Section 12. Compatibility

Introduction

This section discusses the major system differences between the IBM Personal Computer and IBM Personal System/2 product lines. Also included are programming considerations that must be taken into account when designing application programs for the IBM PS/1 computer.

System Board

The IBM PS/1 computer system board uses an 80286 microprocessor.

The 80286 system microprocessor and general architecture of the IBM PS/1 computer have created some fundamental differences between these systems and other systems. These differences must be taken into consideration when designing programs exclusively for the IBM PS/1 computer or programs compatible across the IBM Personal Computer and IBM Personal System/2 product lines. Programming considerations are discussed in "Application Guidelines" on page 12-6.

Diskette Drives and Controller

The following figure shows the read, write, and format capabilities for each type of diskette drive used by the IBM PS/1 computer.

Figure 12-1. Diskette Drive Read, Write, and Format Capabilities

Diskette Drive Type	160/180KB Mode	320/360KB Mode	720KB Mode	1.44MB Mode	1.2MB Mode
5.25-Inch 360KB Diskette Drive:					
Double-Sided	R W F	R W F	—	—	—
5.25-Inch 1.2MB Diskette Drive:					
1.2MB Drive	RWF	RWF	—	—	RWF
3.5-Inch Diskette Drive:					
1.44MB Drive	—	—	R W F	R W F	—

R-Read W-Write F-Format

Notes:

1. 5.25-inch diskette drives are supported by means of a 5.25-inch diskette drive option.
2. 5.25-inch diskettes designed for the 1.2MB mode cannot be used in either a 160/180KB or a 320/360KB diskette drive.
3. Diskettes written or formatted as 160/180KB or 320/360KB on a 5.25-inch 1.2MB drive may not work reliably when returned to a 5.25-inch 360KB diskette drive.

Warning: Sixteen-bit operations to the video subsystem can cause a diskette overrun in the 1.44MB mode because data width conversions may require more than 12 microseconds. If an overrun occurs, BIOS returns an error code and the operation should be retried.

Copy Protection

The following methods of copy protection may not work on systems using the 5.25-inch high-capacity diskette drive or the 3.5-inch 1.44MB diskette drive.

- Bypassing BIOS Routines
  - Track Density: The 5.25-inch high-capacity diskette drive records tracks at a density of 96 tracks per inch (TPI). This drive has to double-step in the 48 TPI mode that is handled by BIOS.

  - Data Transfer Rate: BIOS selects the proper data transfer rate for the media being used.
  - Diskette Parameters Table: Copy protection, which creates its own Diskette Parameters Table, may not work on these drives.

- Diskette Drive Controls
  - Rotational Speed: The time between two events on a diskette is a function of the diskette drive.
  - Access Time: Diskette BIOS routines must set the track-to-track access time for the different types of drives used in the system.
  - Diskette Change Signal: Copy protection may not be able to reset this signal.
- Write Current Control—Copy protection that uses write current control will not work because the controller selects the proper write current for the media being used.

Fixed Disk Drives and Controller

Reading from and writing to the fixed disk drive is initiated in the same way as with IBM Personal Computer products; however, new functions are supported. Detailed information about specific fixed disk drives and fixed disk adapters is available in "3.5-Inch 30MB Fixed Disk Drive and Controller," in Section 8.

Application Guidelines

Use the following information to develop application programs for the IBM PS/1 computer. Whenever possible, BIOS should be used as an interface to hardware in order to provide maximum compatibility and portability of applications across systems.

Hardware Interrupts

IBM PS/1 computer hardware interrupts are edge-triggered. On edge-sensitive 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.

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

Hardware interrupt IRQ9 is defined as the replacement interrupt level for the cascade level IRQ2. Program interrupt sharing should be implemented on IRQ2, interrupt hex 0A. 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 second interrupt controller.
3. When the interrupt occurs, the system microprocessor passes control to the IRQ9 (interrupt hex 71) interrupt handler.
4. This interrupt handler performs an end of interrupt (EOI) to the second interrupt controller and passes control to IRQ2 (interrupt hex 0A) interrupt handler.
5. This IRQ2 interrupt handler, when handling the interrupt, causes the device to reset the interrupt request prior to performing an EOI to the master interrupt controller that finishes servicing the IRQ2 request.

Software Interrupts

With the advent of software interrupt sharing, software interrupt routines must *serially connect* interrupts. Each routine must check the function value and if it is not in the range of function calls for that routine, it must transfer control to the next routine in the chain. Because software interrupts are initially pointed to 0:0, before serially connecting it is necessary to check for this case. If the next routine is pointed to 0:0 and the function call is out of range, the appropriate action is to set the carry flag and do a RET 2 to indicate an error condition.

High-Level Language 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 Computer and IBM Personal System/2 products. Specifically, the use of assembler language subroutines or hardware-specific commands (for example, In, Out, Peek, and Poke) must follow the assembler language rules. See "Assembler Language Programming Considerations" on page 12-7.

Any program that requires precise timing information should obtain it through an operating system or language interface; for example, TIME$ in BASIC. If greater precision is required, the assembler techniques in "Assembler Language Programming Considerations" are available. The use of programming loops may prevent a program from being compatible with other IBM Personal Computer products, IBM Personal System/2 products, and software.

Assembler Language Programming Considerations

This section describes fundamental differences between the systems in the Personal Computer and Personal System/2 product lines that may affect program development.

Opcodes

The following opcodes work differently on systems using the 80286 microprocessor than they do on systems using the 8088 or 8086 microprocessor.

- PUSH SP

  The 80286 microprocessor pushes the current stack pointer; the 8088 and 8086 microprocessors push the new stack pointer, that is, the value of the stack pointer after the PUSH SP instruction is completed.

- Single-step interrupt (when TF=1) on the interrupt instruction (Opcode hex CC, CD):

  The 80286 microprocessor does not perform a single-step interrupt on the INT instruction. The 8088 and 8086 microprocessors do perform a single-step interrupt on the INT instruction.

- The divide error exception (interrupt 0):

  The 80286 microprocessor pushes the CS:IP of the instruction that caused the exception; the 8088 and 8086 microprocessors push the CS:IP of the instruction following the instruction that caused the exception.

- Shift counts for the 80286 microprocessor:

  Shift counts are masked to 5 bits. Shift counts greater than 31 are treated as mod 32. For example, a shift count of 36 shifts the operand four places.

- Multiple lockout instructions:

  There are several microprocessor instructions that, when executed, lock out external bus signals. DMA requests are not honored during the execution of these instructions. Consecutive instructions of this type prevent DMA activity from the start of the first instruction to the end of the last instruction. To allow for necessary DMA cycles, as required by the diskette controller in a multitasking system, multiple lock-out instructions must be separated by a JMP SHORT $+2.

- Consecutive I/O commands:

  Consecutive I/O commands to the same I/O ports do not permit enough recovery time for some I/O adapters. To ensure enough time, a JMP SHORT $+2 must be inserted between IN/OUT instructions to the same I/O adapters.

  Note: A MOV AL,AH type instruction does not allow enough recovery time. An example of the correct procedure follows:

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

- I/O commands followed by an 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.

  Note: A MOV AL,AH type instruction does not allow enough recovery time. An example of the correct procedure follows:

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

80286 Microprocessor Anomalies

In the Protected mode, when any of the null selector values (0000H, 0001H, 0002H, 0003H) are loaded into the DS or ES registers with a MOV or POP instruction or a task switch, the 80286 always loads the null selector 0000H into the corresponding register.

The following describes the operation of all 80286 microprocessor parts:

- Instructions longer than 10 bytes (instructions using multiple redundant prefixes) generate exception #13 (General Purpose Exception) in both the Real Address mode and Protected mode.

- If the second operand of an ARPL instruction is a null selector, the instruction generates an exception #13.

ROM 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 ROM BIOS function calls, if permissible.

- In some environments, program interrupts are used to access these functions. This practice removes the absolute addressing from the program. Only the interrupt number is required.
- In systems using the 80286 microprocessor, IRQ 9 is redirected to INT hex 0A (hardware IRQ 2). This ensures that hardware designed to use IRQ 2 will operate in these systems. See "Hardware Interrupts" on page 12-6 for more information.
- The system can mask hardware sensitivity. Device drivers can be installed to replace the ROM BIOS with the same programming interface for new devices.
- 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. However, 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 hex 1E 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 hex 1E (0:78) to point to the new block.

  The parameters were established to allow the IBM PS/1 computer to operate the 3.5-inch 1.44MB diskette drive, 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).

  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.

Note: Special considerations are required for format operations. Refer to the diskette section of *BIOS Interface Technical Reference for the IBM PS/1™ Computer* for the required details.

If a parameter block contains a head settle time parameter value of 0 milliseconds, and a write or format operation is being performed, the following minimum head settle times are enforced.

Figure 12-2. Write and Format Head Settle Time

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:	
1.44MB	15 milliseconds

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

If a parameter block contains a motor start wait parameter of less than 500 milliseconds for a write or verify operation, diskette BIOS enforces a minimum time of 500 milliseconds. Read and write operations use the motor start time provided by the parameter block.

- Programs may be designed to reside on either 5.25-inch or 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 15 (1 sector).

     If the verify operation fails, either a 5.25-inch 360KB 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.

3. Verify Track 0, Head 0, Sector 18 (1 sector). If the verify operation fails, a 5.25-inch high-capacity diskette is installed. A successful verification indicates that a 3.5-inch 1.44MB diskette is installed.

Hardware Compatibility

The IBM PS/1 computer maintains many of the interfaces used by the IBM Personal Computer AT. In most cases, command and status organization of these interfaces are maintained.

The functional interfaces for the IBM PS/1 computer are compatible with the following interfaces:

- Intel 8259 interrupt controllers (with edge triggering).
- Intel 8254 timers driven from 1.190 MHz (timer 0, 1, and 2).
- Intel 8237 DMA controller address/transfer counters, page registers and status fields only. The Command and Request registers are not supported. The rotate and mask functions are not supported. The Mode register is partially supported.
- NS16450 serial port.
- Intel 8088, 8086, and 80286 microprocessors.
- Intel 8272 diskette drive controller.
- Motorola MC146818A Real Time Clock command and status (RTC/CMOS/RAM reorganized).
- Intel 8042 keyboard port at address hex 0060 and 0064.

  Note: Use the new interface described in "System Control Port A (Hex 0092)" on page 3-167 to change the status of the A20 address line. Use the model and submodel bytes to determine that the program is running on an IBM PS/1 computer.

- Display modes supported by the IBM Monochrome Display and Printer Adapter, IBM Color/Graphics Monitor Adapter, and the IBM Enhanced Graphics Display Adapter.
- Parallel printer ports (Parallel 1, Parallel 2, and Parallel 3) in Compatibility mode.

Multitasking Provisions

The BIOS contains a feature to assist multitasking implementation. Hooks are provided for a multitasking dispatcher. Whenever a busy (wait) loop occurs in the BIOS, a hook is provided for the program to break out of the loop. Also, whenever BIOS services an interrupt, a corresponding wait loop exits and another hook is provided. Thus a program can be written that employs most of the device driver code. The following is valid only in the Real Address mode and must be taken by the code to allow this support:

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

Warning: Sixteen-bit operations to the video subsystem can cause a diskette overrun in the 1.44MB mode because data width conversions may require more than 12 microseconds. If an overrun occurs, BIOS returns an error code and the operation should be retried.

Interfaces

There are four interfaces used by the multitasking dispatcher:

Startup: First, the startup code hooks INT 15H. The dispatcher is responsible to check for function codes of (AH)= hex 90 or 91. The Wait and Post sections describe these codes. The dispatcher must pass all other functions to the previous user of INT 15H. This can be done by a JMP or a CALL. If the function code is hex 90 or 91, the dispatcher should do the appropriate processing and return by the IRET instruction.

Serialization: It is up to the multitasking system to ensure that the device driver code is used serially. Multiple entries into the code can result in serious errors.

Wait: When the BIOS is about to enter a busy loop, it first issues an INT 15H with a function code of hex 90 in AH. 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: Whenever the BIOS has set an interrupt flag for a corresponding busy loop, an INT 15H occurs with a function code of hex 91 in AH. This signals a post condition. At this point, the dispatcher should set the task status to *ready to run* and return to the interrupt routine. The following is an outline of the code added to BIOS that performs this function.

```
MOV AX, 91XXH          ; post code AH and
                       ; type code AL
INT 15H                ; issue call
```

Classes

The following types of wait loops are supported:

- The class for 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 only one task at a time.
- The class for hex 80 to BF is reentrant. There is no restriction on the number of tasks that can access the device.
- The class for hex C0 to FF is noninterrupt. There is no corresponding interrupt for the wait loop. Therefore, it is the responsibility of the dispatcher to determine what satisfies this condition to exit from the loop.

Function Code Classes

Type Code (AL)	Description
00H->7FH	Serially reusable devices; the operating system must serialize access.
80H->0BFH	Reentrant devices; ES:BX is used to distinguish different calls (multiple I/O calls are allowed simultaneously).

0C0H->0FFH Wait-only calls; there is no complementary post for these waits--these are time out only. Times are function-number dependent.

Function Code Assignments: The following are specific assignments for the IBM PS/1 computer BIOS. Times are approximate. They are grouped according to the classes described under "Function Code Classes."

Figure 12-3. Functional Code Assignments

Type Code (AL)	Time Out	Description
00H	Yes (12 seconds)	Fixed Disk
01H	Yes (2 seconds)	Diskette
02H	No	Keyboard
0FCH	Yes (20 seconds)	Fixed Disk Reset
0FDH	Yes (500-ms Read/Write)	Diskette Motor Start
0FEH	Yes (20 seconds)	Printer

The asynchronous support has been omitted. The serial and parallel controllers generate interrupts, but BIOS does not support them in the interrupt mode. Therefore, the support should be included in the multitasking system code if that device is to be supported.

Time Outs

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 amount of time before indicating an error. The dispatcher should return to the BIOS wait loop with the carry bit set if a time out occurs.

IBM PS/1 Computer Considerations

ROM Drive

The IBM PS/1 computer contains IBM DOS 4.01 in ROM. This appears to the user as an extra drive (C: on a single diskette system and D: on a fixed disk system). This ROM drive contains the DOS kernel files (IBMBIO and IBMDOS), COMMAND.COM, an AUTOEXEC.BAT, a CONFIG.SYS, ROMSHELL.COM (which implements the four quadrant screen), and various support programs. The ROM drive is implemented as an installable file system by integrating a version of IFSFUNC into IBMBIO. Normally, IFSFUNC is loaded to run a Local Area Network (LAN). Consequently, IFSFUNC cannot be loaded when the system is booted from ROM which prohibits running a network. In fact, the ROM drive appears to be a network drive to the rest of the system.

A CUSTOMIZ program is included with the software allowing the user to customize the way the system starts. When changing items on the CUSTOMIZ menu, battery-backed CMOS RAM bits are modified. When the system is started, the BIOS looks at these customization bits to determine whether to boot from the ROM drive, the diskette drive, or the fixed disk.

When booting from a diskette or fixed disk, the ROM drive does not exist. When the ROM drive is booted, IBMBIO and COMMAND.COM look at the customization bits to determine whether to read the CONFIG.SYS and AUTOEXEC.BAT from ROM, from diskette, or from fixed disk. After the AUTOEXEC.BAT is processed, COMMAND.COM tries to load ROMSHELL.COM if SHELLSTB.COM has been installed in the CONFIG.SYS. Otherwise, the DOS prompt is displayed.

From the ROMSHELL, the user can get to the DOS prompt by pressing **Shift + F9**. In addition, pressing **Alt + SysRq** from the ROMSHELL causes a conventional PC boot from the diskette or fixed disk, regardless of the state of the CUSTOMIZ bits. This is useful when a "one-time" boot is required.

National Language Support

National Language Support (NLS) is implemented by having a unique ROM for each country. In addition to the files mentioned above, the ROM drive contains NLSFUNC.EXE, MODE.COM, KEYB.COM, KEYBOARD.SYS, COUNTRY.SYS, DISPLAY.SYS, and EGA.CPI. The AUTOEXEC.BAT and CONFIG.SYS contain the necessary statements to load the appropriate code page and keyboard for a given country.

Machine-Sensitive Programs

Programs can select machine-specific features, but they must first identify the machine and model type. IBM has defined methods for uniquely determining the specific machine type. The location of the machine model bytes can be found through INT 15H function code (AH) = hex C0. The model byte for the IBM PS/1 computer is shown in the following figure.

Figure 12-4. Machine Model Byte

Model Byte	Sub-Model Byte	Product Name
FC	0B	IBM PS/1 computer

See *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference* for a listing of model bytes for other IBM products.