurs that is retryable.
- In the event of an error, ABIOS resets the disk system when required.
- For the functions Read (hex 08), Write (hex 09), and Write/Verify (hex 0A), the output parameter at hex 2C represents the number of blocks transferred as determined from the hardware. This value is supplied in the event that the request ended in error before the data transfer was complete. If no error is reported, this value equals the number of blocks that were requested for transfer. It is only valid when the request is completed.
- When error recovery procedures are invoked by the adapter and are successful, disk ABIOS attempts to determine the nature of the recovery performed. It sets the Soft Error Occurred field in the Request Block with the recovered error code.
- Relative block addresses begin ordering with the first block assigned the value 0. For hardware devices that do not support relative block addresses, the equivalent is Cylinder 0, Head 0, and Sector 1. In the formulas below, 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). Disk ABIOS returns physical values for number of sectors per track, number of heads, and number of cylinders on the Read Device Parameters function (hex 03), which should be used for relative block address calculations. ABIOS uses the following to break down the relative block address (RBA):

Sector ID = (RBA MOD Sectors Per Track) + 1

Head = (RBA / Sectors Per Track) MOD Heads

Cylinder = (RBA / Sectors Per Track) / Heads

The RBA may be calculated by the following:

RBA = (Sectors Per Track * Heads * Cylinder) +
(Sectors Per Track * Head) + (Sector ID - 1)

The number of RBAs is:

RBAs = Cylinders * Heads * Sectors Per Track

This is the value returned by Read Device Parameters.

The maximum allowable RBA is:

Maximum RBA = (Cylinders * Heads * Sectors Per Track) - 1.

- When issuing Disk ABIOS and Disk BIOS requests, 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.

Video

Functions

The following are the video functions. The Default Interrupt Handler function and the Return Logical ID Parameters function are described in "Request Block" on page 4-3.

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 used to generate alpha characters when bit 3 of the Attribute byte is a 1. The Character Block Select B field specifies the block used to generate alpha characters when bit 3 of the Attribute byte is a 0. When the Character Block Select A field is equal to the Character Block Select B field, the Character Select function is disabled and bit 3 of the Attribute byte determines the foreground intensity state (1 = On, 0 = Off).
- The Save/Restore Header Size, Hardware State Size, Device Block State Size, and DAC State Size fields 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 6-38 for more information.
- The possible value of the Return Code field is equal to hex 0000.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	+28H	Reserved

03H - Read Device Parameters (continued)

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 6-4 on page 6-48)
Word	+20H	Type of monitor attached Bits 15 to 1 - Reserved Bit 0 - Color vs monochrome 0 - Color monitor 1 - Monochrome monitor
Word	+22H	Character height (bytes/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 required for the Return ROM Fonts function.
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 DAC state in bytes

04H - Set Device Parameters (Reserved)

05H - Reset/Initialize

- This function initializes the video controller to the requested mode (see Figure 6-4 on page 6-48).
- The Character Blocks to Load field tells which character blocks will be loaded with the default ROM character font for the requested mode and scan lines. This parameter is only required when setting an alpha mode (hex 0, 1, 2, 3, or 7).
- Scan lines are only specified when setting an alpha mode (hex 0, 1, 2, 3, or 7).
- The Character Block Specifier field is only specified when setting an alpha mode (hex 0, 1, 2, 3, or 7).
- For the Character Block Specifier field, the Character Block Select A field specifies the block used to generate alpha characters when bit 3 of the Attribute byte is a 1. The Character Block Select B field specifies the block used to generate alpha characters when bit 3 of the Attribute byte is 0. When the Character Block Select A field is equal to the Character Block Select B field, the Character Select function is disabled and bit 3 of the Attribute byte determines the foreground intensity state (1 = On, 0 = Off).
- The Summing bit of the Device Control Flags field is only required when a color display is attached. Summing is done automatically for monochrome displays.
- When using a monochrome display in a color mode, the colors are displayed as shades of gray. There are 16 of 64 gray shades available in all modes except mode hex 13, where 64 gray shades are available.
- Modes hex 0, 2, and 4 are identical to modes hex 1, 3, and 5 respectively.
- The possible value of the Return Code field is equal to hex 0000.

05H - Reset/Initialize (continued)

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 - Don't clear buffer
                        1 - Clear buffer
Byte   +1CH     Requested number of scan lines
                  00H - 200 scan lines (modes 0, 1, 2, 3)
                  01H - 350 scan lines (modes 0, 1, 2, 3, 7)
                  02H - 400 scan lines (modes 0, 1, 2, 3, 7)
                  03H to FFH - Reserved
Word   +1EH     Video mode to set
                  (see Figure 6-4 on page 6-48)
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 load with default ROM font
                Bit 'n' - Block 'n' flag
                          0 - Don't 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 size of character (row and column), whether it is a total or partial font, and if a partial font, which font it relates to.
- There are 12 bytes of information per ROM font. They are stored sequentially in the specified data area.
- The following shows the format of a ROM font entry:

```
Word  - Reserved
DWord - Pointer to ROM font
Word  - Reserved
Byte  - Size of character (number of columns)
Byte  - Size of character (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 using this function, the Read Device Parameters function (hex 03) should be issued to find the size of the buffer required to save the ROM fonts information.
- The possible value of the Return Code field is equal to 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 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) * (size of hardware state)
  C = Environment (bit 1) * (size of device block state)
  D = Environment (bit 2) * (size of digital-to-analog converter state).
- The possible value of the Return Code field is equal to 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
                Bits 15 to 3 - Reserved (set to 0)
                Bit 2 - DAC state (1 = save state)
                Bit 1 - Device block state (1 = save state)
                Bit 0 - Hardware state (1 = save state)
```

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 on the contents and structure of the video environment.
- Unexpected results may occur if you restore a particular state not previously saved.
- The possible value of the Return Code field is equal to 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
                Bits 15 to 1 - Reserved
                Bit 0 - Regenerative buffer flag
                        0 - Don't clear buffer
                        1 - Clear buffer
Word   +2CH     Video environment states to be restored
                Bits 15 to 3 - Reserved (set to 0)
                Bit 2 - DAC state (1 = restore state)
                Bit 1 - Device block state (1 = restore state)
                Bit 0 - Hardware state (1 = restore state)
```

Service Specific Output

```
SIZE   OFFSET   DESCRIPTION

None
```

0EH - Select Character Generator Block

- This function selects up to two character generator blocks.
- For the Character Block Specifier field, the Character Block Select A field specifies the block used to generate alpha characters when bit 3 of the Attribute byte is a 1. The Character Block Select B field specifies the block used to generate alpha characters when bit 3 of the Attribute byte is a 0. When the Character Block Select A field is equal to the Character Block Select B field, the character select function is disabled and bit 3 of the Attribute byte determines the foreground intensity state (1 = On, 0 = Off).
- The possible value of the Return Code field is equal to 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 - Alpha Load

- This function loads the requested character generator or part of one to the specified character blocks.
- This function does not update the hardware registers. Refer to the Enhanced Alpha Load function (hex 10) if hardware updating is required.
- When loading any of the ROM character generators (the Character Generator Type field is equal to 1, 2, or 3), the full set of characters (hex 100) is loaded. Thus, the only parameters required to invoke this function are the Character Generator Type field and the Character Block Specifier field.

- When loading a user font (the Character Generator Type field is equal to 0) all parameters are required.
- When loading a user font, if the Count of Characters field is equal to 0, no character is loaded and the Return Code field is set to Operation Completed Successfully (hex 0000).
- When loading a user font, the sum of the Count of Characters field and the Character Offset field should not exceed the maximum number of characters in a set (hex 100). If it does, the Return Code field is set to Invalid Video Parameter (hex C005).
- The possible values of the Return Code field are equal to 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
                  1 to 100H  - Valid count of characters
Byte   +1DH     Character generator type
                  00H - User's alphanumerics font
                  01H - 8 x 8 alphanumerics ROM font
                  02H - 8 x 14 alphanumerics ROM font
                  03H - 8 x 16 alphanumerics ROM font
                  04H to FFH - Reserved
Word   +22H     Character height (bytes/character)
Word   +24H     Character block to load
                  00H to 07H - Valid character blocks to load values
                  08H to FFFFH - Reserved
Word   +28H     Character offset into the table
```

Service Specific Output

```
SIZE   OFFSET   DESCRIPTION

None
```

10H - Enhanced Alpha Load

- This function loads the requested character generator or part of one to the specified character block and updates the hardware registers.
- When loading any of the ROM character generators (the Character Generator Type field is equal to 1, 2, or 3), the full set of characters (hex 100) are loaded. Thus, the only parameters

required to invoke this function are the Character Generator Type field and the Character Blocks to Load field.

- When loading a user font (Character Generator Type field is equal to 0) all parameters are required.
- When loading a user font, if Count of Characters field is equal to 0, no character is loaded and the Return Code field is set to Operation Completed Successfully (hex 0000).
- When loading a user font, the sum of the Count of Characters field and the Character Offset field should not exceed the maximum number of characters in a set (hex 100). If it does the Return Code field is set to Invalid Video Parameter (hex C005).
- The possible values of the Return Code field are equal to 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
                  1 to 100H - Valid count of characters
Byte   +1DH     Character generator type
                  00H - User's alphanumerics font
                  01H - 8 x  8 alphanumerics ROM font
                  02H - 8 x 14 alphanumerics ROM font
                  03H - 8 x 16 alphanumerics ROM font
                  04H to FFH - Reserved
Word   +22H     Character height (bytes/character)
Word   +24H     Character block to load
                  00H to 07H - Valid character blocks to load values
                  08H 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 possible value of the Return Code field is equal to hex 0000.

Service Specific Input

```
SIZE   OFFSET   DESCRIPTION

Word   +16H     Reserved
Word   +32H     Palette register to read
                  00H to 0FH - Valid palette register to read values
                  10H to FFFFH - Reserved
```

Service Specific Output

```
SIZE   OFFSET   DESCRIPTION

Word   +34H     Palette value read.
```

12H - Write Palette Register

- This function writes a value to a palette register.
- Executing this function when the mode is set to mode hex 13 is not allowed. It is a hardware requirement to have these registers remain programmed as set by the Reset/Initialize function (hex 05). Changing these registers can cause unpredictable results.
- The possible value of the Return Code field is equal to hex 0000.

Service Specific Input

```
SIZE   OFFSET   DESCRIPTION

Word   +16H     Reserved
Word   +32H     Palette register to write
                  00H to 0FH - Valid palette register to write values
                  10H to FFFFH - Reserved
Word   +34H     Palette value to load
                  00H to 3FH - Valid
                  40H 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 possible value of the Return Code field is equal to hex 0000.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	+16H	Reserved
Word	+2AH	Color register to read 00H to FFH - Valid color register to read values 100H to FFFFH - Reserved

Service Specific Output

SIZE	OFFSET	DESCRIPTION
Word	+2CH	Red value read
Word	+2EH	Green value read
Word	+30H	Blue value read

14H - Write Color Register

- This function loads a digital-to-analog converter color register with the specified red, green and blue values.
- For the Device Control Flags field, the Summing bit is disregarded when a monochrome display is attached. Summing always occurs with a monochrome display when in color modes.
- The possible value of the Return Code field is equal to hex 0000.

Service Specific Input

```
SIZE   OFFSET   DESCRIPTION

Word   +16H     Reserved
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     Color register to write
                  00H to FFH - Valid color registers to write values
                  100H to FFFFH - Reserved
Word   +2CH     Red value to write
                  00H to 3FH - Valid red value to write
                  40H to FFFFH - Reserved
Word   +2EH     Green value to write
                  00H to 3FH - Valid green value to write
                  40H - FFFFH - Reserved
Word   +30H     Blue value to write
                  00H to 3FH - Valid blue value to write
                  40H - FFFFH - Reserved
```

Service Specific Output

```
SIZE   OFFSET   DESCRIPTION

None
```

15H - Read Block of Color Registers

- This function reads a block of digital-to-analog color registers into the specified save area beginning at the requested color register.
- The format of the data 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, or blue values is hex 00 to 3F.
- If the Count of Color Registers to Read field equals 0, no action is performed and the Return Code field is set to Operation Completed Successfully (hex 0000).
- If the value of the First Color Register to Read field plus the value of the Count of Color Registers to Read field is greater than the maximum number of color registers, no action is performed and the Return Code field is set to Invalid Video Parameter (hex C005).
- The possible values of the Return Code field are equal to 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     Count of color registers to read
Word   +2AH     First color register to read
                  00H to FFH - Valid first color register to read values
                  100H 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 to be written is (red value, green value, blue value), (red value, green value, blue value), ..., (red value, green value, blue value).
- If the value of the Count of Color Registers to Write field equals 0, no action is performed and the Return Code field is set to Operation Completed Successfully (hex 0000).
- If the value of the First Color Register to Write field plus the value of the Count of Color Registers field is greater than the maximum number of color registers, no action is performed and the Return Code field is set to Invalid Video Parameter (hex C005).
- For the Device Control Flags field, the Summing bit is disregarded when a monochrome display is attached. Summing will always occur with a monochrome display when in color modes.
- The possible values of the Return Code field are equal to 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 write
Word   +1AH     Device control flags
                Bits 15 to 3 - Reserved
                Bit 2        - Summing
                               0 - Summing disable
                               1 - Summing enabled
                Bits 1, 0    - Reserved
Word   +2AH     First color register to write
                  00H to FFH - Valid first color register to write values
                  100H to FFFFH - Reserved
```

Service Specific Output

```
SIZE  OFFSET    DESCRIPTION

None
```

Return Codes

Value	Description
0000H	Operation Completed Successfully
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 6-3. Video Return Codes

Video Modes

The following figure shows the supported Video Modes.

Mode (Hex)	Type	Max Colors	Alpha Format	Buffer Start	Box Size	Max Pages	Display Pel Dimensions
00	A/N	16	40x25	B8000	8x8	8	320x200
00	A/N	16	40x25	B8000	8x14	8	320x350
00	A/N	16	40x25	B8000	9x16	8	360x400
01	A/N	16	40x25	B8000	8x8	8	320x200
01	A/N	16	40x25	B8000	8x14	8	320x350
01	A/N	16	40x25	B8000	9x16	8	360x400
02	A/N	16	80x25	B8000	8x8	8	640x200
02	A/N	16	80x25	B8000	8x14	8	640x350
02	A/N	16	80x25	B8000	9x16	8	720x400
03	A/N	16	80x25	B8000	8x8	8	640x200
03	A/N	16	80x25	B8000	8x14	8	640x350
03	A/N	16	80x25	B8000	9x16	8	720x400
04	APA	4	40x25	B8000	8x8	1	320x200
05	APA	4	40x25	B8000	8x8	1	320x200
06	APA	2	80x25	B8000	8x8	1	640x200
07	A/N	Mono	80x25	B0000	9x14	8	720x350
07	A/N	Mono	80x25	B0000	9x16	8	720x400
08 - 0C	Reserved						
0D	APA	16	40x25	A0000	8x8	8	320x350
0E	APA	16	80x25	A0000	8x8	4	640x200
0F	APA	Mono	80x25	A0000	8x14	2	640x350
10	APA	16	80x25	A0000	8x14	2	640x350
11	APA	2	80x30	A0000	8x16	1	640x480
12	APA	16	80x30	A0000	8x16	1	640x480
13	APA	256	40x25	A0000	8x8	1	320x200

Figure 6-4. Video Modes Table

Keyboard

Functions

The following are the keyboard functions. The Default Interrupt Handler function and the Return Logical ID Parameters function are described in "Request Block" on page 4-3.

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 equal to 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 equal to 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 to allow keyboard data to be passed 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

07H - Disable

- This function disables the keyboard by inhibiting 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 the keyboard scan code. It must be called soon after ABIOS initialization to allow for the processing of Keystroke interrupts. Once 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 with the Return Code field set to Attention, Stage on Interrupt (hex 0009). This Return Code value indicates that there is a valid scan code in the Keyboard Row Scan Code field.
- The possible values of the Return Code field are equal to 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 equal to 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 equal to 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 equal to 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 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 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 equal to 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 equal to 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 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 0, no action is performed and the Return Code field is set to Operation Completed Successfully (hex 0000).
- The possible values of the Return Code field are equal to 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 0, no action is performed and the Return Code field is set to Operation Completed Successfully (hex 0000).
- The possible values of the Return Code field are equal to 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

Value	Description
0000H	Operation Completed Successfully
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 (ABIOS Transfer Convention only)
C001H	Invalid Function
C003H	Invalid Unit Number
C004H	Invalid Request Block Length
FFFFH	Return Code is Not Valid.

Figure 6-5. Keyboard Return Codes

Programming Considerations

- The Keyboard ABIOS does not attempt any retries. It is up to the calling program to perform any retries.
- The Write Keyboard Data String function (hex 11) sends bytes to the keyboard and expects an acknowledgement (ACK) after each byte has been sent.
- The Write Keyboard Controller Data String function (hex 10) sends bytes to the keyboard controller, and 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 Write Indicators function (hex 0C). If the Write Keyboard Data String function (hex 11) is used to change the indicators, the value returned by the Read Indicators function (hex 0B) may not reflect the true state at the keyboard.
- The Keyboard Time-out routine does not attempt to reset the keyboard, rather it sets the Return Code field to Keyboard Error (hex B001 or B101). It is up to the caller to execute the Keyboard Reset/Initialize function (hex 05).
- If Keyboard ABIOS is expecting an acknowledge from the keyboard after issuing a command to the keyboard, it does not pass the acknowledge to the controlling program.

Notes:

Parallel Port

Functions

The following are the parallel port functions. The Default Interrupt Handler function and the Return Logical ID Parameters function are described in “Request Block” on page 4-3.

00H - Default Interrupt Handler

01H - Return Logical ID Parameters

02H - Reserved

03H - Read Device Parameters

- This function returns the device specific information.
- The Printer Initialize Time to Wait Before Resuming Request field contains the wait time value returned on the Reset/Initialize function (hex 05).
- The possible value of the Return Code field is equal to hex 0000.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	16H	Reserved

Service Specific Output

SIZE	OFFSET	DESCRIPTION
DWord	20H	Printer Initialize 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 Interrupt Time-out field must be a nonzero value. If it is set to 0, ABIOS sets the Return Code field to Invalid Time-out (hex C005).
- The Printer Initialize Time to Wait Before Resuming Request field contains the wait time value returned on the Reset/Initialize function (hex 05).
- The Printer Initialize Time to Wait Before Resuming Request field must be a nonzero value. If it is set to 0, ABIOS sets the Return Code field to Invalid Time to Wait (hex C006).
- ABIOS uses these parameters until this function is called to change them.
- The possible values of the Return Code field are equal to hex 0000, 8000, C005, and C006.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	16H	Reserved
DWord	20H	Printer Initialize 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 returned in the Request Block is not valid during intermediate stages.
- The possible values of the Return Code field are equal to hex 0000, 0002, and 8000.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	16H	Reserved

Service Specific Output

SIZE	OFFSET	DESCRIPTION
DWords	20H	Time to wait before resuming request in microseconds
Byte	28H	Printer status Bit 7 - Busy Bit 6 - Acknowledge Bit 5 - End of paper Bit 4 - Selected Bit 3 - I/O error Bit 2 - Interrupt 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 caller must use the Cancel Print Block function (hex 0B) to cancel an existing Print Block function before issuing another Print Block function to the same unit.
- The Printer Status field is valid only when the function is completed. The status returned in the Request Block is not valid during intermediate stages.
- When a print error occurs, the printer is offline, or the printer is busy, ABIOS terminates the print block request. The Number of Characters Printed field indicates the portion of the print block that has been printed. The unprinted portion of the print block can be printed by issuing another print block request when the terminating condition is corrected.
- The possible values of the Return Code field are equal to hex 0000, 0001, 0005, 8000, 8001, and 9000.

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 characters to print

Service Specific Output

SIZE	OFFSET	DESCRIPTION
Word	26H	Number of characters printed
Byte	28H	Printer status Bit 7 - Busy Bit 6 - Acknowledge Bit 5 - End of paper Bit 4 - Selected Bit 3 - I/O error Bit 2 - Interrupt Bits 1, 0 - Reserved

0AH - Additional Data Transfer (Reserved)

0BH - Cancel Print Block

- This function cancels an outstanding Print Block function (hex 09) request.
- The Printer Status field is valid only when the function has completed. The status returned in the Request Block is not valid during intermediate stages.
- The possible value of the Return Code field is equal to hex 0000.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	16H	Reserved

Service Specific Output

SIZE	OFFSET	DESCRIPTION
Word	26H	Number of characters printed
Byte	28H	Printer status Bit 7 - Busy Bit 6 - Acknowledge Bit 5 - End of paper Bit 4 - Selected Bit 3 - I/O Error Bit 2 - Interrupt 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 completed successfully.
- The possible values of the Return Code field are equal to 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 - Busy
                      Bit 6 - Acknowledge
                      Bit 5 - End of paper
                      Bit 4 - Selected
                      Bit 3 - I/O error
                      Bit 2 - Interrupt
                      Bits 1, 0 - Reserved
```

Return Codes

Value	Description
0000H	Operation Completed Successfully
0001H	Stage on Interrupt
0002H	Stage on Time
0005H	Not My Interrupt, Stage on Interrupt
8000H	Device in Use
8001H	Device Busy
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

Figure 6-6. Parallel Port Return Codes

Programming Considerations

- For the Print Block function (hex 09), if the Printer Status field is busy, ABIOS checks the printer status for some time, and the function is terminated if the device is still busy after that time. ABIOS sets the Return Code field to Device Busy (hex 8001). The Number of Characters Printed field indicates to the caller the number of characters that were printed.
- For the Print Block function (hex 09), if the printer is put off-line in the middle of a print block, ABIOS checks the printer status for some time, and the function is terminated if the device is still busy after that time. ABIOS sets the Return Code field to Device Busy (hex 8001). The caller can issue a new Print Block function to print the remaining characters when the printer is put back online.
- When the Reset/Initialize function is called, some printers perform a printer 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.
- The parallel port ABIOS supports the parallel port in transmit mode only.

Notes:

Asynchronous Communications

Functions

The following are the asynchronous communications functions. The Default Interrupt Handler function and the Return Logical ID Parameters function are described in "Request Block" on page 4-3.

00H - Default Interrupt Handler

01H - Return Logical ID Parameters

02H - Reserved

03H - Read Device Parameters

- This function returns the communication port information. It has no affect 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 ABIOS. If the asynchronous ports are programmed directly or through the BIOS Interrupt Hex 14 functions, the asynchronous parameters in the Device Block will be incorrect. In this event the Read Device Parameters function returns the previously stored values.
- To synchronize these parameters, a Reset/Initialize function (hex 05) must be issued.
- During ABIOS initialization, the serial communication port is initialized to the following default parameters: baud rate is 1200, no parity, 1 stop bit, 7 bits per character and no break.
- The possible value of the Return Code field is equal to hex 0000.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	18H	Reserved

Service Specific Output

SIZE	OFFSET	DESCRIPTION
Byte	28H	Modem control
		Bits 7 to 2 - Reserved (set to 0)
		Bit 1 - Request to send
		0 - Disable
		1 - Enable
		Bit 0 - Data terminal ready
		0 - Disable
		1 - Enable
Byte	29H	Async interrupt status byte
		Bits 7 to 6 - Reserved (set to 0)
		Bit 5 - Modem status interrupt
		0 - Disabled
		1 - Enabled
		Bit 4 - Reserved
		Bit 3 - Transmit interrupt
		0 - Disabled
		1 - Enabled
		Bit 2 - Receive interrupt
		0 - Disabled
		1 - Enabled
		Bit 1, 0 - Reserved
Byte	2AH	Receive trigger level
		00H - 1 byte
		01H - 4 bytes
		02H - 8 bytes
		03H - 14 bytes
		04H to 0FFFFH - Reserved
Byte	2BH	FIFO mode status
		Bits 7 to 4 - Reserved
		Bit 3 - FIFO support available
		0 - Not present
		1 - Present
		Bit 2 - Reserved
		Bit 1 - FIFO device
		0 - Not present
		1 - Present
		Bit 0 - FIFO Indicator
		0 - Disabled
		1 - Enabled

Service Specific Output (continued)

```
SIZE    OFFSET  DESCRIPTION

Byte     44H    Transmission baud rate
                  00H - 110
                  01H - 150
                  02H - 300
                  03H - 600
                  04H - 1200
                  05H - 2400
                  06H - 4800
                  07H - 9600
                  08H - 19200
                  09H to 0FFH - Reserved
Byte     45H    Type of parity
                  00H - None
                  01H - Odd
                  02H - Even
                  03H - Stick parity odd
                  04H - Stick parity even
                  05H to 0FFH - Reserved
Byte     46H    Number of stop bits
                  00H - 1
                  01H - 2
                       (If the number of bits per character field is 1,2, or 3)
                      - 1 1/2 stop bits
                       (If the number of bits per character field is 0)
                  02H to 0FFH - Reserved
Byte     47H    Number of bits per character
                  00H - 5
                  01H - 6
                  02H - 7
                  03H - 8
                  04H to 0FFH - Reserved
Byte     48H    Break status
                  00H - Disabled
                  01H - Enabled
                  02H to 0FFH - Reserved
```

04H - Set Device Parameters (Reserved)

05H - Reset/Initialize

- This function initializes the asynchronous communications port according to the input parameters.
- All communication interrupts (receive, transmit, and modem status) are disabled. It is up to the caller to clean up all outstanding Request Blocks and the interrupt controller where appropriate. From the ABIOS standpoint all outstanding Request Blocks are canceled.

- Any receive data that is pending at the communication 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 possible value of the Return Code field is equal to hex 0000.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	18H	Reserved
Byte	28H	Modem control
		Bit 7 to 2 - Reserved (set to 0)
		Bit 1 - Request to send
		0 - Disable
		1 - Enable
		Bit 0 - Data terminal ready
		0 - Disable
		1 - Enable
Byte	29H	Reserved
Byte	2AH	Receive trigger level
		00H - 1 byte
		01H - 4 bytes
		02H - 8 bytes
		03H - 14 bytes
		04H to 0FFFFH - Reserved
Byte	2BH	FIFO mode control
		00H - Disable
		01H - Enable and reset FIFO
		02H - Enable without resetting FIFO
		03H to 0FFH - Reserved
Byte	44H	Transmission baud rate
		00H - 110
		01H - 150
		02H - 300
		03H - 600
		04H - 1200
		05H - 2400
		06H - 4800
		07H - 9600
		08H - 19200
		09H to 0FFH - Reserved
Byte	45H	Type of parity
		00H - None
		01H - Odd
		02H - Even
		03H - Stick parity odd
		04H - Stick parity even
		05H to 0FFH - Reserved

Service Specific Input (continued)

SIZE	OFFSET	DESCRIPTION
Byte	46H	Number of stop bits
		00H - 1
		01H - 2 - For 6-bit, 7-bit, or 8-bit word length
		- 1 1/2 - For 5-bit word length
		02H to 0FFH - Reserved
Byte	47H	Number of bits per character
		00H - 5
		01H - 6
		02H - 7
		03H - 8
		04H to 0FFH - Reserved
Byte	48H	Break status
		00H - Disabled
		01H - Enabled
		02H to 0FFH - Reserved

Service Specific Output

SIZE	OFFSET	DESCRIPTION
Byte	49H	Line status
		Bit 7 - Error in receiver FIFO
		0 - Disable
		1 - Enable
		Bit 6 - Transmitter empty
		0 - Disable
		1 - Enable
		Bit 5 - Transmitter holding register empty
		0 - Disable
		1 - Enable
		Bit 4 - Break interrupt
		0 - Disable
		1 - Enable
		Bit 3 - Framing error
		0 - Disable
		1 - Enable
		Bit 2 - Parity error
		0 - Disable
		1 - Enable
		Bit 1 - Overrun error
		0 - Disable
		1 - Enable
		Bit 0 - Data ready
		0 - Disable
		1 - Enable

05H - Reset/Initialize (continued)

Service Specific Output (continued)

SIZE	OFFSET	DESCRIPTION
Byte	4AH	Modem status Bit 7 - Data carrier detect 0 - Disable 1 - Enable Bit 6 - Ring indicator 0 - Disable 1 - Enable Bit 5 - Data set ready 0 - Disable 1 - Enable Bit 4 - Clear to send 0 - Disable 1 - Enable Bit 3 - Delta data carrier detect 0 - Disable 1 - Enable Bit 2 - Trailing edge ring indicator 0 - Disable 1 - Enable Bit 1 - Delta data set ready 0 - Disable 1 - Enable Bit 0 - Delta clear to send 0 - Disable 1 - Enable

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 parameter. This function does not affect the interrupt state of any other Stage on Interrupt requests.
- The possible value of the Return Code field is equal to hex 0000.

Service Specific Input

```
SIZE    OFFSET  DESCRIPTION

Word     18H    Reserved
Byte     28H    Modem control
                Bits 7 to 2 - Reserved (set to 0)
                Bit 1 - Request to send
                        0 - Disable
                        1 - Enable
                Bit 0 - Data terminal ready
                        0 - Disable
                        1 - Enable
```

Service Specific Output

```
SIZE    OFFSET  DESCRIPTION

None
```

0CH - Set Line Control

- This function sets the line control according to the input parameters. This function does not affect the interrupt state of any other Stage on Interrupt requests.
- The possible value of the Return Code field is equal to 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 0FFH - Reserved
Byte     46H    Number of stop bits
                  00H - 1
                  01H - 2 - For 6-bit, 7-bit, or 8-bit word length
                      - 1 1/2 - For 5-bit word length
                  02H to 0FFH - Reserved
Byte     47H    Number of bits per character
                  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 0FFH - Reserved
```

Service Specific Output

```
SIZE    OFFSET  DESCRIPTION

None
```

0DH - Set Baud Rate

- This function sets the baud rate according to the input parameter, and does not affect the interrupt state of any other Stage on Interrupt requests.
- The possible value of the Return Code field is equal to hex 0000.

Service Specific Input

```
SIZE    OFFSET  DESCRIPTION

Word     18H    Reserved
Byte     44H    Transmission baud rate
                  00H - 110
                  01H - 150
                  02H - 300
                  03H - 600
                  04H - 1200
                  05H - 2400
                  06H - 4800
                  07H - 9600
                  08H - 19200
                  09H to 0FFH - Reserved
```

Service Specific Output

```
SIZE    OFFSET  DESCRIPTION

None
```

0EH - Transmit Interrupt

- This function enables the transmit interrupt but it does not transmit any data until a transmit interrupt occurs. The Transmit Tail Pointer field points to the first byte to be transmitted and the Transmit Head Pointer field points to 1 byte logically beyond the last byte to be sent.
- The values of the Transmit Head and Transmit Tail Pointer fields are relative to the beginning of the transmit buffer where a value of 0 indicates the first physical byte of the buffer and a value of the Transmit Buffer Length field minus 1 indicates the last physical byte of the buffer. Values of the Transmit Head Pointer field and Transmit Tail Pointer field can never be out of that range.
- The maximum number of characters that a single view of the transmit buffer can indicate to be transmitted is 1 less than the value of the Transmit Buffer Length field.

- When a transmit interrupt occurs, the Interrupt routine increments the value of the Transmit Tail Pointer field according to the number of bytes transmitted. The Operation Status field will have the Transmit in Progress bit (bit 1) set. Because of the possibility of the buffer being checked asynchronously to the transmit Interrupt routine, the data is written to the hardware Transmit buffer before the value of the Transmit Tail Pointer field being incremented.
- The value of the Transmit Tail Pointer field chases the value of the Transmit Head Pointer field as interrupts occur. If the value of the Transmit Tail Pointer field reaches the end of the transmit buffer, the Transmit Tail Pointer field is wrapped back 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 the transmit buffer-empty condition occurs after sending data to the communication port, ABIOS abruptly stops sending data to the communication port and informs the caller of the condition. The Operation Status field has the Transmit Buffer-Empty bit (bit 6) set indicating that the transmit buffer is empty, but the transmit interrupt is still enabled. On the following transmit interrupt or any transmit interrupt, if the buffer-empty condition exists, ABIOS disables the transmit interrupt. If the transmit interrupt is disabled due to a buffer-empty condition the Request Block is considered canceled. The Operation Status field has the Transmit Buffer-Empty, Transmitter Holding Register Empty bit (bit 7) set. The Transmit Buffer-Empty bit (bit 6) and the Transmit Buffer-Empty, Transmitter Holding Register Empty bit (bit 7) are mutually and exclusively set bits.
- A transmit buffer-full condition occurs when the value of the Transmit Head Pointer field is one less than the value of the Transmit Tail Pointer field. Also, when the value of the Transmit Tail Pointer field equals 0, and the value of the Transmit Head Pointer field equals 1 less than the value of the Transmit Buffer Length field, the transmit buffer is full.
- The values of the Transmit Buffer Segment field, the Transmit Buffer Offset field, and Transmit Buffer Length field can be altered across calls to the Transmit Interrupt routine. Because ABIOS removes the data from the buffer before changing the Transmit Tail Pointer field, the caller may add data to be transmitted during the processing of a transmit interrupt by

placing the data in the buffer and then logically incrementing the value of the Transmit Head Pointer field. The caller must not allow the value of the Transmit Head Pointer field to be equal to the value of the Transmit Tail Pointer field because this indicates an empty transmit buffer.

- The Operation Status field contains the current operation status (the Transmit in Progress or Transmit buffer empty). The Return Code field has the status as described by the ABIOS structure. The Operation Status field should be initialized to 0 by the caller prior to calling the Start routine.
- If the Transmit Buffer Length field is 0 when called through the Start routine, no action is performed and the Return Code field is set to 0.
- When the Reset/Initialize function (hex 05) is executed, if the Number of Bits per Character field that is set is less than 8, the high-order bits of each byte transmitted are undefined. These high-order bits are sent to the communication port without change.
- The possible values of the Return Code field are equal to hex 0000, 0001, 0005, 0081, 8000, and 9000.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	10H	Reserved
Word	12H	Transmit buffer offset
Word	14H	Transmit buffer segment
Word	18H	Reserved
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

Service Specific Output

SIZE	OFFSET	DESCRIPTION
Word	34H	Transmit tail pointer
Word	36H	Reserved
Word	4BH	Operation status. Return only from interrupt routine. (See "Programming Considerations" on page 6-91 for bit definitions)

0FH - Receive Interrupt

- This function enables the receive interrupt. Data is read from the communication port when a receive interrupt is generated.
- 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 a value of the Receive Buffer Length field minus 1 indicates the last physical byte of the buffer.
- The value of the Receive Head Pointer field points to the first character position to be filled by ABIOS. The value of the Receive Tail Pointer field points to the first received character to be removed by the caller.
- The maximum number of characters that a single view of the receive buffer can indicate to be received is 1 less than the value of the Receive Buffer Length field.
- When a receive interrupt is generated, the Interrupt routine increments the value of the Receive Head Pointer field by 1. Upon exit, the Operation Status field has the Receive Interrupt in Progress bit (bit 0) set. Because of the possibility of the buffer being checked asynchronously by the Receive Interrupt routine, the character is written to the receive buffer before the value of the Receive Head Pointer field is incremented.
- As interrupts occur, the value of the Receive Head Pointer field chases 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 wrapped back to 0. A receive buffer-full condition occurs when the value of the Receive Head Pointer field is 1 less than the value of the Receive Tail Pointer field. Also, when the value of the Receive Tail Pointer field equals 0 and the value of the Receive Head Pointer field is 1 less than the value of the Receive Buffer Length field, the receive buffer is considered full. The Operation Status field has the Receive Buffer-Full bit (bit 4) set when the Receive Buffer-Full condition occurs. If the buffer-full condition exists and a receive interrupt occurs, the current byte is lost. The Operation Status field has the Receive Buffer Full With Data Erased bit (bit 5) set. The Receive Buffer Full bit (bit 4) and the Receive Buffer Full With Data Erased bit (bit 5) are mutually and exclusively set bits.

- A Receive Buffer-Empty condition occurs when the value of the Receive Head Pointer field equals the value of the Receive Tail Pointer field. ABIOS never sets the value of the Receive Head Pointer field equal to the value of the Receive Tail Pointer field; however the caller may set the value of the Receive Head Pointer field to the value of the Receive Tail Pointer field as it empties out the receive buffer and logically increments the value of the Receive Tail Pointer field.

- 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 across calls to the Receive Interrupt routine. Because ABIOS places data in the receive buffer and then updates the value of the Receive Head Pointer field, the caller may remove received data from the receive buffer during the processing of a receive interrupt by logically incrementing the value of the Receive Tail Pointer field.

- The Operation Status field has the current operation status (Receive In Progress, Receive Buffer-Full, or Receive In Progress and Parity Error), and the Return Code field has the status as described by the ABIOS structure. The Operation Status field should be initialized to 0 by the caller before calling the Start routine.

- The receive interrupt can only be terminated using a call to the Cancel function.

- When called through the Start routine, if the Receive Buffer Length field is 0, no action is performed and the Return Code field is set to 0.

- If any error conditions occur at the communication port, ABIOS stores the current data byte in the receive buffer and returns immediately to the caller. The Operation Status field has the appropriate error bits (bits 8-10) set accordingly. If a Break interrupt occurs, the data byte is a 0. If an overrun error occurs, the overrun character is lost, and the data byte contains the valid character that caused the overrun. If there is a parity error, the data byte contains the character with the incorrect parity. If there is a framing error, the data byte contains the character that does not have a valid stop bit.

0FH - Receive Interrupt (continued)

- If Null Stripping mode is enabled, a data character of 0 that is received is not stored in the Receive buffer. If an overrun error occurs and a Null Data byte is the byte that generated the error, ABIOS discards the Null Data byte. The Operation Status field indicates that an overrun error was found and the Null Data byte was found and discarded.
- When the Reset/Initialize function (hex 05) is executed, if the Number of Bits per Character field set is less than 8, the high-order bits of each byte are set to 0 when data is received.
- The possible values of the Return Code field are equal to hex 0000, 0001, 0005, 0081, 8000, and 9000.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	18H	Reserved
DWord	1AH	Reserved
Word	20H	Reserved
Word	22H	Receive buffer offset
Word	24H	Receive buffer segment
Byte	28H	Null stripping indicator 00H - Disabled 01H - Enabled 02H to 0FFH - 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

Service Specific Output

SIZE	OFFSET	DESCRIPTION
Word	3CH	Receive head pointer
Word	3EH	Reserved
Word	4BH	Operation status. Return only from interrupt routine. (See "Programming Considerations" on page 6-91 for bit definitions)

10H - Combined (Transmit and Receive) Interrupts

- A combined request is used to initiate a single request to both transmit and receive. Both transmit and receive or receive-only can be initiated. The receive operation can be made by calling the Start routine with only the receive parameters set to enable only the receive interrupt (the Transmit Buffer Length field is set to 0). If this function is called to enable only the receive interrupt, it can be recalled through the Start routine to enable the transmit interrupt with a nonzero value of the Transmit Buffer Length field. When the Start routine is called to enable the receive interrupt, the Return Code field should be set to Return Code Not Valid (hex FFFF). However, when the Start routine is called the second time to enable the transmit interrupt, the Return Code field should not be changed from its previously stored value. In addition, the Return Code field is deemed undefined upon return from the call to the second Start routine (to enable the transmit interrupt) and therefore ignored. The combined interrupt function holds true for only the case described above but not the reverse. That is, the transmit interrupt only cannot be enabled through the Start routine the first time.
- This function treats a transmit interrupt in the same manner as described for the Transmit Interrupt function (hex 0E).
- This function treats a receive interrupt in the same manner as described for the Receive Interrupt function (hex 0F).
- During a receive interrupt, the transmit interrupt is never disabled; no test is made to determine if the Transmit Head Pointer field is equal to the Transmit Tail Pointer field.
- To optimize performance, after servicing a receive interrupt, a transmit interrupt pending condition is checked. If a transmit interrupt is pending, this interrupt is serviced and the Return Code field is set to Attention, Stage on Interrupt (hex 0009). If a second receive interrupt is pending, ABIOS does not service the pending interrupt and returns to the caller.

- If the Return Code field is set to Attention, Stage on Interrupt (hex 0009), the Operation Status of Previous Interrupt field contains the status obtained when the first interrupt was serviced. Likewise, the Return Code of the Previous Interrupt field contains the Return Code value of the first interrupt. The Operation Status field and the Return Code field contain the values for the second interrupt.

- The Operation Status of the Previous Interrupt field and the Return Code of the Previous Interrupt field should be initialized to 0 by the caller before the Start routine is called.

- When called through the Start routine to enable the combined function, if the value of the Receive Buffer Length field equals 0, no action is performed and the Return Code field is set to Operation Completed Successfully (hex 0000).

- When called through the Start routine the first time, if the value of the Transmit Buffer Length field equals 0 and the value of the Receive Buffer Length field is greater than 0, this function acts as a Receive Interrupt function (hex 0F). When called through the Start routine the second time, if the value of the Transmit Buffer Length field equals 0, no action is performed and the Return Code field is set to Stage on Interrupt (hex 0001).

- The Return Code field, Bad Communications Port (hex 9000), indicates a hardware failure.

- The possible values of the Return Code field are equal to hex 0000, 0001, 0005, 0009, 0081, 8000, and 9000.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	10H	Reserved
Word	12H	Transmit buffer offset
Word	14H	Transmit buffer segment
Word	18H	Reserved
DWord	1AH	Reserved
Word	20H	Reserved
Word	22H	Receive buffer offset
Word	24H	Receive buffer segment
Byte	28H	Null stripping indicator 00H - Disabled 01H - Enabled 02H to 0FFH - Reserved
Word	2CH	Transmit buffer length in bytes
Word	2EH	Reserved
Word	30H	Transmit buffer head in bytes
Word	32H	Reserved
Word	34H	Transmit buffer tail in bytes
Word	36H	Reserved
Word	38H	Receive buffer length in bytes
Word	3AH	Reserved
Word	3CH	Receive buffer head in bytes
Word	3EH	Reserved
Word	40H	Receive buffer tail in bytes
Word	42H	Reserved

Service Specific Output

SIZE	OFFSET	DESCRIPTION
Word	34H	Transmit buffer tail in bytes
Word	36H	Reserved
Word	3CH	Receive buffer head in bytes
Word	3EH	Reserved
Word	4BH	Operation status; return only from interrupt routine. (See "Programming Considerations" on page 6-91 for bit definitions)
Word	4DH	Operation Status of previous interrupt
Word	4FH	Return code of previous interrupt

11H - Modem Status Interrupt

- This function enables the modem status interrupt. The Modem Status field is returned to the caller upon exit from the Start and Interrupt routines.
- If the Modem Status Interrupt Request Block is processed before the receive or transmit Request Block at interrupt time, the ABIOS routine detects a change in the modem status even if the higher priority interrupts in the interrupt identification register are still pending. This allows an interrupt handler to detect a change in modem status before receiving or transmitting any data.
- The possible values of the Return Code field are equal to hex 0001, 0005, 0081, and 8000.

Service Specific Input

```
SIZE    OFFSET  DESCRIPTION

Word     18H    Reserved
DWord    1AH    Reserved
```

Service Specific Output

```
SIZE    OFFSET  DESCRIPTION

Byte     4AH    Modem status
                Bit 7 - Data carrier detect
                        0 - Disable
                        1 - Enable
                Bit 6 - Ring indicator
                        0 - Disable
                        1 - Enable
                Bit 5 - Data set ready
                        0 - Disable
                        1 - Enable
                Bit 4 - Clear to send
                        0 - Disable
                        1 - Enable
                Bit 3 - Delta data carrier detect
                        0 - Disable
                        1 - Enable
                Bit 2 - Trailing edge ring indicator
                        0 - Disable
                        1 - Enable
                Bit 1 - Delta data set ready
                        0 - Disable
                        1 - Enable
                Bit 0 - Delta clear to send
                        0 - Disable
                        1 - Enable
```

12H - Cancel

- This function cancels a requested interrupt operation. All or any combination of the interrupts may be canceled. Associated interrupts will be disabled upon return. If an outstanding request is associated with the interrupt type that is canceled, that Request Block is considered canceled and should be appropriately deallocated.
- The exception to this is the combined function. If the Combined (Transmit and Receive) Interrupts function (hex 10) has both the receive and the transmit interrupts enabled and the Cancel function is called to cancel both receive and transmit, the combined Request Block is considered canceled. If the Combined (Transmit and Receive) Interrupts function (hex 10) has both the receive and the transmit interrupts enabled and the Cancel function is called to cancel only one of the transmit and receive interrupts, the Request Block is considered active. If the Combined (Transmit and Receive) Interrupts function (hex 10) has both the receive and the transmit interrupts enabled and cancel is called to cancel transmit, the call through the Start routine to reenable transmit is permitted. If the Combined (Transmit and Receive) Interrupts function (hex 10) has both the receive and the transmit interrupts enabled and the Cancel function is called to cancel receive, to reenable the receive interrupt by way of the Combined (Transmit and Receive) Interrupts function (hex 10), the caller must first cancel the transmit interrupt, then call the Start Routine Combined function with receive enabled. If the Combined (Transmit and Receive) Interrupts function (hex 10) has only the receive interrupt enabled and the Cancel function is called to cancel receive, the combined Request Block is considered canceled.
- The possible value of the Return Code field is equal to hex 0000.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	18H	Reserved
Byte	51H	Interrupt type to cancel Bit 7 - Reserved Bit 6 - Reserved Bit 5 - Modem status interrupt 0 - Do not disable 1 - Disable Bit 4 - Reserved Bit 3 - Transmit interrupt 0 - Do not disable 1 - Disable Bit 2 - Receive interrupt 0 - Do not disable 1 - Disable Bit 1 - Reserved Bit 0 - Reserved

Service Specific Output

SIZE	OFFSET	DESCRIPTION

None

13H - Return Line Status

- This function returns the line status.
- The possible value of the Return Code field is equal to 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 - Disable
                        1 - Enable
                Bit 6 - Transmitter empty
                        0 - Disable
                        1 - Enable
                Bit 5 - Transmitter holding register empty
                        0 - Disable
                        1 - Enable
                Bit 4 - Break interrupt
                        0 - Disable
                        1 - Enable
                Bit 3 - Framing error
                        0 - Disable
                        1 - Enable
                Bit 2 - Parity error
                        0 - Disable
                        1 - Enable
                Bit 1 - Overrun error
                        0 - Disable
                        1 - Enable
                Bit 0 - Data ready
                        0 - Disable
                        1 - Enable
```

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 Device Busy (hex 8000).
- The possible values of the Return Code field are equal to hex 0000 and 8000.

Service Specific Input

```
SIZE    OFFSET  DESCRIPTION

None
```

Service Specific Output

```
SIZE    OFFSET  DESCRIPTION

Byte     4AH    Modem status
                Bit 7 - Data carrier detect
                        0 - Disable
                        1 - Enable
                Bit 6 - Ring indicator
                        0 - Disable
                        1 - Enable
                Bit 5 - Data set ready
                        0 - Disable
                        1 - Enable
                Bit 4 - Clear to send
                        0 - Disable
                        1 - Enable
                Bit 3 - Delta data carrier detect
                        0 - Disable
                        1 - Enable
                Bit 2 - Trailing edge ring indicator
                        0 - Disable
                        1 - Enable
                Bit 1 - Delta data set ready
                        0 - Disable
                        1 - Enable
                Bit 0 - Delta clear to send
                        0 - Disable
                        1 - Enable
```

15H - Used Internally by ABIOS

16H - Set Receive Trigger Level

- This function sets the FIFO trigger level for the asynchronous communications port according to the input parameters.
- The possible value of the Return Code field is equal to hex 0000.

Service Specific Input

```
SIZE    OFFSET  DESCRIPTION

Word     18H    Reserved
Byte     28H    Receive trigger level
                  00H - 1 byte
                  01H - 4 bytes
                  02H - 8 bytes
                  03H - 14 bytes
                  04H to 0FFH - Reserved
```

Return Codes

Value	Description
0000H	Operation Completed Successfully
0001H	Incomplete - Stage on Interrupt
0005H	Incomplete - Not My Interrupt, Stage on Interrupt
0009H	Attention, Stage on Interrupt
0081H	Spurious Interrupt
8000H	Device Busy, Request Refused
9000H	Bad Communications Port
C000H	Invalid Logical ID (ABIOS Transfer Convention only)
C001H	Invalid Function
C003H	Invalid Unit Number
C004H	Invalid Request Block Length

Figure 6-7. Asynchronous Communications Return Codes

Programming Considerations

- For all interrupt operations (transmit, receive, combined, and modem status interrupts), the Operation Status (offset hex 4B) field contains the current communication port status. None of the bits in the Operation Status field takes precedence over any other bits in this field. If the bit is set, the condition associated with the bit is active.

```
Bit 15 to 13 - Reserved
Bit 12 - Overrun error with null data byte found
         0 - Not active
         1 - Active only if null data found and discarded
Bit 11 - Break detected
         0 - Not active
         1 - Active
Bit 10 - Framing error
         0 - Not active
         1 - Active
Bit 9  - Parity error
         0 - Not active
         1 - Active
Bit 8  - Overrun error
         0 - Not active
         1 - Active
Bit 7  - Transmit buffer-empty, transmitter holding register empty
         0 - Not active
         1 - Active
Bit 6  - Transmit buffer-empty
         0 - Not active
         1 - Active
Bit 5  - Receive buffer-full, data discarded
         0 - Not active
         1 - Active
Bit 4  - Receive buffer-full
         0 - Not active
         1 - Active
Bits 3 to 2 - Reserved
Bit 1  - Transmit interrupt in progress
         0 - Not active
         1 - Active
Bit 0  - Receive interrupt in progress
         0 - Not active
         1 - Active
```

- If any errors are in the character received, the data byte is stored in the receive buffer and the Operation Status field indicates the appropriate errors. ABIOS returns to the caller on the first error condition encountered even if more data is left in the hardware buffer.

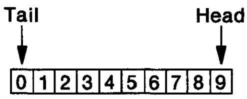
- If the Null Stripping mode is enabled, an overrun error occurs and a Null Data byte is the byte that generated the error, the following takes place. ABIOS discards the Null Data byte, and sets bit 12 of the Operation Status field and returns to the caller.

- ABIOS has no dependence on the values in the BIOS data area except during ABIOS initialization. Asynchronous Communications Logical IDs are ordered by ABIOS in the same sequence as they are ordered in the BIOS data area. The communication port base table for BIOS is located at hex 40:00.

The BIOS Asynchronous Communications device whose port base value is located at hex 40:00 is the first Asynchronous Communications Logical ID to be initialized. The BIOS Asynchronous Communications device whose port base value is located at hex 40:02 is the second Asynchronous Communications Logical ID to be initialized and so on.

- The following is an example of a transmit sequence:

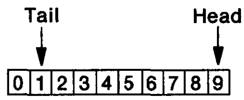
  1. The caller initializes the Transmit Request Block and calls the Start routine. The Transmit Head Pointer field and Transmit Tail Pointer field must be set within the range of 0 to the Transmit Buffer Length field, inclusive. In this example, the buffer length is set to 10 and the transmit buffer is full when the Transmit Head Pointer field equals 9. The maximum characters that may be sent without the caller changing the Transmit Head Pointer field is 9 (bytes 0 through 8).



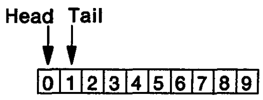
  2. The Start routine enables the transmit interrupt but does not send any data to the communication port.

  3. Because the Transmitter Holding Register is empty, an interrupt is generated.

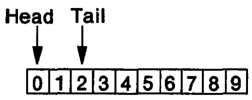
  4. The caller calls the Interrupt routine. A character is sent to the communication port and the Transmit Tail pointer is incremented by 1. In this example, byte 0 is sent, and the Transmit Tail Pointer field is set to 1.



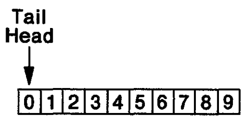
5. There are 8 bytes left to send, bytes 1 through 8. To increase the number of bytes in the buffer to send from 8 to 9, the caller may wrap the Transmit Head Pointer field back to the beginning of the buffer. In this example, the Transmit Head Pointer field is reset to 0.



6. Again, when the transmitter holding register empties, an interrupt is generated to the processor. The Interrupt routine is called, and a character is sent to the communication port. The Transmit Tail pointer is incremented by 1. In this example, byte 1 is sent, and the Transmit Tail Pointer field is set to 2.

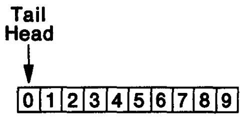


7. Assuming the caller does not change the Transmit Head Pointer field, the process repeats as interrupts occur. Byte 2 is sent, and the Transmit Tail pointer is set to 3 and so forth until byte 9 is sent and the Transmit Tail Pointer field is wrapped back to 0. At this point, the Transmit Tail Pointer field equals the Transmit Head Pointer field and the caller is informed of the buffer-empty condition (Operation Status). If the buffer-empty condition persists when the next transmit interrupt is generated, ABIOS disables the transmit interrupt and sets the Return Code field value equal to 0.



- The following is an example of a receive sequence:

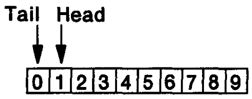
  1. The caller initializes the receive Request Block and calls the Start routine. The Receive Head pointer and Receive Tail pointer must be set within the range from 0 to the Receive Buffer Length field, inclusive. In this example, the buffer length is set to 10 and the receive buffer is full when the Receive Head pointer is logically 1 byte less than the Receive Tail pointer. The maximum number of characters received without the caller changing the Receive Tail pointer is 9 (bytes 0 through 8).

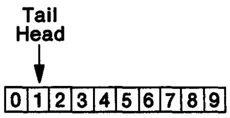
  2. The Start routine enables the receive interrupt but does not read any data.

  3. When data is available at the communication port, an interrupt is generated.

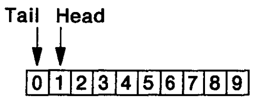
  4. The Interrupt routine is called. A character is read from the communication port, placed at location 0 and the Receive Head pointer is increased by 1.

  5. Eight locations are available in the buffer to receive data (bytes 1 through 8). To increase the number of bytes in the buffer to receive data, the caller may read data out of the buffer and logically increase the Receive Tail pointer up to and including the Receive Head pointer. If the Receive Head pointer and Receive Tail pointer are equal, the buffer is empty.

6. Nine available locations are in the receive buffer (bytes 1 through 9). When the next receive interrupt is generated, the Interrupt routine is called, and a character is read from the communication port. The Receive Head pointer is increased by 1. Assuming that the caller makes no changes to the Receive Tail pointer, the process continues. ABIOS receives a character, places the character at location 1, and increases the Receive Head pointer by 1. This process is repeated until ABIOS receives a character and places the character at location 9. Because the Receive Head pointer, when increased by 1, is outside the buffer range, ABIOS wraps the Receive Head pointer back to 0.



7. The receive buffer is full when the Receive Head pointer is 1 byte less than the Receive Tail pointer or if the Receive Tail pointer is 0 and the Receive Head pointer is 1 byte less than the Receive Buffer Length. If the buffer-full condition exists at entry to the Interrupt routine, the character that caused the interrupt is lost.

System Timer

Functions
The following are the system timer functions. The Default Interrupt Handler function and the Return Logical ID Parameters function are described in “Request Block” on page 4-3.

00H - Default Interrupt Handler

01H - Return Logical ID Parameters

02H - Reserved

03H - Read Device Parameters (Reserved)

02H - 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

Value	Description
0000H	Operation Completed Successfully
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 6-8. System Timer Return Codes

Programming Considerations

- The system timer interrupt is handled through the Default Interrupt Handler.

Real-Time Clock

Functions

The following are the real-time clock functions. The Default Interrupt Handler function and the Return Logical ID Parameters function are described in "Request Block" on page 4-3.

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.
- The Periodic Interrupt Rate field is valid only when the Periodic interrupt is enabled.
- The Alarm Hours, Minutes, and Seconds fields are valid only when the Alarm interrupt is enabled.
- If the Real-Time Clock is not started, the Return Code field is set to Real-Time Clock Not Started (hex 8001), and no other output fields are valid.
- The possible values of the Return Code field are equal to hex 0000, 8000, and 8001.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	16H	Reserved

03H - Read Device Parameters (continued)

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 µs 0010 - 61.035 µs 0011 - 122.070 µs 0100 - 244.141 µs 0101 - 488.281 µs 0110 - 976.562 µs 0111 - 1.953125 ms 1000 - 3.90625 ms 1001 - 7.8125 ms 1010 - 15.625 ms 1011 - 31.250 ms 1100 - 62.500 ms 1101 - 125.00 ms 1110 - 250.00 ms 1111 - 500.00 ms
Byte	11H	Real-Time Clock status byte Bit 7 - Set Bit Status 0 - Clock started 1 - Clock not started Bit 6 - Periodic interrupt bit 0 - Interrupt disabled 1 - Interrupt enabled Bit 5 - Alarm interrupt bit 0 - Interrupt disabled 1 - Interrupt enabled Bit 4 - Update-ended interrupt bit 0 - Interrupt disabled 1 - Interrupt enabled Bits 3, 2 - Reserved Bit 1 - Clock mode 0 - 12-hour mode 1 - 24-hour mode Bit 0 - Reserved
Byte	12H	Alarm hour in binary coded decimal (00 - 23D)
Byte	13H	Alarm minute in binary coded decimal (00-59D)
Byte	14H	Alarm seconds in binary coded decimal (00-59D)

04H - Set Device Parameters

- This routine sets the Real-Time Clock to its different modes.
- The possible 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 routine sets the alarm time according to the input values.
- The Real-Time Clock must be started before calling this function by using the Write Time and Date function (hex 12).
- A single Real-Time Clock interrupt may indicate the occurrence of multiple interrupt types (alarm, periodic, or update-ended interrupt). These interrupt types are activated individually through the Set Alarm Interrupt (hex 0B), Set Periodic Interrupt (hex 0D), or Set Update-Ended Interrupt (hex 0F) functions. When a Real-Time Clock interrupt occurs with multiple interrupt type requests active, the caller is required to make a single call with any one of the outstanding request blocks to service the multiple outstanding requests. Upon return, the interrupt pending status field indicates which interrupts occurred and were serviced.
- The Cancel Alarm Interrupt function (hex 0C) is provided to cancel an outstanding Set Alarm Interrupt function request and must be called before changing a previously set alarm time.
- The possible values of the Return Code field are equal to hex 0001, 0005, 8000, 8001, and 8002.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Byte	12H	Hours in binary coded decimal (00 - 23D)
Byte	13H	Minutes in binary coded decimal (00 - 59D)
Byte	14H	Seconds in binary coded decimal (00 - 59D)
DWord	16H	Reserved

Service Specific Output

SIZE	OFFSET	DESCRIPTION
Byte	1AH	Interrupt pending status Bit 7 - Reserved Bit 6 - Periodic interrupt Bit 5 - Alarm interrupt Bit 4 - Update-Ended interrupt Bits 3 to 0 - Reserved

0CH - Cancel Alarm Interrupt

- This function disables the alarm interrupt.
- The possible value of the Return Code field is equal to hex 0000.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	16H	Reserved

Service Specific Output

SIZE	OFFSET	DESCRIPTION
None		

0DH - Set Periodic Interrupt

- This routine sets the periodic interrupt periodic interval.
- The Real-Time Clock must be started before calling this function by way of the Write Time and Date function (hex 12).
- A single Real-Time Clock interrupt may indicate the occurrence of multiple interrupt types (alarm, periodic, or update-ended interrupt). These interrupt types are activated individually through the Set Alarm Interrupt (hex 0B), Set Periodic Interrupt (hex 0D), or Set Update-Ended Interrupt (hex 0F) functions. When a Real-Time Clock interrupt occurs with multiple interrupt type requests active, the caller is required to make a single call with any one of the outstanding request blocks to service the multiple outstanding requests. Upon return, the interrupt pending status field indicates which interrupts occurred and were serviced.
- The Cancel Periodic Interrupt function (hex 0E) is provided to cancel an outstanding Set Periodic Interrupt request, and must be called before changing a previously set periodic interrupt interval.
- The possible values of the Return Code field are equal to hex 0001, 0005, 8001, and 8002.

0DH - Set Periodic Interrupt (continued)

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Byte	10H	Periodic interrupt rate set Bits 7 to 4 - Reserved Bits 3 to 0 - Rate value set 0000 - None 0001 - 30.517 µs 0010 - 61.035 µs 0011 - 122.070 µs 0100 - 244.141 µs 0101 - 488.281 µs 0110 - 976.562 µs 0111 - 1.953125 ms 1000 - 3.90625 ms 1001 - 7.8125 ms 1010 - 15.625 ms 1011 - 31.250 ms 1100 - 62.500 ms 1101 - 125.00 ms 1110 - 250.00 ms 1111 - 500.00 ms
DWord	16H	Reserved

Service Specific Output

SIZE	OFFSET	DESCRIPTION
Byte	1AH	Interrupt pending status Bit 7 - Reserved Bit 6 - Periodic interrupt Bit 5 - Alarm interrupt Bit 4 - Update-Ended interrupt Bits 3 to 0 - Reserved

0EH - Cancel Periodic Interrupt

- This routine disables the periodic interrupt.
- The possible value of the Return Code field is equal to 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 Real-Time Clock must be started before calling this function by way of the Write Time and Date function (hex 12).
- A single Real-Time Clock interrupt may indicate the occurrence of multiple interrupt types (alarm, periodic, or update-ended interrupt). These interrupt types are activated individually through the Set Alarm Interrupt (hex 0B), Set Periodic Interrupt (hex 0D) or Set Update-Ended Interrupt (hex 0F) functions. When a Real-Time Clock interrupt occurs with multiple interrupt type requests active, the caller is required to make a single call with any one of the outstanding request blocks to service the multiple outstanding requests. Upon return, the interrupt pending status field indicates which interrupts occurred and were serviced.
- The Cancel Update-Ended Interrupt function (hex 10) is provided to cancel an outstanding Set Update-Ended Interrupt request, and must be called before restarting the Update-Ended interrupt.
- The possible values of the Return Code field are equal to 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 Bit 5 - Alarm interrupt Bit 4 - Update-Ended interrupt Bits 3 to 0 - Reserved

10H - Cancel Update-Ended Interrupt

- This function disables the Update-Ended Interrupt.
- The possible value of the Return Code field is equal to 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 before calling this function by using the Write Time and Date function (hex 12).
- The Time and Date fields are valid only when the Return Code field is set to Operation Completed Successfully (hex 0000).
- The possible values of the Return Code field are equal to hex 0000, 8000, and 8001.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	10H	Reserved

Service Specific Output

SIZE	OFFSET	DESCRIPTION
Byte	12H	Alarm hour in binary coded decimal (00 - 23D)
Byte	13H	Alarm minute in binary coded decimal (00-59D)
Byte	14H	Alarm seconds in binary coded decimal (00-59D)
Byte	15H	Century in binary coded decimal (19D or 20D)
Byte	16H	Year in binary coded decimal (00 - 99D)
Byte	17H	Month in binary coded decimal (01D - 12D)
Byte	18H	Day in binary coded decimal (01D - 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 possible value of the Return Code field is equal to hex 0000.

Service Specific Input

```
SIZE    OFFSET     DESCRIPTION

Word     10H       Reserved
Byte     12H       Hours in binary coded decimal (00 - 23D)
Byte     13H       Minutes in binary coded decimal (00 - 59D)
Byte     14H       Seconds in binary coded decimal (00 - 59D)
Byte     15H       Century in binary coded decimal (19D or 20D)
Byte     16H       Year in binary coded decimal (00 - 99D)
Byte     17H       Month in binary coded decimal (01D - 12D)
Byte     18H       Day in binary coded decimal (01D - 31D)
```

Service Specific Output

```
SIZE    OFFSET     DESCRIPTION

None
```

Return Codes

Value	Description
0000H	Operation Completed Successfully
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 6-9. Real-Time Clock Return Codes

Programming Considerations

- If the Real-Time Clock is in a clock update cycle, the Return Code field is set to Device In Use (hex 8000).

Notes:

System Services

Functions

The following are the system services functions. The Default Interrupt Handler function and the Return Logical ID Parameters function are described in "Request Block" on page 4-3.

00H - Default Interrupt Handler

01H - Return Logical ID Parameters

02H - Reserved

03H - Used Internally by ABIOS

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, the Nonmaskable interrupt (NMI), I/O Channel Check, and Parity are disabled. ABIOS gives control back to the caller at the location pointed to by the Resume Pointer field. The caller must reenable interrupts and the NMI.
- For 80386 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).
- 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 it can be checked upon 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 ABIOS

0DH - Enable Address Line 20

- This function enables Address Line 20.
- The possible value of the Return Code field is equal to hex 0000.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
DWord	10H	Reserved

Service Specific Output

```
SIZE     OFFSET       DESCRIPTION

None
```

0EH - Disable Address Line 20

- This function disables Address Line 20.
- The possible value of the Return Code field is equal to 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 equals 0, or the value of the Duration Counter field is equal to 0, no action is performed, and the Return Code field is set to Operation Completed Successfully (hex 0000).
- The possible value of the Return Code field is equal to hex 0000.

Service Specific Input

```
SIZE    OFFSET      DESCRIPTION

DWord    10H        Reserved
Word     60H        Frequency divisor
                    (1.19 MHz/freq.)= # frequency
                    (freq. div. = 1331 for 886 Hz freq.)
Byte     66H        Duration counter in 1/64 seconds
```

Service Specific Output

```
SIZE    OFFSET      DESCRIPTION

None
```

Return Codes

Value	Description
0000H	Operation Completed Successfully
C000H	Invalid Logical ID (ABIOS Transfer Convention only)
C001H	Invalid Function Number
C003H	Invalid Unit Number
C004H	Invalid Request Block Length

Figure 6-10. System Services Return Codes

Nonmaskable Interrupt (NMI)

Functions
The following are the NMI functions. The Default Interrupt Handler function and the Return Logical ID Parameters function are described in “Request Block” on page 4-3.

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 unmasks the NMI.
- The possible value of the Return Code field is equal to 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 system board memory parity and I/O channel check.
- The DMA bus time-out NMI and the watchdog time-out NMI cannot be disabled through ABIOS.
- The possible value of the Return Code field is equal to hex 0000.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	21H	Reserved

Service Specific Output

SIZE	OFFSET	DESCRIPTION
None		

08H - Continuous Read

- This function returns the NMI status.
- If the NMI is caused by a DMA arbitration bus time-out, the Interrupt routine returns the arbitration level that caused the time-out.
- Upon return from the NMI Interrupt routine, the NMI is disabled. The caller should reenable the NMI through the Enable function (hex 06) if NMI interrupts are desired.
- The possible values of the Return Code field are equal to hex 0001 and 0005.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	21H	Reserved

08H - Continuous Read (continued)

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
                       which caused the DMA bus time-out
Byte      1FH        Slot number which caused the I/O channel check
```

09H - Write (Reserved)

0AH - Additional Data Transfer (Reserved)

Return Codes

Value	Description
0000H	Operation Completed Successfully
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 6-11. Nonmaskable Interrupt Return Codes

Notes:

Pointing Device

Functions

The following are the pointing device functions. The Default Interrupt Handler function and the Return Logical ID Parameters function are described in "Request Block" on page 4-3.

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 possible values for the Data Package Size field are 1 through 8.
- The possible values of the Current Sample Rate field are hex 0A, 14, 28, 3C, 50, and C8.
- The possible value of the Return Code field is equal to 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 enable
                        0 - Disabled
                        1 - Enabled
                Bits 4 to 0 - Reserved
Byte     11H    Data package size
```

Service Specific Output (continued)

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 1:1 1 - Scaling 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/1 mm 01H - 2 count/1 mm 02H - 4 count/1 mm 03H - 8 count/1 mm 04H to FFFFH - Reserved
Word	16H	Current sample rate
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. Upon completion the pointing device is initialized as follows:
  - Package size remains unchanged.
  - Resolution = 4 counts/mm.
  - Sample rate = 100 reports/second.
  - Scaling = 1:1.
  - The pointing device is disabled.
- The possible values of the Return Code field are equal to 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 equal to 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 equal to 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 - Continuous Read

- This function is called to read the pointing device; the pointing device remains disabled. This function must be called before any other pointing device function except the Return Logical ID Parameters function (hex 01). Once it is called, the pointing device can be enabled by calling the Enable function (hex 06).
- The possible values of the Return Code field are equal to hex 0001H, 0002H, 0005H, 0009H, 8000H, 8003H, and C005H.

Service Specific Input

```
SIZE    OFFSET  DESCRIPTION

Byte     10H    Package size (in bytes)
                  00H - Reserved
                  01H - 1
                  02H - 2
                  03H - 3
                  04H - 4
                  05H - 5
                  06H - 6
                  07H - 7
                  08H - 8
                  09H to FFH - Reserved
DWord    12H    Reserved
```

Service Specific Output

```
SIZE    OFFSET   DESCRIPTION

Byte      1CH    String of 12 bytes of data returned from the pointing device
```

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 Sample Rate field are hex 0A, 14, 28, 3C, 50, 64, and C8.
- The possible values of the Return Code field are equal to hex 0000, 0001, 0002, 0005, 8000, 8001, 8002, and 8003.

Service Specific Input

```
SIZE    OFFSET  DESCRIPTION

Word     12H    Sample rate in reports/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 equal to hex 0000, 0001, 0002, 0005, 8000, 8001, 8002, and 8003.

Service Specific Input

```
SIZE    OFFSET  DESCRIPTION

Word     12H    Resolution
                  00H - 1 count/1 mm
                  01H - 2 counts/1 mm
                  02H - 4 counts/1 mm
                  03H - 8 counts/1 mm
                  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 equal to 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 1:1 02H - Set scaling 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 equal to 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	Description
0000H	Operation Completed Successfully
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
9100H	Controller Failure
C000H	Invalid Logical ID (ABIOS Transfer Convention only)
C001H	Invalid Function Number
C003H	Invalid Unit Number
C004H	Invalid Request Block Length

Figure 6-12. Pointing Device Return Codes

Programming Considerations

- The Read Pointing Device ID function (hex 0E) returns the Device ID read from the auxiliary device interface. The IBM Pointing Device ID is hex 00.

- For pointing device Read function (hex 08), the following byte definitions apply:

```
Byte 1 - Status from pointing device
         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 2 - X data from the pointing device
Byte 3 - Y data from the pointing device
Bytes 4 to 12 - Reserved
```

Nonvolatile Random Access Memory (NVRAM)

Functions

The following are the NVRAM functions. The Default Interrupt Handler function and the Return Logical ID Parameters function are described in “Request Block” on page 4-3.

00H - Default Interrupt Handler

01H - Return Logical ID Parameters

02H - Reserved

03H - Read Device Parameters

- This function returns the NVRAM device parameters
- The possible value of the Return Code field is equal to hex 0000.

Service Specific Input

```
SIZE    OFFSET  DESCRIPTION

Word     16H    Reserved
```

Service Specific Output

```
SIZE    OFFSET  DESCRIPTION

Word     22H    Start of user RAM
Word     24H    Length of user RAM
```

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 specific location of the requested RAM (64-byte RAM or extended RAM).
- If the value of the Number of Bytes to Transfer field is equal to 0, no action is performed and the Return Code field is set to Operation Completed Successfully (hex 0000).
- If the value of the Number of Bytes to Transfer 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 Invalid NVRAM Parameter (hex C005). The maximum number of bytes is returned by the Read Device Parameters function (hex 03).
- The possible values of the Return Code field are equal to hex 0000, 80FE, 80FF, and C005.

Service Specific Input

```
SIZE    OFFSET  DESCRIPTION

Word     10H    Reserved
DWord    12H    Data Pointer 1
Word     16H    Reserved
DWord    1AH    Data Pointer 2
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 transfer
```

Service Specific Output

```
SIZE    OFFSET  DESCRIPTION

None
```

09H - Write NVRAM

- This function writes the data supplied to the specified location of the requested RAM (64-byte RAM or extended RAM).
- If the value of the Number of Bytes to Transfer field is equal to 0, no action is performed, and the Return Code field is set to Operation Completed Successfully (hex 0000).
- If the value of the Number of Bytes to Transfer 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 Invalid NVRAM Parameter (hex C005).
- The possible values of the Return Code field are equal to hex 0000, 80FE, 80FF, and C005.

Service Specific Input

```
SIZE    OFFSET  DESCRIPTION

Word     10H    Reserved
DWord    12H    Data pointer 1
Word     16H    Reserved
DWord    1AH    Data pointer 2
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 transfer
```

Service Specific Output

```
SIZE    OFFSET  DESCRIPTION

None
```

0AH Additional Data Transfer Function (Reserved)

0BH - Recompute Checksum

- This function recomputes the checksum for the requested RAM (64-byte RAM or extended RAM).
- The possible values of the Return Code field are equal to hex 0000, 80FE, 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

Value	Description
0000H	Operation Completed Successfully
80FEH	NVRAM Check Sum 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 NVRAM Parameter

Figure 6-13. Nonvolatile Random Access Memory (NVRAM) Return Codes

Programming Considerations

- The Read Device Parameters function (hex 03) returns the locations of RAM within the extended RAM that are allocated to the user. All other areas of the extended RAM and all areas of the 64-byte RAM are reserved by IBM.

Direct Memory Access (DMA)

Functions

The following are the DMA functions. The Default Interrupt Handler function and the Return Logical ID Parameters function are described in "Request Block" on page 4-3.

00H - Default Interrupt Handler

01H - Return Logical ID Parameters

02H - Reserved

03H Read Device Parameters

- This function returns the DMA device parameters.
- The possible value of the Return Code field is equal to hex 0000.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	16H	Reserved

Service Specific Output

SIZE	OFFSET	DESCRIPTION
Word	10H	Maximum address in 1MB
Word	12H	Maximum direct memory access transfer size in 1KB
Byte	14H	Number of arbitration levels
Byte	15H	Number of direct memory access channels

04H - Set Device Parameters (Reserved)

05H - Reset/Initialize (Reserved)

06H - Enable - for Interrupts (Reserved)

07H - Disabled - for Interrupts (Reserved)

08H - Read (Reserved)

09H - Write (Reserved)

0AH - Additional Data Transfer (Reserved)

0BH - Allocate Arbitration Level

- This function allocates an arbitration level. The value of the Arbitration Level to Allocate field should not exceed the value of the Number of Arbitration Levels field returned in the Read Device Parameters function.
- The possible values of the Return Code field are equal to hex 0000, 8001, 8006, and C005.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	16H	Reserved
Byte	1FH	Arbitration level to allocate

Service Specific Output

SIZE	OFFSET	DESCRIPTION

None

0CH - Deallocate Arbitration Level

- This function makes available a previously user-allocated arbitration level.
- Users of this function should only deallocate arbitration levels that have been previously allocated by the user.
- The possible values of the Return Code field are equal to hex 0000, 8002, 8004, and 8007.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	16H	Reserved
Byte	1FH	Arbitration level to deallocate

Service Specific Output

SIZE	OFFSET	DESCRIPTION

None

0DH - Disable Arbitration Level

- This function disables the arbitration level in the DMA controller for transfers that have been completed. Use the Abort function (hex 0F) to disable an arbitration level for a transfer that is in progress.
- An auto-initialized mode arbitration level can be disabled only through the Abort function (hex 0F).
- The possible values of the Return Code field are equal to hex 0000, 8002, and 8004.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	16H	Reserved
Byte	1FH	Arbitration level to disable

Service Specific Output

SIZE	OFFSET	DESCRIPTION
None		

0EH - Transfer Status

- This function returns the number of bytes left to transfer as read from the DMA controller.
- The possible values of the Return Code field are equal to hex 0000, 8002, and 8003.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	16H	Reserved
Byte	1FH	Arbitration level to check

Service Specific Output

SIZE	OFFSET	DESCRIPTION
DWord	18H	Number of bytes left to transfer

0FH - Abort

- This function accesses the DMA controller to disable an arbitration level in the middle of a transfer and to read the number of bytes left to transfer.
- This function should be used to disable a auto-initialized mode arbitration level.
- The possible values of the Return Code field are equal to hex 0000, 8002, 8003, and 8005.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
Word	16H	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	Count of data not transferred when abort was issued

10H - Read from Memory and Write to I/O

- This function programs the DMA controller with the indicated values from the Request Block.
- The possible values of the Return Code field are equal to hex 0000, 8002, 8004, and C005.

Service Specific Input

```
SIZE     OFFSET    DESCRIPTION

DWord     10H      Physical address of memory
DWord     14H      Physical address of I/O
DWord     18H      Count of data to transfer
Byte      1CH      Mode Control
                   Bits 7 to 3 - Reserved
                   Bit 2 - Programmable I/O
                   Bit 1 - Reserved
                   Bit 0 - Auto initialization
Byte      1DH      Transfer Control 1
                   Bits 7 to 3 - Reserved
                   Bit 2 - Count Control
                           0 - Increment
                           1 - Decrement
                   Bit 1 - Reserved
                   Bit 0 - Device size
                           0 - 8-bit
                           1 - 16-bit
Byte      1EH      Transfer Control 2
                   Bits 7 to 1 - Reserved
                   Bit 0 - Device size
                           0 - 8-bit
                           1 - 16-bit
Byte      1FH      Arbitration level to use
```

Service Specific Output

```
SIZE     OFFSET    DESCRIPTION

None
```

11H - Read from I/O and Write to Memory

- This function programs the DMA controller with the indicated values from the Request Block.
- The possible values of the Return Code field are equal to hex 0000, 8002, 8004, and C005.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
DWord	10H	Physical address of memory
DWord	14H	Physical address of I/O
DWord	18H	Count of data to transfer
Byte	1CH	Mode Control Bits 7 to 3 - Reserved Bit 2 - Programmable I/O Bit 1 - Reserved Bit 0 - Auto initialization
Byte	1DH	Transfer control 1 Bits 7 to 3 - Reserved Bit 2 - Count control 0 - Increment 1 - Decrement Bit 1 - Reserved Bit 0 - Device size 0 - 8-bit 1 - 16-bit
Byte	1EH	Transfer control 2 Bits 7 to 1 - Reserved Bit 0 - Device size 0 - 8-bit 1 - 16-bit
Byte	1FH	Arbitration level to use

Service Specific Output

```
SIZE    OFFSET    DESCRIPTION

None
```

12H - Verify

- This function programs the DMA controller with the indicated values from the Request Block.
- The possible values of the Return Code field are equal to hex 0000, 8002, 8004, and C005.

Service Specific Input

SIZE	OFFSET	DESCRIPTION
DWord	10H	Physical address of memory
DWord	14H	Physical address of I/O
DWord	18H	Count of data to transfer
Byte	1CH	Mode control
		Bits 7 to 3 - Reserved
		Bit 2 - Programmable I/O
		Bit 1 - Reserved
		Bit 0 - Auto initialization
Byte	1DH	Transfer control 1
		Bits 7 to 3 - Reserved
		Bit 2 - Count control
		0 - Increment
		1 - Decrement
		Bit 1 - Reserved
		Bit 0 - Device size
		0 - 8-bit
		1 - 16-bit
Byte	1EH	Transfer control 2
		Bits 7 to 1 - Reserved
		Bit 0 - Device size
		0 - 8-bit
		1 - 16-bit
Byte	1FH	Arbitration level to use

Service Specific Output

SIZE	OFFSET	DESCRIPTION
None		

Return Codes

Value	Description
0000H	Operation Completed Successfully
8000H	Device in Use
8001H	Arbitration Level Not Available
8002H	Arbitration Level Not Allocated
8003H	Arbitration Level Disabled
8004H	Transfer in Progress or the Arbitration Level is not Disabled
8005H	No Transfer in Progress
8006H	No Channel Available
8007H	Arbitration Level Not Disabled
C000H	Invalid Logical ID (ABIOS Transfer Convention only)
C001H	Invalid Function Number
C003H	Invalid Unit Number
C004H	Invalid Request Block Length
C005H	Invalid DMA Parameters

Figure 6-14. Direct Memory Access (DMA) Return Codes

Programming Considerations

- DMA channels are defined as being physical or virtual. A physical channel can only have 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 or virtual channels. Priority of the channels is determined by the arbitration level, with arbitration level 0 having the highest priority and arbitration level hex 0E having the lowest.

  Note: Arbitration level F is reserved.
- To perform a DMA transfer operation, a caller performs the following steps:
  1. Ask for an arbitration level.
  2. Set up a transfer to a device.
  3. Disable the arbitration level.
  4. Deallocate the arbitration level.
- Direct reading/writing of the DMA controller ports may cause unpredictable results.

- The Mode Control field, and the Transfer Control fields 1 and 2 are filled in by the caller to indicate what characteristics should be programmed to the DMA controller for the requested service.
- The Mode Control field bits are provided to use the auto-initialization and programmable I/O options provided by the DMA controller. The auto-initialization bit determines if auto-initialization should occur when the transfer reaches terminal count. The programmable I/O bit is set by the caller to indicate that the I/O address is to be programmed to the DMA controller. This will drive the indicated I/O address on the bus during the DMA cycles instead of an I/O address of 0.
- Transfer Control Fields 1 and 2 are used to give information relating to the physical address of memory field and the physical address of I/O field for memory to I/O, I/O to memory, and verify functions. The device size bits indicate if the transfer is 8 or 16 bits. The Count control field is used to specify if the physical address is incremented or decremented during a transfer.

Notes:

Programmable Option Select (POS)

Functions

The following are the POS functions. The Default Interrupt Handler function and the Return Logical ID Parameters function are described in "Request Block" on page 4-3.

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 requested 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 four 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 Invalid POS Parameter (hex C005).
- The possible values of the Return Code field are equal to 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
                              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
```

0CH - Write Stored POS Data from Memory

- This function writes the programmable option select data to the requested 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 Invalid POS Parameter (hex C005).
- The possible values of the Return Code field are equal to 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
                                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
```

0DH - Read Dynamic POS Data to Memory

- This function reads the supplied programmable option select data to the adapter in the requested 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 Invalid POS Parameter (hex C005).
- The possible values of the Return Code field are equal to 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
                                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
```

0EH - Write Dynamic POS Data from Memory

- This function writes the supplied programmable option select data to the adapter in the requested 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 Invalid POS Parameter (hex C005).
- The possible values of the Return Code field are equal to 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
                                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
```

Return Codes

Value	Description
0000H	Operation Completed Successfully
80FEH	NVRAM Check Sum 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 6-15. Programmable Option Select (POS) Return Codes

Keyboard Security

Functions

The following are the keyboard security functions. The Default Interrupt Handler function and the Return Logical ID Parameters function are described in "Request Block" on page 4-3.

00H - Default Interrupt Handler

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 equal to 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 equal to 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.
- If the Password Length field is 0, or greater than the maximum password length, no action is performed and the Return Code field is set to Invalid Keyboard Security Parameter (hex C005).
- The maximum password length is returned in the Read Device Parameters function (hex 03).
- The possible values of the Return Code field are equal to 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. This byte is used to signal the system that security is enabled with a valid password. After enabling security, the system sends this byte (by using the keyboard interrupt) to the operating system as if it were a scan code. If the invocation byte is 0, the system does not send this byte after enabling keyboard security.
- The possible values of the Return Code field are equal to 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. This byte is used to signal the system that security is deactivated with the correct password. After the correct sequence is typed, the system sends this byte (by using the keyboard interrupt) as if it were a scan code to the operating system. If the match byte is 0, the system does not send the this byte when keyboard security is disabled.
- The possible values of the Return Code field are equal to hex 0000, 8000, and 8003.

Service Specific Input

```
SIZE    OFFSET  DESCRIPTION

Byte     10H    Scan code
Byte     11H    Reserved
```

Service Specific Output

```
SIZE    OFFSET  DESCRIPTION

None
```

0EH - Write Filter Byte 1

- This function changes filter byte 1. The filter bytes are scan codes that are ignored during password validation. For example, it might be desirable to ignore the scan code for the shift keys.
- The possible values of the Return Code field are equal to 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
```

0FH - Write Filter Byte 2

- This function changes filter byte 2. The filter bytes are scan codes that are ignored during password validation. For example, it might be desirable to ignore the scan code for the shift keys.
- The possible values of the Return Code field are equal to 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

Value	Description
0000H	Operation Completed Successfully
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 6-16. Keyboard Security Return Codes

Index

Contents - Supplements

File supplements behind this page. Enter the name and the date of each supplement in the space provided below.

NAME	DATE
Asynchronous Communications	July, 1987
Programmable Option Select	May, 1988

NAME	DATE

Asynchronous Communications Supplement

This supplement contains programming considerations that apply to the Advanced BIOS section of the *IBM Personal System/2 and IBM Personal Computer BIOS Interface Technical Reference*.

This information should be used in addition to the material covered in the Advanced BIOS (ABIOS) section of this manual.

Programming Considerations

In the following listing the specific ABIOS function affected is identified by its Device ID, Secondary Device ID, and Revision level. This information is contained in the readable public data area of the Device Block.

Asynchronous Communications

Device ID	06H
Secondary Device ID	00H
Revision	00H
Function	Reset/Initialize (hex 05)
Description	The hardware receive buffer is not cleared.

Asynchronous Communications

Device ID	06H
Secondary Device ID	00H
Revision	00H
Functions	Transmit Interrupt (hex 0E) Receive Interrupt (hex 0F) Combined Interrupts (hex 10)
Description	The head and tail pointers may equal the buffer length when their maximum values should be buffer length minus 1.

Asynchronous Communications

Device ID	06H
Secondary Device ID	00H
Revision	00H
Functions	Receive Interrupt (hex 0F) Combined Interrupts (hex 10)
Description	When an overrun error occurs with null stripping activated and the character that caused the overrun is a null character, the receive routine discards the null character but does not indicate that the overrun character was a null character. This condition is normally indicated by setting bit 12 of the Operation Status field.

Programmable Option Select Supplement

This supplement contains programming considerations that apply to the Advanced BIOS section of the *IBM Personal System/2 and IBM Personal Computer BIOS Interface Technical Reference*.

This information should be used in addition to the material covered in the Advanced BIOS (ABIOS) section of this manual.

May 5, 1988

Programming Considerations

In the following listing the specific ABIOS function affected is identified by its Device ID, Secondary Device ID, and Revision level. This information is contained in the readable public data area of the Device Block.

Programmable Option Select (POS)

Device ID	10H
Secondary Device ID	00H
Revision	00H, 01H
Functions	Read Stored POS Data to Memory (hex 0B) Write Stored POS Data from Memory (hex 0C) Read Dynamic POS Data to Memory (hex 0D) Write Dynamic POS Data from Memory (hex 0E)
Description	The value of the Slot Number field is not checked against the maximum number of slots available. Therefore, the Return Code field is not set to Invalid POS Parameter.

15F0306