Section 3. BIOS Data Areas

The BIOS data areas are allocated specifically as work areas for system BIOS and adapter BIOS. The BIOS routines use 256 bytes of memory from absolute address hex 400 to hex 4FF. A description of the BIOS data areas follows.

Address (Hex)	Function	Size
40:00	RS-232C communication line 1 port base address	Word
40:02	RS-232C communication line 2 port base address	Word
40:04	RS-232C communication line 3 port base address	Word
40:06	RS-232C communication line 4 port base address	Word

Note: The RS-232C communication line port base address fields can be initialized to 0 by the POST if the system configuration contains less than four serial ports. The POST never puts 0 in the RS-232C communication line port base address table between two valid RS-232C communication line port base addresses.

Figure 3-1. RS-232C Port Base Address Data Area

Address (Hex)	Function	Size
40:08	Printer 1 port base address	Word
40:0A	Printer 2 port base address	Word
40:0C	Printer 3 port base address	Word
40:0E	Reserved	Word
Exception:		
40:0E	Printer 4 port base address (PC, PC/XT, AT, and PC Convertible)	Word

Note: The printer port base address fields can be initialized to 0 by the POST if the system configuration contains less than four parallel ports. The POST never puts 0 in the printer port base address table between two valid printer port base addresses.

Figure 3-2. Printer Port Base Address Data Area

Address (Hex)	Function	Size
40:10	Installed hardware	Word
Bits 15, 14	Number of printer adapters	
Bit 13	Reserved	
Bit 12	Reserved	
Bits 11 to 9	Number of RS-232C adapters	
Bit 8	Reserved	
Bits 7, 6	Number of diskette drives (0-based)	
Bits 5, 4	Video mode type (values are binary) = 00 - Reserved = 01 - 40x25 color = 10 - 80x25 color = 11 - 80x25 monochrome	
Bit 3	Reserved	
Bit 2	Pointing device	
Bit 1	Math coprocessor	
Bit 0	IPL diskette	
Exceptions:		
Bit 13	Internal modem (for PC Convertible and Personal System/2 Model 55 LS)	
Bit 2	Reserved (for PC, PC/XT, AT, and PC Convertible)	

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

Figure 3-3. System Equipment Data Area

Address (Hex)	Function	Size
40:12	Reserved	Byte
Exceptions:		
40:12	Power-on self-test status (for PC Convertible only)	Byte
	POST system flag (for Personal System/2 Model 25 only)	Byte
Bit 1	Real-time clock (RTC) installed	
Bit 0	Optional memory failed; memory remapped	

Figure 3-4. Miscellaneous Data Area 1

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

Figure 3-5. Memory Size Data Area

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	
Note: For 101/102-key keyboard support, bits 2 and 3 are set if either or both the Alt key and the Ctrl key are pressed.		
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	SysRq 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 = 00H - No error = 01H - Invalid diskette drive parameter = 02H - Address mark not found = 03H - Write-protect error = 04H - Requested sector not found = 06H - Diskette change line active = 08H - DMA overrun on operation = 09H - Attempt to DMA across a 64KB boundary = 0CH - Media type not found = 10H - CRC error on diskette read = 20H - General controller failure = 30H - Drive does not support media sense = 31H - No media in drive = 32H - Drive does not support media type = 40H - Seek operation failed = 80H - Time-out = AAH - Diskette drive not ready	Byte

Figure 3-7 (Part 1 of 2). Diskette Drive Data Area

Address (Hex)	Function	Size
40:42	Diskette-drive controller status bytes	7 bytes

Figure 3-7 (Part 2 of 2). 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	Display controller base address	Word
40:65	Current setting of 3x8 register	Byte
40:66	Current setting of 3x9 register	Byte

Figure 3-8. Video-Control Data Area 1

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 (for Personal System/2 products except Model 25 and Model 30) Reset flag at hex 40:72 = 4321H	DWord

Figure 3-9. System Data Area 1

Address (Hex)	Function	Size
40:6C	Timer counter	DWord
40:70	Timer overflow (If nonzero, the timer has counted past 24 hours.)	Byte

Figure 3-10. System-Timer Data Area

Address (Hex)	Function	Size
40:71 Bit 7	Break key state = 1 - Break key pressed = 0 - Break key not pressed	Byte
40:72	Reset flag = 1234H - Bypass memory test = 4321H - Preserve memory (for Personal System/2 products except Model 25 and Model 30) = 5678H - System suspended (for PC Convertible) = 9ABCH - Manufacturing test mode (for PC Convertible) = ABCDH - System POST loop mode (for PC Convertible)	Word

Figure 3-11. System Data Area 2

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
40:75	Number of fixed disk drives attached	Byte
40:76	Reserved	Byte
40:77	Reserved	Byte
Exceptions:		
40:74	Reserved (for devices using ESDI-type commands)	Byte
40:76	Fixed disk drive control (for PC/XT)	Byte
40:77	Fixed disk drive controller port (for PC/XT)	Byte

Figure 3-12. Fixed Disk Drive Data Area

Address (Hex)	Function	Size
40:78	Printer 1 time-out value	Byte
40:79	Printer 2 time-out value	Byte
40:7A	Printer 3 time-out value	Byte
40:7B	Generic SCSI CBIOS and bus-master filter flags	
Bits 7, 6	Reserved	
Bit 5	Operating-system DMA services indicator = 0 - Not supported; all memory is mapped linear = physical = 1 - Supported	
Bit 4	Reserved	
Bit 3	Interrupt-4BH-intercepted indicator = 0 - Interrupt vector is not intercepted = 1 - Interrupt vector is intercepted	
Bit 2	Reserved	
Bit 1	Generic SCSI CBIOS services support = 0 - Interrupt 4BH does not support SCSI = 1 - Interrupt 4BH supports SCSI	
Bit 0	Reserved	
Exception:		
40:7B	Printer 4 time-out value (for PC, PC/XT, AT, and Personal System/2 Model 25 and Model 30)	Byte

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

Address (Hex)	Function	Size
40:7C	RS-232C communication line 1 time-out value	Byte
40:7D	RS-232C communication line 2 time-out value	Byte
40:7E	RS-232C communication line 3 time-out value	Byte
40:7F	RS-232C communication line 4 time-out value	Byte

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

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 - 1MB per second	
Bits 5, 4	Last diskette drive step rate selected = 00 - 0CH = 01 - 0DH = 10 - 0EH = 11 - 0AH	
Bit 3	Reserved	
Bit 2	Reserved	
Bit 1	Reserved	
Bit 0	Reserved	
40:8C	Fixed disk drive controller status	Byte
40:8D	Fixed disk drive controller error status	Byte
40:8E	Fixed disk drive interrupt control	Byte
40:8F	Reserved	Byte

Figure 3-17 (Part 1 of 5). Diskette Drive/Fixed Disk Drive Control Data Area

Address (Hex)	Function	Size
40:90	Drive 0 media state	Byte
Bits 7, 6	Data transfer rate (values in binary) = 00 - 500KB per second = 01 - 300KB per second = 10 - 250KB per second = 11 - 1MB per second	
Bit 5	Double stepping = 1 - Double stepping required = 0 - Double stepping not required	
Bit 4	Media/drive established = 1 - Media/drive has been established = 0 - Media/drive has not been established	
Bit 3	4MB media capability = 1 - Drive is capable of supporting 4MB media = 0 - Drive is not capable of supporting 4MB media	
Bits 2 to 0	Drive/media state (values in binary) = 000 - 360KB diskette in 360KB drive is not established = 001 - 360KB diskette in 1.2MB drive is not established = 010 - 1.2MB diskette in 1.2MB drive is not established = 011 - 360KB diskette in 360KB drive is established = 100 - 360KB diskette in 1.2MB drive is established = 101 - 1.2MB diskette in 1.2MB drive is established = 110 - Reserved = 111 - None of the above	

Figure 3-17 (Part 2 of 5). Diskette Drive/Fixed Disk Drive Control Data Area

Address (Hex)	Function	Size
40:91	Drive 1 media state	Byte
Bits 7, 6	Data transfer rate (values in binary) = 00 - 500KB per second = 01 - 300KB per second = 10 - 250KB per second = 11 - 1MB per second	
Bit 5	Double stepping = 1 - Double stepping required = 0 - Double stepping not required	
Bit 4	Media/drive established = 1 - Media/drive has been established = 0 - Media/drive has not been established	
Bit 3	4MB media capability = 1 - Drive is capable of supporting 4MB media = 0 - Drive is not capable of supporting 4MB media	
Bits 2 to 0	Drive/media state (values in binary) = 000 - 360KB diskette in 360KB drive is not established = 001 - 360KB diskette in 1.2MB drive is not established = 010 - 1.2MB diskette in 1.2MB drive is not established = 011 - 360KB diskette in 360KB drive is established = 100 - 360KB diskette in 1.2MB drive is established = 101 - 1.2MB diskette in 1.2MB drive is established = 110 - Reserved = 111 - None of the above	

Figure 3-17 (Part 3 of 5). Diskette Drive/Fixed Disk Drive Control Data Area

Address (Hex)	Function	Size
40:92	Drive 2 media state	Byte
Bits 7, 6	Data transfer rate (values in binary) = 00 - 500KB per second = 01 - 300KB per second = 10 - 250KB per second = 11 - 1MB per second	
Bit 5	Double stepping = 1 - Double stepping required = 0 - Double stepping not required	
Bit 4	Media/drive established = 1 - Media/drive has been established = 0 - Media/drive has not been established	
Bit 3	4MB media capability = 1 - Drive is capable of supporting 4MB media = 0 - Drive is not capable of supporting 4MB media	
Bit 2	Multiple data rate capability determined = 1 - Multiple data rate capability is determined = 0 - Multiple data rate capability is not determined	
Bit 1	Multiple data rate capability = 1 - Multiple data rate capability = 0 - No multiple data rate capability	
Bit 0	80-track capability (drive 2) = 1 - Drive 2, 80 tracks = 0 - Drive 2, 40 tracks	

Figure 3-17 (Part 4 of 5). Diskette Drive/Fixed Disk Drive Control Data Area

Address (Hex)	Function	Size
40:93	Drive 3 media state	Byte
Bits 7, 6	Data transfer rate (values in binary) = 00 - 500KB per second = 01 - 300KB per second = 10 - 250KB per second = 11 - 1MB per second	
Bit 5	Double stepping = 1 - Double stepping required = 0 - Double stepping not required	
Bit 4	Media/drive established = 1 - Media/drive has been established = 0 - Media/drive has not been established	
Bit 3	4MB media capability = 1 - Drive is capable of supporting 4MB media = 0 - Drive is not capable of supporting 4MB media	
Bit 2	Multiple data rate capability determined = 1 - Multiple data rate capability is determined = 0 - Multiple data rate capability is not determined	
Bit 1	Multiple data rate capability = 1 - Multiple data rate capability = 0 - No multiple data rate capability	
Bit 0	80-track capability (drive 3) = 1 - Drive 3, 80 tracks = 0 - Drive 3, 40 tracks	
40:94	Drive 0 current cylinder	Byte
40:95	Drive 1 current cylinder	Byte
Exception: 40:8B to 40:95	Reserved (for PC, PC*jr*, PC/XT BIOS dated 11/8/82, and PC Convertible)	Byte

Figure 3-17 (Part 5 of 5). 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 set to 0)	
Bits 2 to 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 (in microseconds)	Word
40:9E	User wait count - high word (in microseconds)	Word
40:A0	Wait active flag	Byte
Bit 7	Wait time elapsed and Post	
Bits 6 to 1	Reserved	
Bit 0	Interrupt 15H, Wait function ((AH) = 86H) has occurred	
40:A1 to 40:A7	Reserved	Byte

Figure 3-19. Real-Time Clock Data Area

For Personal System/2 products and systems with EGA capability, the save pointer table contains pointers that define specific dynamic overrides for Interrupt 10H, Set Mode function ((AH) = 00H).

Address (Hex)	Function	Size
40:A8	Pointer to video parameters and overrides	DWord
DWord 1	Video parameter table pointer Initialized to the BIOS video parameter table; this value must contain a valid pointer.	
DWord 2	Dynamic save area pointer (for all systems except Personal System/2 Model 25 and Model 30) Initialized to hex 00:00, this value is optional. When nonzero, this value points to an area in RAM where certain dynamic values are saved. This area holds the 16 EGA palette register values plus the overscan value in bytes (0 16), respectively. A minimum of 256 bytes must be allocated for this area.	
DWord 3	Alphanumeric mode auxiliary character generator Initialized to hex 00:00, this value is optional. When nonzero, this value points to a table that is described as follows:	
	Bytes/character	Byte
	Block to be loaded = 0 - Normal operation	Byte
	Count to be stored = 256 - Normal operation	Word
	Character offset = 0 - Normal operation	Word
	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 might cause unexpected cursor type operation. For an explanation of cursor type, see Interrupt 10H, Set Cursor Type function ((AH) = 01H).

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

Address (Hex)	Function	Size
DWord 4	Graphics mode auxiliary character generator pointer Initialized to hex 00:00, this value is optional. When nonzero, this value points to a table that is described as follows:	
	Displayable rows	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 (for all systems except EGA and Personal System/2 Model 25 and Model 30) Initialized to the BIOS secondary save pointer; this value must contain a valid pointer.	
DWord 6	Reserved and set to hex 00:00	
DWord 7	Reserved and set to hex 00:00	

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

Address (Hex)	Function	Size
DWord 1	Table length Initialized to the BIOS secondary save pointer table length	
DWord 2	Display combination code (DCC) table pointer Initialized to ROM DCC table. This value must exist. It points to a table described as follows:	
	Number of entries in table	Byte
	DCC table version number	Byte
	Maximum display type code	Byte
	Reserved	Byte
	00,00 - Entry 0 No displays	
	00,01 - Entry 1 Monochrome display and printer adapter (MDPA)	
	00,02 - Entry 2 Color/graphics monitor adapter (CGA)	
	02,01 - Entry 3 MDPA + CGA	
	00,04 - Entry 4 Enhanced graphics adapter (EGA)	
	04,01 - Entry 5 EGA + MDPA	
	00,05 - Entry 6 EGA with monochrome display (MEGA)	
	02,05 - Entry 7 MEGA + CGA	
	00,06 - Entry 8 Professional graphics controller (PGC)	
	01,06 - Entry 9 PGC + MDPA	
	05,06 - Entry 10 PGC + MEGA	
	00,08 - Entry 11 Video graphics array (VGA) based with color display (CVGA)	
	01,08 - Entry 12 CVGA + MDPA	
	00,07 - Entry 13 VGA based with monochrome display (MVGA)	
	02,07 - Entry 14 MVGA + CGA	
	02,06 - Entry 15 MVGA + PGC	

Figure 3-21 (Part 1 of 2). Secondary Save Pointer Data Area

Address (Hex)	Function	Size
DWord 3	Second alphanumeric mode auxiliary character generator pointer Initialized to hex 00:00, this value is optional. When nonzero, this value points to a table that is described as follows:	
	Bytes/character	Byte
	Block to be loaded	Byte
	= 0 - Normal operation	
	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 might 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 nonzero, this value points to a table that is described as follows:	
	Underlining flag	Byte
	= 1 - On	
	= 0 - Ignore	
	= −1 - Off	
	= 0 - Normal operation	
	Reserved	Byte
	Reserved	Word
	Internal palette count (0 to 17)	Word
	= 17 - Normal operation	
	Internal palette index (0 to 16)	Word
	= 0 - Normal operation	
	Pointer to internal palette	DWord
	External palette count (0 to 256)	Word
	= 256 - Normal operation	
	External palette index (0 to 255)	Word
	= 0 - Normal operation	
	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 (set to hex 00:00)	

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

Address (Hex)	Function	Size
40:AC to 40:FF	Reserved	Byte
50:00	Print screen status (Interrupt 05H status)	Byte

Figure 3-22. Miscellaneous Data Area 2

Extended BIOS Data Areas

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

To access the extended BIOS data area segment, call Interrupt 15H, Return Extended BIOS Data Area Segment Address function ((AH) = C1H). To determine whether an extended BIOS data area is allocated, call Interrupt 15H, Return System Configuration Parameters function ((AH) = C0H).

Section 4. ROM Tables

The following ROM tables are used by BIOS to define the characteristics of the hardware devices that are supported by the system or adapter BIOS.

Fixed Disk Drive Parameter Table

The fixed disk drive parameter table is defined as follows:

Offset	Length	Description
0	Word	Maximum number of cylinders
2	Byte	Maximum number of heads
3	Word	For PC/XT: Starting reduced write current cylinder For all others: Not used
5	Word	Starting write precompensation cylinder
7	Byte	For PC/XT: Maximum ECC data-burst length For all others: Not used
8	Byte	Control byte For PC/XT: Bit 7 - Disable disk-access retries Bit 6 - Disable ECC retries Bits 5 to 3 = 0 Bits 2 to 0 - Drive option For all others: Bit 7 - Disable retries – or – Bit 6 - Disable retries Bit 5 - Manufacturer's defect map present at maximum cylinders + 1 Bit 3 - More than eight heads Bits 2 to 0 - Reserved
9	Byte	For PC/XT: Standard time-out value For all others: Not used
10	Byte	For PC/XT: Time-out value for format drive For all others: Not used
11	Byte	For PC/XT: Time-out value for check drive For all others: Not used
12	Word	For PC/XT: Reserved For all others: Landing zone
14	Byte	For PC/XT: Reserved For all others: Number of sectors per track
15	Byte	Reserved

Figure 4-1. Fixed Disk Drive Parameter Table

For AT and Personal System/2 products, the following table lists the fixed disk drive parameters for the various fixed disk drive types. Values are decimal unless noted otherwise.

Type	Number of Cylinders	Number of Heads	Number Write Precompensation	Landing Zone	Defect Map	Number of Sectors
0	– No fixed disk drive installed –					
1	306	4	128	305	No	17
2	615	4	300	615	No	17
3	615	6	300	615	No	17
4	940	8	512	940	No	17
5	940	6	512	940	No	17
6	615	4	0FFFFH (None)	615	No	17
7	462	8	256	511	No	17
8	733	5	0FFFFH (None)	733	No	17
9	900	15	0FFFFH (None)	901	No	17
10	820	3	0FFFFH (None)	820	No	17
11	855	5	0FFFFH (None)	855	No	17
12	855	7	0FFFFH (None)	855	No	17
13	306	8	128	319	No	17
14	733	7	0FFFFH (None)	733	No	17
15	– Reserved –					
16	612	4	0 (All cylinders)	663	No	17
17	977	5	300	977	No	17
18	977	7	0FFFFH (None)	977	No	17
19	1024	7	512	1023	No	17
20	733	5	300	732	No	17
21	733	7	300	732	No	17
22	733	5	300	733	No	17
23	306	4	0 (All cylinders)	336	No	17
24	612	4	305	663	No	17
25	306	4	0FFFFH (None)	340	No	17
26	612	4	0FFFFH (None)	670	No	17
27	698	7	300	732	Yes	17
28	976	5	488	977	Yes	17
29	306	4	0 (All cylinders)	340	No	17
30	611	4	306	663	Yes	17
31	732	7	300	732	Yes	17
32	1023	5	0FFFFH (None)	1023	Yes	17
33	614	4	0FFFFH (None)	663	Yes	25

Types 34 – 255 are reserved.

Figure 4-2. Fixed Disk Drive Parameters (for AT and Personal System/2 Products)

Notes:

1. Software Interrupt 41H points to the entry in the table for drive 0. Software Interrupt 46H points to the entry in the table for drive 1.
2. AT BIOS dated 1/10/84 supports types 0 through 14.
3. AT BIOS dated 6/10/85 or 11/15/85 supports types 0 through 23.
4. PC/XT Model 286 supports types 0 through 24.
5. Personal System/2 products except Model 25 and Model 30 support types 0 through 32.
6. Personal System/2 Model 30 supports types 0 through 26.
7. For Personal System/2 Model 70 and Model 80 BIOS dated 10/7/87 and later and self-identifying drives, the fixed disk drive parameters in Figure 4-2 on page 4-2 and Figure 4-3 do not apply; software Interrupts 41H and 46H are reserved.
8. To obtain the fixed disk drive parameters, use Interrupt 13H, Read Drive Parameters function ((AH) = 08H).

For Personal System/2 products except Model 25 and Model 30, the following fixed disk drive parameters apply:

Offset	Length	Value	Description
0	2 bytes	41	Length of fixed disk drive table
2	22 bytes	(ID)	ASCII string 'IBM HARDFILE TYPE xxx', where xxx is the type number, in ASCII
24	1 byte	yyy	Type number (in binary)
25	2 bytes	*	Maximum number of cylinders
27	1 byte	*	Maximum number of heads
28	2 bytes	0	Reserved
30	2 bytes	*	Start write precompensation cylinder
32	1 byte	0	Reserved
33	1 byte	*	Control byte Bit 7 or 6 - Disable retries Bit 5 - Defect map installed Bit 3 - More than eight heads (AT only)
34	3 bytes	0	Reserved
37	2 bytes	*	Landing zone
39	1 byte	*	Number of sectors per track
40	1 byte	0	Reserved

Note: This information is located at head 0, track 0, sector 2; it applies only to ST412 and ST506 drive types.

Figure 4-3. Fixed Disk Drive Parameters (for Personal System/2 Products except Model 25 and Model 30)

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

Size	Value	Description
DW	306	Maximum cylinders
DB	2	Maximum heads
DW	306	Start reduced write current cylinder
DW	0	Start write precompensation cylinder
DB	0BH	Maximum ECC burst data length
DB	00H	Control byte
DB	0CH	Standard time-out
DB	0B4H	Time-out for format drive
DB	028H	Time-out for check drive
DB	0,0,0,0	Reserved

Figure 4-4. Fixed Disk Drive Parameter Table 00 (for PC/XT BIOS Dated 11/10/82)

Size	Value	Description
DW	375	Maximum cylinders
DB	8	Maximum heads
DW	375	Start reduced write current cylinder
DW	0	Start write precompensation cylinder
DB	0BH	Maximum ECC burst data length
DB	05H	Control byte
DB	0CH	Standard time-out
DB	0B4H	Time-out for format drive
DB	028H	Time-out for check drive
DB	0,0,0,0	Reserved

Figure 4-5. Fixed Disk Drive Parameter Table 01 (for PC/XT BIOS Dated 11/10/82)

Size	Value	Description
DW	306	Maximum cylinders
DB	6	Maximum heads
DW	128	Start reduced write current cylinder
DW	256	Start write precompensation cylinder
DB	0BH	Maximum ECC burst data length
DB	05H	Control byte
DB	0CH	Standard time-out
DB	0B4H	Time-out for format drive
DB	028H	Time-out for check drive
DB	0,0,0,0	Reserved

Figure 4-6. Fixed Disk Drive Parameter Table 02 (for PC/XT BIOS Dated 11/10/82)

Size	Value	Description
DW	306	Maximum cylinders
DB	4	Maximum heads
DW	306	Start reduced write current cylinder
DW	0	Start write precompensation cylinder
DB	0BH	Maximum ECC burst data length
DB	05H	Control byte
DB	0CH	Standard time-out
DB	0B4H	Time-out for format drive
DB	028H	Time-out for check drive
DB	0,0,0,0	Reserved

Figure 4-7. Fixed Disk Drive Parameter Table 03 (for PC/XT BIOS Dated 11/10/82)

Note: Software Interrupt 41H points to the beginning of the table. The switch settings on the adapter are used as an index into the table.

For PC/XT BIOS dated 1/8/86 and later, the following fixed disk drive parameters apply:

Size	Value	Description
DW	306	Maximum cylinders
DB	4	Maximum heads
DW	306	Start reduced write current cylinder
DW	0	Start write precompensation cylinder
DB	0BH	Maximum ECC burst data length
DB	05H	Control byte
DB	0CH	Standard time-out
DB	0B4H	Time-out for format drive
DB	028H	Time-out for check drive
DB	0,0,0,0	Reserved

Figure 4-8. Fixed Disk Drive Parameter Table 00 – Type 1 (for PC/XT BIOS Dated 1/8/86)

Size	Value	Description
DW	612	Maximum cylinders
DB	4	Maximum heads
DW	612	Start reduced write current cylinder
DW	0	Start write precompensation cylinder
DB	0BH	Maximum ECC burst data length
DB	05H	Control byte
DB	20H	Standard time-out
DB	0B4H	Time-out for format drive
DB	028H	Time-out for check drive
DB	0,0,0,0	Reserved

Figure 4-9. Fixed Disk Drive Parameter Table 01 – Type 16 (for PC/XT BIOS Dated 1/8/86)

Size	Value	Description
DW	615	Maximum cylinders
DB	4	Maximum heads
DW	615	Start reduced write current cylinder
DW	300	Start write precompensation cylinder
DB	0BH	Maximum ECC burst data length
DB	05H	Control byte
DB	18H	Standard time-out
DB	0B4H	Time-out for format drive
DB	028H	Time-out for check drive
DB	0,0,0,0	Reserved

Figure 4-10. Fixed Disk Drive Parameter Table 02 – Type 2 (for PC/XT BIOS Dated 1/8/86)

Size	Value	Description
DW	306	Maximum cylinders
DB	8	Maximum heads
DW	306	Start reduced write current cylinder
DW	128	Start write precompensation cylinder
DB	0BH	Maximum ECC burst data length
DB	05H	Control byte
DB	0CH	Standard time-out
DB	0B4H	Time-out for format drive
DB	028H	Time-out for check drive
DB	0,0,0,0	Reserved

Figure 4-11. Fixed Disk Drive Parameter Table 03 – Type 13 (for PC/XT BIOS Dated 1/8/86)

Note: Software Interrupt 41H points to the beginning of the table. The switch settings on the adapter are used as an index into the table.

Diskette Drive Parameter Table

The diskette drive parameter table is defined as follows:

Offset	Length	Description
0	Byte	First specification byte
1	Byte	Second specification byte
2	Byte	Number of timer ticks to wait before turning the diskette drive motor off
3	Byte	Number of bytes per sector = 02H - 512 bytes per sector
4	Byte	Sectors per track
5	Byte	Gap length
6	Byte	Dtl (data length)
7	Byte	Gap length for format
8	Byte	Fill byte for format
9	Byte	Head settle time (in milliseconds)
10	Byte	Motor startup time (in ⅛-second units) (For example, 8 = 1-second wait)

Figure 4-12. Diskette Drive Parameter Table

Note: Software Interrupt 1EH points to the beginning of the diskette drive parameter table.

If the caller changes the values of the head settle time (byte 9) and the motor startup time (byte 10) to values that are inconsistent with the diskette drive specifications, BIOS enforces the minimum values for these parameters as specified for the diskette drive. The values of these parameters can be increased to allow for correcting possible future problems if some diskette drives require more than the nominal values for these parameters.

Section 5. Additional Information

Notes:

Interrupt Sharing

This section defines an interrupt-sharing protocol that enables multiple hardware adapters to share a single interrupt-request line.

Considerations

When implementing interrupt sharing, consider the following:

- The interrupt-sharing protocol is intended to run only in the real address mode. It is not intended to run in the protected (virtual address) mode.
- The interrupt-sharing protocol does not apply to the sharing of an interrupt level between an interrupt handler that is running in the real mode and an interrupt handler that is running in the protected mode.
- The interrupt-sharing protocol is not necessarily compatible with all operating systems.
- Interrupts must be disabled before control is passed to the next handler on the chain. Disabling of the interrupts enables the next handler to receive control as if a hardware interrupt had caused it to receive control.
- Interrupts must be disabled before the nonspecific End of Interrupt (EOI) instruction is issued. To ensure that the Return from Interrupt (IRET) instruction is executed, interrupts must not be reenabled in the interrupt handler. The flags are restored and the interrupts are reenabled before another interrupt is serviced, protecting the stack from excessive build-up.
- Each interrupt handler must have a routine that can be executed after power-on to disable its adapters' interrupts. Executing this routine and resetting the interrupt-sharing hardware ensures that adapters are deactivated if the user resets the system.
- Interrupt-handler implementations must store data in memory using Intel** data format; that is, word hex 424B is stored as 4BH,42H in memory.

** Intel is a trademark of Intel Corporation.

Interrupt Request (IRQ n) Reset

The Micro Channel interrupt mechanism is level sensitive, whereas the PC-type I/O channel interrupt mechanism is edge sensitive. The level-sensitive Micro Channel mechanism simplifies the interrupt hardware that is needed for the adapters.

An interrupt request in the PC-type I/O channel is implicitly reset because of the edge-sensitive characteristic of the signal. In the Micro Channel architecture, because of the level-sensitive characteristic of the signal, an interrupt request must be explicitly reset by the bus-slave interrupt-handler software. This is not the case when the bus-slave hardware implicitly resets the interrupt request. The system timer is an example of a bus-slave device that implicitly resets an interrupt.

Interrupt-Sharing Software Requirements

All interrupt-sharing software that is developed for Micro Channel bus slaves must reset the interrupt request. The interrupt-sharing chaining structure must be provided by all interrupt handlers. The 16-byte interrupt-sharing chaining structure must begin at the third byte from the entry point of the interrupt handler. Pointers and flags that are stored in the interrupt-sharing chaining structure must be stored in Intel data format (see "Interrupt-Sharing Chaining Structure and Signature" on page 5-6). These requirements are specified to support the portability of interrupt handlers across hardware operating environments.

The interrupt-handling software for all adapters that share an interrupt-request line must implement this interrupt-sharing software standard. Interrupt-sharing software that operates in a multitasking environment must support the linking of a task's interrupt handler to a chain of interrupt handlers; the sharing of the interrupt level while that task is active; and the unlinking of the interrupt handler from the chain when the task is complete.

To link an interrupt handler, the interrupt handler of a newly-activated task replaces the interrupt vector in low memory with a pointer to the interrupt handler. (See "ROM Considerations" on page 5-7 for information about interrupt handlers that are stored in ROM.) The interrupt handler must preserve the interrupt vector that it is replacing and use it as a forward pointer to the next interrupt handler in the chain. The old interrupt vector must be stored at a fixed offset from the entry point of the new task's interrupt handler.

When the system acknowledges an interrupt request, each interrupt handler must determine whether it is the appropriate interrupt handler for the adapter that is presenting the interrupt request. To make this determination, the interrupt handler reads the contents of the interrupt-status register of the adapter.

When an interrupt handler determines that its device caused the interrupt, the interrupt handler must service the interrupt, reset the interrupt-status bit, clear the interrupts, issue a nonspecific EOI instruction to the interrupt controller, then execute an IRET instruction.

When an interrupt handler determines that its device did not cause the interrupt, the interrupt handler passes control to the next interrupt handler in the chain, using the previously-stored forward pointer.

To unlink an interrupt handler from a chain, a task first locates the position of its interrupt handler in the chain. The task searches the chain, starting at the interrupt vector in low memory, and using the offset of each interrupt handler's forward pointer to locate the entry point of each interrupt handler. The task repeats this process until it finds its own interrupt handler. The signature (hex 424B) of each interrupt handler must be checked to ensure that a valid forward pointer exists. The forward pointer of the task replaces the forward pointer of the previous interrupt handler in the chain, thus removing the interrupt handler from the chain.

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

An application-dependent unlinking error-recovery procedure must be incorporated into the unlinking routine for situations in which the unlinking routine determines that the interrupt chain contains an interrupt handler that is linked but does not have a valid signature. To avoid this error condition, all interrupt-sharing handlers, except those in ROM (see "ROM Considerations" on page 5-7), must use hex 424B as their signature.

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

Operating-system environments that support dynamic relocation of software must manage the entire interrupt-sharing process. Interrupt-handler software that is written exclusively for dynamic-relocation operating-system environments does not have to provide an interrupt-sharing chaining structure. These interrupt

handlers do not have to provide linking and unlinking support, but they must provide support for disabling the interrupt capability of the bus slave that they support.

Interrupt-Sharing Chaining Structure and Signature

The interrupt-sharing software chaining structure has a 16-byte format that contains a 4-byte forward pointer (FPTR), a 2-byte signature, and 8 reserved bytes (RES_BYTES), as shown in the following example:

```
ENTRY:  JMP          SHORT   PAST       ; Jump around structure
        FPTR         DD      0          ; Forward pointer
        SIGNATURE    DW      424BH      ; Used when unlinking to identify
                                        ;  compatible interrupt handlers
        FLAGS        DB      0          ; Flags
        FIRST        EQU     80H        ; Flag for being first in chain
        JMP          SHORT   RESET
        RES_BYTES    DB      DUP 7(0)   ; -Reserved-
PAST:   ...                             ; Actual start of code
```

The interrupt-sharing software chaining structure begins at the third byte from the entry point of the interrupt handler. The first instruction of each interrupt handler is a short jump around the structure, placing the structure at a known offset from the beginning of the interrupt-handler routine. Because the position of the chaining structure of each interrupt handler is known (except for the interrupt handlers on adapter ROM), the forward pointers can be updated during linking and unlinking.

The FIRST flag is used to determine the position of the interrupt handler in the chain during linking and unlinking for shared interrupt levels. The value of the FLAGS byte is changed to the value of the FIRST flag (hex 80) to indicate that the interrupt handler is the first interrupt handler that is linked in the chain. Each interrupt handler that is not stored in ROM must store the FIRST flag (hex 80) in the FLAGS byte when it is the first interrupt handler in the chain.

The Reset routine, an entry point for the operating system, must disable the adapter interrupt and return to the operating system.

ROM Considerations

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

Implementation Information

The Interrupt Mask register is located at I/O port hex 21. Specific End of Interrupt (EOI) values for the various interrupt levels are listed (hex 67 for level 7). To execute a specific EOI, issue an OUT instruction to the Programmable Interrupt Controller Operation Command register (port hex 20), using operation-command byte 2. To execute a nonspecific EOI, issue an OUT value of hex 20 to the Programmable Interrupt Controller Operation Command register (port hex 20).

The following are examples of code that implements interrupt sharing:

Linking

```
          PUSH      ES
          CLI                         ;Clear interrupts
;Set forward pointer to value of interrupt vector in low memory
          ASSUME    CS:CODESEG,DS:CODESEG
          PUSH      ES
          MOV       AX,350FH          ;DOS get interrupt vector
          INT       21H
          MOV       SI,OFFSET CS:FPTR ;Get offset of your forward pointer
                                      ; in an indexable register
          MOV       CS:[SI],BX        ;Store the old interrupt vector
          MOV       CS:[SI+2],ES      ; in your forward pointer for
                                      ; chaining
          CMP       ES:BYTE PTR[BX],0CFH  ;Test for IRET
   if iret_test_only_is_needed   ; See NOTE below
          JNE       SETVECTR
   else
          JE        FRSTVCTR
          CMP       ES:WORD PTR[BX+6],424BH ; Is signature present?
          JE        SETVECTR
          MOV       AX,ES
```

```
            CMP       AX,0F000H          ;See if pointing to dummy handler
            JNE       SETVCTR
            CMP       BX,WORD PTR ES:[0FF01H] ; Dummy vector pointer?
            JNE       SETVECTR           ;If dummy, then first
FRSTVCTR:
   endif
            MOV       CS:FLAGS,FIRST     ;Set up first in chain flag
SETVECTR:   POP       ES
            PUSH      DS
;Make interrupt vector in low memory point to your handler
            MOV       DX,OFFSET ENTRY    ;Make interrupt vector point to
                                         ; your handler
            MOV       AX,SEG ENTRY       ;If DS ≠ CS, get it and
            MOV       DS,AX              ; put it in DS
            MOV       AX,250FH           ;DOS set interrupt vector
            INT       21H
            POP       DS
;Unmask (enable) interrupts for your level
            IN        AL,IMR             ;Read interrupt mask register
            JMP       $+2                ;I/O delay
            AND       AL,07FH            ;Unmask interrupt level 7
            OUT       IMR,AL             ;Write new interrupt mask
            MOV       AL,SPC_EOI         ;Issue specific EOI for level 7
            JMP       $+2                ; to allow pending level 7 interrupts
            OUT       OCR,AL             ; (if any) to be serviced
            STI                          ;Enable interrupts
            POP       ES
```

Notes:

1. The operating system must ensure that SEG:OFF points to a valid interrupt handler or to an IRET (hex 0CF) for levels 7 and 15.
2. Each ROM interrupt handler during ROMSCAN (before the operating system is loaded) and each interrupt handler on other than IRQ 7 must test SEG:OFF as shown in the "else" clause in this listing to determine whether it is the first interrupt handler in the chain. Checking SEG:OFF to determine whether it points to an IRET as the sole determination of FIRST is allowed only on IRQ 7, and then only after the operating system is loaded.

Interrupt Handler

```
YOUR_CARD   EQU       xxxx               ;Location of your card interrupt
                                         ; control/status register
ISB         EQU       xx                 ;Interrupt bit in your card
                                         ; 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                ;Nonspecific EOI
OCR         EQU       20H                ;Location of programmable interrupt
                                         ; controller operation command register
```

```
IMR         EQU       21H               ;Location of programmable interrupt
                                        ; controller interrupt mask
MYCSEG      SEGMENT   PARA
            ASSUME    CS:MYCSEG,DS:DSEG
ENTRY       PROC      FAR
            JMP       SHORT PAST        ;Entry point of handler
FPTR        DD        0                 ;Forward pointer
SIGNATURE   DW        424BH             ;Used when unlinking to identify
                                        ; compatible interrupt handlers
FLAGS       DB        0                 ;Flags
FIRST       EQU       80H
JMP         SHORT     RESET
RES_BYTES   DB        7 DUP (0)         ;Future expansion
PAST:       STI                         ;Actual start of handler code
            PUSH      ...               ;Save needed registers
            MOV       DX,YOUR_CARD      ;Select your status register
            IN        AL,DX             ;Read the status register
            TEST      AL,ISB            ;Your card caused interrupt?
            JNZ       SERVICE           ;Yes, branch to service logic
            TEST      CS:FLAGS,FIRST    ;Are we the first ones in?
            JNZ       EXIT              ;If yes, branch for EOI and Rearm
            POP       ...               ;Restore registers
            CLI                         ;Clear interrupts
            JMP       DWORD PTR CS:FPTR ;Pass control to next handler on chain
SERVICE:    ...                         ;Service the interrupt
EXIT:
            CLI                         ;Clear interrupts
            MOV       AL,EOI
            OUT       OCR,AL            ;Issue nonspecific EOI to programmable
                                        ; interrupt controller
            MOV       DX,REARM          ;Rearm the cards
            OUT       DX,AL
            POP       ...               ;Restore registers
            IRET
RESET:      ...                         ;Disable your card
            RET                         ;Return FAR to operating system
ENTRY       ENDP
MYCSEG      ENDS
            END       ENTRY
```

Unlinking

```
            PUSH      DS
            PUSH      ES
            CLI                         ;Clear interrupts
            MOV       AX,350FH          ;DOS get interrupt vector
            INT       21H               ;ES:BX points to first of chain
            MOV       CX,ES             ;Pick up segment part of interrupt vector
;Are we the first handler in the chain?
            MOV       AX,CS             ;Get code seg into comparable register
            CMP       BX,OFFSET ENTRY   ;Interrupt vector in low memory
                                        ; pointing to your handler offset?
            JNE       UNCHAIN_A         ;No, branch
            CMP       AX,CX             ;Vector pointing to your handler
                                        ; segment?
            JNE       UNCHAIN_A         ;No, branch
```

```
;Set interrupt vector in low memory to point to the handler pointed to
; by your pointer
            PUSH      DS
            MOV       DX,WORD PTR CS:FPTR
            MOV       DS,WORD PTR CS:FPTR[2]
            MOV       AX,250FH          ;DOS set interrupt vector
            INT       21H
            POP       DS
            JMP       UNCHAIN_X
UNCHAIN_A:     ; BX = FPTR offset, ES = FPTR segment, CX = CS
            CMP       ES:[BX+6],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

BIOS provides a method for integrating adapters with on-board ROM code into the system. During power-on self-test (POST), interrupt vectors are established for BIOS calls. When the default vectors are in place, a scan for adapter-ROM modules occurs. At this point, an adapter-ROM routine can gain control. The routine can establish or intercept interrupt vectors to hook into the system.

Early in POST, the absolute addresses hex C0000 through hex C7FFF are scanned in 2KB blocks to search for adapter-ROM modules that need to be initialized (for example, valid video adapter ROM).

Later in POST, the absolute addresses hex C8000 through hex DFFFF are scanned in 2KB blocks to search for devices with valid adapter-ROM modules. Valid adapter ROM is defined as follows:

Byte 0: Hex 55

Byte 1: Hex AA

Byte 2: A length indicator, representing the number of 512-byte blocks (the limit is hex 7F) in the ROM (length divided by 512). A checksum tests the integrity of the ROM module. Each byte in the defined ROM is summed modulo hex 100. This sum must be 0 for the module to be valid.

When POST identifies valid adapter ROM, it executes a far call to byte 3 of the ROM (which should contain executable code). The device can now perform power-on initialization. The adapter ROM should return control to POST by executing a far return.

For the PC Convertible, if the adapter ROM detects a self-test error, it should perform the following steps before returning:

- Set bit 4 of hex 40:12 (POST status) to 1.
- Set the device number for the supported adapter in (AH).
- Set a two-digit error code in (AL).

If no self-test error is found, the adapter ROM should reset bit 4 of hex 40:12 (POST status) to 0 before returning.

For Personal System/2 products with Micro Channel architecture, a video adapter in the channel has a ROM signature code that identifies it. During POST, when CMOS is not valid (abnormal condition), the signature code is used to find the first video adapter and set up its ROM programmable option select (POS) parameters.

The code starts at hex 0C in the ROM address space and consists of:

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

The POS parameters are accessed from offset hex 30 in the ROM address space and are in the following order:

```
POS Byte 102, POS Byte 103, POS Byte 104, POS Byte 105
```

Video ROM scan remains from hex C0000 to hex C7FFF.

For Personal System/2 BIOS dated 10/7/87 and later, the adapter-ROM integration method is as follows:

Early in POST, the absolute addresses hex C0000 through DFFFF are scanned in 2KB blocks to search for adapter-ROM modules that have video-adapter signature code, as previously described. Only adapters with a video signature code are initialized early in POST.

Later in POST, the absolute addresses hex C0000 through DFFFF are scanned in 2KB blocks to initialize other adapter-ROM modules that were not initialized in the early scan.

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

On return from a call to a video-adapter-ROM module, (BH) indicates the following:

```
(BH) = 00H - Not a video adapter
     = 02H - Video adapter supporting video in the
             color/graphics adapter range
     = 04H - Video adapter supporting video in the
             monochrome adapter range
```

Video-Function Compatibility

The following procedures are recommended to provide video-function compatibility for application software.

Video-Presence Test

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

1. Issue Interrupt 10H, Read/Write Display-Combination Code function with ((AH) = 1AH), Read Display-Combination Code ((AL) = 00H).

   On return, if the value of (AL) is not hex 1A, the Read/Write Display-Combination Code function is not supported, and step 2 should be followed to determine video presence.

   On return, if the value of (AL) is hex 1A, the information that is returned in (BX) defines the video environment. The active display code is returned in (BL). The alternative display code, if any, is returned in (BH). Refer to Interrupt 10H, Read/Write Display-Combination Code function ((AH) = 1AH) for display-code definitions.

2. To determine the presence of an IBM Enhanced Graphics Adapter (EGA) when the Read/Write Display-Combination Code function is not supported, issue Interrupt 10H, Alternative Selection function ((AH) = 12H), Return EGA Information ((BL) = 10H).

   On return, if the value of (BL) is hex 10, an EGA is not present, and step 3 should be followed.

   On return, if the value of (BL) is not hex 10, an EGA is present. Note that an IBM Color/Graphics Monitor Adapter or an IBM Monochrome Display and Printer Adapter might also be present, depending on the EGA switch settings.

3. Complete steps 1 and 2 before performing this step. The video functions that might be present at this point are the IBM Color/Graphics Monitor Adapter, the IBM Monochrome Display and Printer Adapter, or both. Perform a presence test on video buffer addresses hex 0B8000 and hex 0B0000 to determine which video functions are present.

Video-Mode Switching

Use the following video-mode switching procedure when applications will switch between monochrome and color video modes. A correct video-presence test, as previously described, is required. The following three system-video environments are possible:

1. A single video function that supports either monochrome or color video modes. If a monochrome function is present, only monochrome video modes are available. If a color function is present, only color video modes are available.
2. Two video functions, one supporting color video modes, and the other supporting monochrome video modes. In this case, both monochrome and color video modes are available. To switch from monochrome to color or from color to monochrome, the application program must change bits 5 and 4 (video mode type) of data area hex 40:10 to monochrome or color and issue Interrupt 10H, Set Mode function ((AH) = 00H).
3. A single video function that supports both monochrome and color video modes. To determine whether a single video function supports both monochrome and color video modes, the application program should issue Interrupt 10H, Return Functionality/State Information function ((AH) = 1BH).

   On return, if the value of (AL) is not hex 1B, the Return Functionality/State Information function is not supported, and support for both monochrome and color video modes on a single video function is not available.

   On return, if the value of (AL) is hex 1BH, use the returned information to determine whether bit 0 (all modes on all displays active) of byte (DI + 2DH) is set to 1. If it is set to 1, both monochrome and color modes are available, and the application program should change the video-modes bits for monochrome or color and issue Interrupt 10H, Set Mode function ((AH) = 00H). If it is set to 0, only monochrome modes or color modes are available, depending on the results of the video-presence test.

Multitasking Provisions

BIOS provides hooks to assist in multitasking information. Whenever a Busy (Wait) loop occurs in BIOS, a hook is provided for the program to break out of the loop. Also, when BIOS services an interrupt, a corresponding Wait loop is ended, and another hook is provided. A program can be written that employs most of the device-driver code. The following information is valid only in the microprocessor real-address mode, and the code must use the following procedures to enable this support.

The program is responsible for matching corresponding Wait and Post calls and for the serialization of access to the device driver. The BIOS code is not reentrant.

The multitasking dispatcher uses the following four interfaces:

Startup: The startup code hooks Interrupt 15H. The dispatcher is responsible for checking for function codes (AH) = 90H and (AH) = 91H (see the following descriptions of Wait and Post). The dispatcher must pass all other functions to the previous user of Interrupt 15H (use a JMP or CALL instruction). If the value of (AH) is hex 90 or hex 91, the dispatcher must perform the appropriate processing and return by the IRET instruction.

Serialization: The multitasking system must ensure that the device driver code is used serially. Multiple entries into the code can result in errors.

Wait (Busy): When BIOS is about to enter a Wait loop, it first issues Interrupt 15H, (AH) = 90H. This signals a Wait condition. At this point, the dispatcher should save the task status and dispatch another task. This enables overlapped execution of tasks when the hardware is busy. The following is an outline of the code that has been added to BIOS to perform this function.

```
MOV AX, 90xxH          ;Wait code in AH and
                       ; type code in AL
INT 15H                ;Issue call
JC  TIMEOUT            ;Optional: for time-out or
                       ; if carry is set, time-out
                       ; occurred
NORMAL TIMEOUT LOGIC   ;Normal time-out
```

Post (Interrupt): When BIOS has set an interrupt flag for a corresponding Busy loop, it issues Interrupt 15H, (AH)=91H. This signals a Post condition. At this point, the dispatcher must set the task status to "ready to run" and return to the interrupt routine. The following is an outline of the code that has been added to BIOS to perform this function.

```
MOV AX, 91xxH          ;Post code AH and
                       ; type code AL
INT 15H                ;Issue call
```

Three Wait loop function-code classes are supported:

- The first (hex 0 to hex 7F) is serially reusable. This means that for devices that use these codes, access to BIOS must be restricted to one task at a time, and the operating system must serialize access.
- The second (hex 80 to hex BF) is for reentrant devices. There is no restriction on the number of tasks that can use a device. (ES:BX) is used to distinguish between different calls.
- The third (hex C0 to hex FF) is noninterrupt (wait-only calls). There is no corresponding interrupt for the Wait loop. The dispatcher must take the appropriate action to satisfy this condition and exit from the loop. There is no complementary Post for these Wait conditions. They are time-out only, and the times are function-number dependent.

To support time-outs properly, the multitasking dispatcher must be aware of time. If a device enters a Busy loop, generally, it should remain there for a specific length of time before indicating an error. The dispatcher must return to the BIOS Wait loop with the carry flag set to 1 if a time-out occurs.

Application Guidelines

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

Math-Coprocessor Testing

Interrupt 11H (Equipment Determination) should be used where possible as the method for detecting the presence of the math coprocessor.

Hardware Interrupts

Hardware interrupts are level sensitive for systems that use the Micro Channel architecture; hardware interrupts are edge triggered for systems that use the PC-type I/O channel. In edge-triggered-interrupt systems, the interrupt controller clears its internal 'interrupt in progress' latch when the interrupt routine sends an End of Interrupt (EOI) command to the controller. The EOI command is sent, regardless of whether the incoming interrupt request to the controller is active or inactive.

In level-sensitive-interrupt systems, the 'interrupt in progress' latch is readable at an I/O-address bit position. This latch is read during the Interrupt Service routine; it can be reset by the read operation, or it might require an explicit reset.

Note: For performance and latency considerations, designers might want to limit the number of devices that share an interrupt level.

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

To avoid this problem, a level-sensitive interrupt handler must clear the interrupt condition, usually by a Read or Write command to an I/O port on the device that is causing the interrupt.

Because of the speed of the processor, a delay must be added after every repetitive I/O operation to a controller or device to ensure that

I/O operations are synchronized with the processor. The JMP $+2 instruction is not recommended for this purpose.

Before the interrupt controllers are programmed, interrupts should be disabled through a Clear Interrupt Flag (CLI) instruction. This includes programming of the Mask register and issuing EOI instructions, initialization-command bytes, and operation-command bytes.

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

Hardware interrupt IRQ 9 is defined as the replacement interrupt level for the cascade level IRQ 2. Program interrupt sharing should be implemented on IRQ 2, Interrupt 0AH. The following processing occurs to maintain compatibility with the IRQ 2 that is used by IBM Personal Computer products:

1. A device drives the interrupt request that is active on IRQ 2 of the channel.
2. This interrupt request is mapped in hardware to IRQ 9 input on the slave interrupt controller.
3. When the interrupt occurs, the system microprocessor passes control to the IRQ 9 (Interrupt 71H) interrupt handler.
4. This interrupt handler sends an EOI command to the slave interrupt controller and passes control to the IRQ 2 (Interrupt 0AH) interrupt handler.
5. When handling the interrupt, the IRQ 2 interrupt handler causes the device to reset the interrupt request before sending an EOI command to the master interrupt controller that finishes servicing the IRQ 2 request.

Programming Considerations

The IBM-supported languages of IBM C, BASIC, FORTRAN, COBOL, and Pascal are the best choices for writing compatible programs. If a program uses specific features of the hardware, the program might not be compatible with all IBM Personal System/2 and Personal Computer products.

Any program that requires precise timing information should obtain it through an operating system or language interface (for example, TIME$ in BASIC). The use of programming loops can prevent a

program from being compatible with other Personal System/2 and IBM Personal Computer products and software.

BIOS and Operating-System Function Calls

For maximum portability, programs should perform all I/O operations through operating-system function calls. In environments where the operating system does not provide the necessary programming interfaces, programs should access the hardware through BIOS function calls, if it is permissible. When writing programs, consider the following:

- In some environments, program interrupts are used for access to these functions. This practice removes the absolute addressing from the program. Only the interrupt number is required.
- The system can mask hardware sensitivity. New devices can change BIOS to accept the same programming interface on the new device.
- When BIOS provides parameter tables, such as for video or diskette, a program can substitute new parameter values by building a new copy of the table and changing the vector to point to that table. The program should copy the current table, using the current vector, and then modify those locations in the table that need to be changed. In this way, the program does not inadvertently change any values that should be left the same.
- The diskette parameters table that is pointed to by Interrupt 1EH consists of 11 parameters that are required for diskette operation. It is recommended that the values that are supplied in ROM be used. If it becomes necessary to modify any of the parameters, build another parameter block, and modify the address at Interrupt 1EH (hex 00:78) to point to the new block.

  The parameters were established to allow:

  - Some models of the IBM Personal Computer to operate both the 5.25-inch high-capacity diskette drive (96 tracks per inch) and the 5.25-inch double-sided diskette drive (48 tracks per inch)
  - Some models of the Personal System/2 system to operate both the 3.5-inch 1.44MB diskette drive and the 3.5-inch 720KB diskette drive.

  The Gap Length parameter is not always retrieved from the parameter block. The gap length that is used during diskette read, write, and verify operations is derived from within diskette

BIOS. The gap length for format operations is still obtained from the parameter block.

If a parameter block contains a Head Settle Time parameter value of 0 milliseconds, BIOS enforces a minimum head settle time of 15 milliseconds. Read and verify operations use the head settle time that is provided by the parameter block.

For any function that requires a parameter block that contains a Motor Start Wait parameter of less than 500 milliseconds (1 second for a Personal Computer product), diskette BIOS enforces a minimum time of 500 milliseconds (1 second for a Personal Computer product).

- Programs can be designed to reside on both 5.25-inch and 3.5-inch diskettes. Because not all programs are operating-system dependent, the following procedure can be used to determine which type of media is in a diskette drive:

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

     If the verify operation fails, issue the Reset function ((AH) = 00H) to diskette BIOS and try the operation again. If it fails again, the media needs to be formatted or is defective.

  2. Verify track 0, head 0, sector 16 (1 sector).

     If the verify operation fails, either a 5.25-inch (48 TPI) or a 3.5-inch 720KB diskette is installed. To determine the type, verify track 78, head 1, sector 1 (1 sector). A successful verification of track 78 indicates that a 3.5-inch 720KB diskette is installed; a failed verification indicates that a 5.25-inch (48 TPI) diskette is installed.

     Note: Refer to the *DOS Technical Reference* for the file-allocation-table parameters for single-sided and double-sided diskettes.

  3. Read the diskette-controller status in BIOS, starting with address hex 40:42. The fifth byte defines the head that the operation ended with. If the operation ended with head 1, the diskette is a 5.25-inch high-capacity (96 TPI) diskette; if the operation ended with head 0, the diskette is a 3.5-inch 1.44MB diskette.

  4. Newer Personal System/2 products might support Interrupt 13H—Diskette, Get Media Type function ((AH) = 20H).

Section 6. System Identification

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

A model byte changes when there are major processing-unit architecture changes. For example, 80286-based systems have model byte hex FC and have the following additional features:

- Protected virtual address mode
- 24 bits of addressing.

The 80386- and 80486-based systems have model byte hex F8 and have the following additional features:

- 32-bit registers
- 32-bit addressing
- 32-bit flat model mode
- Virtual 8086 mode.

A submodel byte changes when a system implements a software-detectable change from previous systems with the same model byte.

A BIOS revision level changes when the ROM image changes for a model or submodel.

The following table lists the system-identification information for Personal Computer and Personal System/2 products.

Product	BIOS Date	Model Byte	Submodel Byte	Revision
PC	4/24/81	FF	00	00
PC	10/19/81	FF	00	01
PC	10/27/82	FF	00	02
PC/XT	11/8/82	FE	00	00
PC/XT	1/10/86	FB	00	01
PC/XT	5/9/86	FB	00	02
PC*jr*	6/1/83	FD	00	00

Figure 6-1 (Part 1 of 3). System Identification

Product	BIOS Date	Model Byte	Submodel Byte	Revision
AT	1/10/84	FC	00	00
AT	6/10/85	FC	00	01
AT	11/15/85	FC	01	00
PC/XT Model 286	4/21/86	FC	02	00
PC Convertible	9/13/85	F9	00	00
PS/2 Model 25	6/26/87	FA	01	00
PS/2 Model 25	11/2/88	FA	01	01
PS/2 Model 30	9/2/86	FA	00	00
PS/2 Model 30	1/31/89	FA	00	04
PS/2 Model 30 286	8/25/88	FC	09	00
PS/2 Model 30 286	11/30/88	FC	09	01
PS/2 Model 30 286	5/30/89	FC	09	02
PS/2 Model 30 286		FC	09	03
PS/2 Model 40 SX/35 SX	3/15/91	F8	19	05
PS/2 Model 40 SX/35 SX	4/4/91	F8	19	06
PS/2 Model 40 SX/35 SX	6/4/91	F8	19	07
PS/2 Model L40 SX		F8	23	00
PS/2 Model 50 (Type 1)	2/13/87	FC	04	00
PS/2 Model 50		FC	04	01
PS/2 Model 50 (Type 1)	11/2/89	FC	04	02
PS/2 Model 50 (Type 2)	1/28/88	FC	04	03
PS/2 Model 50	5/12/88	FC	04	04
PS/2 Model 55 SX	11/2/88	F8	0C	00
PS/2 Model 55 SX		F8	0C	01
PS/2 Model 55 LS	2/8/90	F8	1E	00
PS/2 Model 57 SX	5/10/91	F8	26	00
PS/2 Model 60	2/13/87	FC	05	00
PS/2 Model 65 SX	2/8/90	F8	1C	00
PS/2 Model 70 (Type 2)	4/11/88	F8	04	02
PS/2 Model 70	3/17/89	F8	04	03
PS/2 Model 70	12/15/89	F8	04	04
PS/2 Model 70 (Type 1)	4/11/88	F8	09	02
PS/2 Model 70	3/17/89	F8	09	03
PS/2 Model 70	12/15/89	F8	09	04
PS/2 Model 70 (Type 3)	6/8/88	F8	0D	00
PS/2 Model 70	2/20/88	F8	0D	01
PS/2 Model 70 486 (Type 4)	12/1/89	F8	1B	00
PS/2 Model P70		F8	0B	00
PS/2 Model P70		F8	0B	02
PS/2 Model 80 (Type 1)	3/30/87	F8	00	00
PS/2 Model 80		F8	00	01
PS/2 Model 80	6/19/89	F8	00	02
PS/2 Model 80 (Type 2)	10/7/87	F8	01	00
PS/2 Model 80	11/21/89	F8	80	01
PS/2 Model 80	2/15/90	F8	80	02

Figure 6-1 (Part 2 of 3). System Identification

Product	BIOS Date	Model Byte	Submodel Byte	Revision
PS/2 Model 90 (Type 1)	10/1/90	F8	11	00
PS/2 Model 90 (Type 2)	10/1/90	F8	13	00
PS/2 Model 90 (Type 3)	4/24/91	F8	2D	01
PS/2 Model 90 (Type 3)	4/24/91	F8	2F	01
PS/2 Model 95 (Type 1)	10/1/90	F8	14	00
PS/2 Model 95 (Type 2)	10/1/90	F8	16	00
PS/2 Model 95 (Type 3)	4/24/91	F8	2C	01
PS/2 Model 95 (Type 3)	4/24/91	F8	2E	01

Figure 6-1 (Part 3 of 3). System Identification

Note: Information about a specific system board can be found in the technical reference for that model.

Notes:

Section 7. Scan Code/Character Code Combinations

Note: For scan code/character code combinations for the PC Space Saving (84-/85-key) Keyboard, refer to the *Personal System/2 Model 25 Technical Reference* or the "Keyboards" section of the *Personal System/2 Hardware Interface Technical Reference—Common Interfaces.*

The Keyboard Read function ((AH) = 00H) and the Keyboard Status function ((AH) = 01H) of Interrupt 16H apply to the 83-/84-key keyboard, the 101-/102-key keyboard (standard and extended function), and the 122-key keyboard. However, these functions return only the scan code/character code combinations that are supported by the 83-/84-key keyboard and the 101-/102-key standard-function keyboard.

The Extended-Keyboard Read function ((AH) = 10H) and the Extended Keystroke Status function ((AH) = 11H) apply to the 83-/84-key keyboard, the 101-/102-key keyboard (standard and extended function), and the 122-key keyboard. However, these functions return only the scan code/character code combinations that are supported by the 101-/102-key extended-function keyboard.

The Keyboard Read for the 122-Key Keyboard function ((AH) = 20H) and the Keystroke Status for the 122-Key Keyboard function ((AH) = 21H) apply to the 83-/84-key keyboard, the 101-/102-key keyboard (standard and extended function), and the 122-key keyboard. See the table on page 7-13 for the additional keystrokes and scan code/character code combinations that are supported by the 122-key keyboard.

The following table lists the keyboard keystrokes and the scan code/character code combinations that are returned through Interrupt 16H.

Keystroke	83-/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
f	21/66	21/66	21/66

Figure 7-1 (Part 1 of 3). Keyboard Keystrokes

Keystroke	83-/84-Key Standard Function	101-/102-Key Standard Function	101-/102-Key Extended Function
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

Figure 7-1 (Part 2 of 3). Keyboard Keystrokes

Keystroke	83-/84-Key Standard Function	101-/102-Key Standard Function	101-/102-Key Extended Function
F10	44/00	44/00	44/00
F11	(no key)	--	85/00
F12	(no key)	--	86/00
Num Lock	**	**	**
Scroll Lock	**	**	**
Home	47/00	47/00	47/00
Up Arrow	48/00	48/00	48/00
PgUp	49/00	49/00	49/00
-	4A/2D	4A/2D	4A/2D
Left Arrow	4B/00	4B/00	4B/00
Center Key	--	--	4C/00
Right Arrow	4D/00	4D/00	4D/00
+	4E/2B	4E/2B	4E/2B
End	4F/00	4F/00	4F/00
Down Arrow	50/00	50/00	50/00
PgDn	51/00	51/00	51/00
Ins	52/00	52/00	52/00
Del	53/00	53/00	53/00
SysRq	**	(no key)	(no key)
Key 45	(no key)	56/5C	56/5C
Enter	(no key)	1C/0D	E0/0D
/	(no key)	35/2F	E0/2F
PrtSc	(no key)	**	**
Pause	(no key)	**	**
Home	(no key)	47/00	47/E0
Up Arrow	(no key)	48/00	48/E0
PageUp	(no key)	49/00	49/E0
Left Arrow	(no key)	4B/00	4B/E0
Right Arrow	(no key)	4D/00	4D/E0
End	(no key)	4F/00	4F/E0
Down Arrow	(no key)	50/00	50/E0
PageDown	(no key)	51/00	51/E0
Insert	(no key)	52/00	52/E0
Delete	(no key)	53/00	53/00

** These combinations do not provide a keystroke for the application that is currently running, but they perform some other action. They are not put into the Interrupt 16H queue.
-- These combinations have no function and are ignored.

Figure 7-1 (Part 3 of 3). Keyboard Keystrokes

The following table lists the Shift keyboard keystrokes and the scan code/character code combinations that are returned through Interrupt 16H.

Keystroke	83-/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 7-2 (Part 1 of 3). Shift Keyboard Keystrokes

Keystroke	83-/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 SysRq	**	(no key)	(no key)
Shift Key 45	(no key)	56/7C	56/7C
Shift Enter	(no key)	1C/0D	E0/0D
Shift /	(no key)	35/2F	E0/2F

Figure 7-2 (Part 2 of 3). Shift Keyboard Keystrokes

Keystroke	83-/84-Key Standard Function	101-/102-Key Standard Function	101-/102-Key Extended Function
Shift PrtSc	(no key)	**	**
Shift Pause	(no key)	**	**
Shift Home	(no key)	47/00	47/E0
Shift Up Arrow	(no key)	48/00	48/E0
Shift PgUp	(no key)	49/00	49/E0
Shift Left Arrow	(no key)	4B/00	4B/E0
Shift Right	(no key)	4D/00	4D/E0
Shift End	(no key)	4F/00	4F/E0
Shift Down Arrow	(no key)	50/00	50/E0
Shift PgDn	(no key)	51/00	51/E0
Shift Insert	(no key)	52/00	52/E0
Shift Delete	(no key)	53/00	53/E0

** These combinations do not provide a keystroke for the application that is currently running, but they perform some other action. They are not put into the Interrupt 16H queue.
-- These combinations have no function and are ignored.

Figure 7-2 (Part 3 of 3). Shift Keyboard Keystrokes

The following table lists the Ctrl keyboard keystrokes and the scan code/character code combinations that are returned through Interrupt 16H.

Keystroke	83-/84-Key Standard Function	101-/102-Key Standard Function	101-/102-Key Extended Function
Ctrl Esc	01/1B	01/1B	01/1B
Ctrl 1	--	--	--
Ctrl 2 (NUL)	03/00	03/00	03/00
Ctrl 3	--	--	--
Ctrl 4	--	--	--
Ctrl 5	--	--	--
Ctrl 6 (RS)	07/1E	07/1E	07/1E
Ctrl 7	--	--	--
Ctrl 8	--	--	--
Ctrl 9	--	--	--
Ctrl 0	--	--	--
Ctrl _	0C/1F	0C/1F	0C/1F
Ctrl =	--	--	--
Ctrl Backspace (DEL)	0E/7F	0E/7F	0E/7F
Ctrl Tab	--	--	94/00

Figure 7-3 (Part 1 of 3). Ctrl Keyboard Keystrokes

Keystroke	83-/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
Ctrl F4	61/00	61/00	61/00
Ctrl F5	62/00	62/00	62/00

Figure 7-3 (Part 2 of 3). Ctrl Keyboard Keystrokes

Keystroke	83-/84-Key Standard Function	101-/102-Key Standard Function	101-/102-Key Extended Function
Ctrl F6	63/00	63/00	63/00
Ctrl F7	64/00	64/00	64/00
Ctrl F8	65/00	65/00	65/00
Ctrl F9	66/00	66/00	66/00
Ctrl F10	67/00	67/00	67/00
Ctrl F11	(no key)	--	89/00
Ctrl F12	(no key)	--	8A/00
Ctrl Num Lock	--	--	--
Ctrl Scroll Lock	--	--	--
Ctrl Home	77/00	77/00	77/00
Ctrl Up Arrow	--	--	8D/00
Ctrl PgUp	84/00	84/00	84/00
Ctrl Keypad -	--	--	8E/00
Ctrl Left Arrow	73/00	73/00	73/00
Ctrl Center	--	--	8F/00
Ctrl Right Arrow	74/00	74/00	74/00
Ctrl Keypad +	--	--	90/00
Ctrl End	75/00	75/00	75/00
Ctrl Down Arrow	--	--	91/00
Ctrl PgDn	76/00	76/00	76/00
Ctrl Ins	--	--	92/00
Ctrl Del	--	--	93/00
Ctrl SysRq	**	(no key)	(no key)
Ctrl Key 45	(no key)	--	--
Ctrl Enter	(no key)	1C/0A	E0/0A
Ctrl /	(no key)	--	95/00
Ctrl PrtSc	(no key)	72/00	72/00
Ctrl Break	(no key)	00/00	00/00
Ctrl Home	(no key)	77/00	77/E0
Ctrl Up	(no key)	--	8D/E0
Ctrl PageUp	(no key)	84/00	84/E0
Ctrl Left	(no key)	73/00	73/E0
Ctrl Right	(no key)	74/00	74/E0
Ctrl End	(no key)	75/00	75/E0
Ctrl Down	(no key)	--	91/E0
Ctrl PageDown	(no key)	76/00	76/E0
Ctrl Insert	(no key)	--	92/E0
Ctrl Delete	(no key)	--	93/E0

** These combinations do not provide a keystroke for the application that is currently running, but they perform some other action. They are not put into the Interrupt 16H queue.
-- These combinations have no function and are ignored.

Figure 7-3 (Part 3 of 3). Ctrl Keyboard Keystrokes

The following table lists the Alt keyboard keystrokes and the scan code/character code combinations that are returned through Interrupt 16H.

Keystroke	83-/84-Key Standard Function	101-/102-Key Standard Function	101-/102-Key Extended Function
Alt Esc	--	--	01/00
Alt 1	78/00	78/00	78/00
Alt 2	79/00	79/00	79/00
Alt 3	7A/00	7A/00	7A/00
Alt 4	7B/00	7B/00	7B/00
Alt 5	7C/00	7C/00	7C/00
Alt 6	7D/00	7D/00	7D/00
Alt 7	7E/00	7E/00	7E/00
Alt 8	7F/00	7F/00	7F/00
Alt 9	80/00	80/00	80/00
Alt 0	81/00	81/00	81/00
Alt -	82/00	82/00	82/00
Alt =	83/00	83/00	83/00
Alt Backspace	--	--	0E/00
Alt Tab	--	--	A5/00
Alt q	10/00	10/00	10/00
Alt w	11/00	11/00	11/00
Alt e	12/00	12/00	12/00
Alt r	13/00	13/00	13/00
Alt t	14/00	14/00	14/00
Alt y	15/00	15/00	15/00
Alt u	16/00	16/00	16/00
Alt i	17/00	17/00	17/00
Alt o	18/00	18/00	18/00
Alt p	19/00	19/00	19/00
Alt [	--	--	1A/00
Alt]	--	--	1B/00
Alt Return	--	--	1C/00
Alt Ctrl	**	**	**
Alt a	1E/00	1E/00	1E/00
Alt s	1F/00	1F/00	1F/00
Alt d	20/00	20/00	20/00
Alt f	21/00	21/00	21/00
Alt g	22/00	22/00	22/00
Alt h	23/00	23/00	23/00
Alt j	24/00	24/00	24/00
Alt k	25/00	25/00	25/00
Alt l	26/00	26/00	26/00
Alt ;	--	--	27/00
Alt '	--	--	28/00
Alt `	--	--	29/00

Figure 7-4 (Part 1 of 3). Alt Keyboard Keystrokes

Keystroke	83-/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 Numbers	#	#	#
Alt Del	--	--	--
Alt SysRq	**	(no key)	(no key)
Alt Key 45	(no key)	--	--
Alt Enter	(no key)	--	A6/00
Alt /	--	--	A4/00
Alt Print Screen	(no key)	**	**
Alt Pause	(no key)	**	**
Alt Home	(no key)	--	97/00
Alt Up	(no key)	--	98/00
Alt PageUp	(no key)	--	99/00
Alt Left	(no key)	--	9B/00
Alt Right	(no key)	--	9D/00
Alt End	(no key)	--	9F/00
Alt Down	(no key)	--	A0/00

Figure 7-4 (Part 2 of 3). Alt Keyboard Keystrokes

Keystroke	83-/84-Key Standard Function	101-/102-Key Standard Function	101-/102-Key Extended Function
Alt PageDown	(no key)	--	A1/00
Alt Insert	(no key)	--	A2/00
Alt Delete	(no key)	--	A3/00

** These combinations do not provide a keystroke for the application that is currently running, but they perform some other action. They are not put into the Interrupt 16H queue.
-- These combinations have no function and are ignored.
See the note below for use of the Alt key with the keypad number keys.

Figure 7-4 (Part 3 of 3). Alt Keyboard Keystrokes

Note: For all keyboards, the numeric keypad can be used in combination with the Alt key to type any ASCII character. The scan code (always 00) and character code are returned after the Alt key is released. For example, pressing the Alt key and keypad 1, then releasing the Alt key returns scan code/character code combination hex 00/01; pressing the Alt key and keypad 255, then releasing the Alt key returns scan code/character code combination hex 00/FF.

The following table lists the keyboard keystrokes and the scan code/character code combinations that the 122-keyboard supports in addition to those listed previously. The Keyboard Read for the 122-Key Keyboard function ((AH)=20H) and the Keystroke Status for the 122-Key Keyboard function ((AH)=21H) must be used to return scan code/character code combinations for these keystrokes.

Keystroke	122-Key Extended Extended Function
F13	EC/00
F14	ED/00
F15	EE/00
F16	EF/00
F17	F4/00
F18	F5/00
F19	F6/00
F20	F7/00
F21	F8/00
F22	F9/00
F23	FA/00
F24	C0/00
PA1	E8/00
EraseEOF	F0/00
Clear	FB/00
Shift F13	C1/00
Shift F14	C3/00
Shift F15	C4/00
Shift F16	C5/00
Shift F17	C6/00
Shift F18	C7/00
Shift F19	C8/00
Shift F20	C9/00
Shift F21	CA/00
Shift F22	CB/00
Shift F23	CC/00
Shift F24	CD/00
Shift PA1	E9/00
Shift EraseEOF	F1/00
Shift Clear	FB/00
Ctrl F13	CE/00
Ctrl F14	CF/00
Ctrl F15	D0/00
Ctrl F16	D1/00
Ctrl F17	D2/00
Ctrl F18	D3/00
Ctrl F19	D4/00
Ctrl F20	D5/00
Ctrl F21	D6/00
Ctrl F22	D7/00
Ctrl F23	D8/00

Figure 7-5 (Part 1 of 2). Keyboard Keystrokes — 122-Key Keyboard

Keystroke	122-Key Extended Extended Function
Ctrl F24	D9/00
Ctrl PA1	EA/00
Ctrl EraseEOF	F2/00
Ctrl Clear	FC/00
Alt F13	DA/00
Atl F14	DB/00
Alt F15	DC/00
Alt F16	DD/00
Alt F17	DE/00
Alt F18	DF/00
Alt F19	E2/00
Alt F20	E3/00
Alt F21	E4/00
Alt F22	E5/00
Alt F23	E6/00
Alt F24	E7/00
Alt PA1	EB/00
Alt EraseEOF	F3/00
Alt Clear	FD/00

Figure 7-5 (Part 2 of 2). Keyboard Keystrokes — 122-Key Keyboard

The following keys on the leftmost keypad of the 122-key keyboard do not produce scan code/character code combinations:

- Attn
- CrSel
- ExSel
- Copy Play
- The two blank keys.