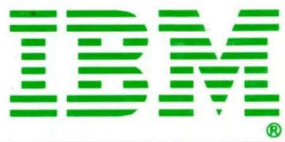
Volume 1, Number 3 November 1983

IBM Personal Computer Seminar Proceedings

The Publication for Software Developers
of the IBM Personal Computers

Published by International Business Machines Corporation
Entry Systems Division



IBM believes the statements contained herein are accurate as of the date of publication of this document. However, IBM makes no warranty of any kind with respect to the contents hereof.

All specifications are subject to change without notice.

IBM®

Contents

Introduction and Welcome

The IBM Entry Systems Division is pleased to provide the Proceedings of the IBM Personal Computer Seminar, designed for independent developers of software for the IBM Personal Computer. The purpose of these Proceedings is to aid you in your software development efforts by providing relevant information about new product announcements and enhancements to existing products. This issue is prepared in conjunction with this seminar. Future issues will be prepared in conjunction with other seminars for the IBM Personal Computers and will cover topics presented at those seminars.

Throughout these proceedings, the term Personal Computers addresses the IBM Personal Computer and the IBM Personal Computer XT. The term family of IBM Personal Computers addresses the Personal Computer, the IBM Personal Computer XT and the recently announced PC*jr*.

Purpose

What is our purpose in putting out a publication such as this? It is quite simple.

The IBM Personal Computer family is a resounding success. We've had a lot of help in achieving this success, and much of it came from the independent software developers.

As you proceed with your software development, do you at times wish for some bit of information or direction which would make the job easier? Information which IBM can provide? This is the type of information we want to make available to you.

Since we want to be assured of giving you the information you need, we ask you to complete the questionnaire which appears at the end of these Proceedings. Your response to this questionnaire will be taken into account in preparing the content of future issues, as well as the content of seminars we will present at microcomputer industry trade shows.

Topics

The following list gives a general indication of the topics we plan to cover in future seminars and include in the IBM Personal Computer Seminar Proceedings:

- Information exchange forum — letters to the editor format
- Development tools — languages, database offerings
- Compatibility issues
- New devices — capacities and speeds
- System capacities — disk and memory
- Enhancements in maintenance releases
- Tips and techniques
- New system software
- Hardware design parameters
- Tips on organizing and writing documents for clear and easy reading
- Changes to terms and conditions

DOS 2.1, 2.0 and 1.1 Comparisons

Below is a partial comparison of the DOS 2.1, 2.0 and 1.1 products. Total disk or memory requirements cannot be determined by adding the requirements of the members shown. Refer to the appropriate publications for specific information. DOS 2.1 supports the family of IBM Personal Computers. DOS 2.1 does not replace DOS 1.1. DOS 2.1 is oriented to larger systems and requires a minimum 64KB.

Feature	------ Storage Required -----		
	DOS 1.1	**DOS 2.0**	**DOS 2.1**
DOS	12KB	24KB	24KB
BASIC	8KB	11KB	11KB
BASICA	13KB	21KB	21KB
	--------- Disk Size ---------		
	DOS 1.1	**DOS 2.0**	**DOS 2.1**
COMMAND.COM	5KB	17KB	17KB
BASIC	11KB	16KB	16KB
BASICA	16KB	25KB	26KB
	--------- Functions ---------		
	DOS 1.1	**DOS 2.0**	**DOS 2.1**
160KB/320KB Disk Support	Yes	Yes	Yes
180KB/360KB Disk Support	No	Yes	Yes
Fixed Disk Support	No	Yes	Yes
Background File Printer	No	Yes	Yes
Loadable Device Drivers	No	Yes	Yes
Memory Management	No	Yes	Yes
Function Call to Load Programs	No	Yes	Yes
BASIC Double Precision Functions	No	Yes	Yes
Cartridge Support	No	No	Yes
Compact Printer Support	No	No	Yes

IBM Personal Computer Disk Operating System 2.1

The IBM Personal Computer Disk Operating System (DOS) 2.1 is an enhanced release of DOS 2.0 and has the same functions and storage requirements as DOS 2.0. DOS 2.1 supports the IBM Personal Computer, the IBM Personal Computer XT, and the new IBM PC*jr*. IBM Personal Computer DOS 2.1 provides the required support between IBM Personal Computer hardware and an application program.

DOS 2.1 is packaged with the IBM Personal Computer Disk Operating System User's Guide (Level 2.1) and the IBM Personal Computer Disk Operating System Reference Manual (Level 2.1).

DOS 2.1 was written to take advantage of the differences in the PC*jr*, such as:

- minimum memory environment
- IBM PC Compact Printer
- diskette characteristics
- BASIC interface

Thus all DOS commands are supported in the 64KB entry environment of PC*jr*. On the IBM PC*jr*, the IBM PC Compact Printer is automatically redirected as the parallel printer if no parallel printer is attached. The specific diskette drive characteristics of PC*jr* are supported; and PC*jr* Cartridge BASIC can be activated.

IBM PC*jr* Architecture

The IBM PC*jr* is the newest member of the IBM Personal Computer family. The PC*jr* is compatible with the IBM Personal Computer and the IBM Personal Computer XT, and contains many of the same components. The block diagram for the IBM PC*jr* is shown in Figure 2.

Following are brief descriptions of the major components of the IBM PC*jr*:

- 8088 Microprocessor

  The 8088 Microprocessor in the PC*jr* is the same microprocessor used in the IBM Personal Computer and IBM Personal Computer XT. The 8088 runs at the same speed in all three machines. The 8088 is the 8-bit data bus version of the 16-bit 8086 microprocessor. Both the 8088 and 8086 have the same one megabyte address space and instructions.

- 64KB Read Only Memory (ROM)

  The system board contains 64KB of Read Only Memory (ROM). This ROM contains the system Basic Input/Output System (BIOS), power-on diagnostics, customer level diagnostics, Cassette BASIC, and the Keyboard Adventure program that teaches how to use the 62-key keyboard.

- 64KB Random Access Memory (RAM) and 64KB Memory and Display Expansion

  The PC*jr* system board comes with 64KB of Random Access Memory (RAM). The system board also has a connector for a 64KB Memory and Display Expansion Option, which increases the total RAM to 128KB. The 128KB of RAM in the PC*jr* is shared between the processor and the video circuitry.

- 128KB Cartridge Interface

  The PC*jr* has a 128KB ROM cartridge interface to allow ROM programs to be inserted into the system.

- Video Graphics Subsystem

  The functions of the IBM Personal Computer Color/Graphics Adapter have been moved, with some enhancements, to the IBM PC*jr* system board. All the modes that are shared between the Color/Graphics Adapter and the PC*jr* are compatible.

- Sound Subsystem

  Four sound inputs are available on the PC*jr* sound subsystem — a Complex Sound Generator, 8253 timer sound, cassette audio input and audio input from the I/O channel.

- Game Adapter

  The PC*jr* system board comes with a game adapter. The game adapter connects joysticks and paddles to the system. This adapter is compatible with the game adapter available on the IBM Personal Computer and IBM Personal Computer XT.

- Serial Port

  The serial port on the PC*jr* system board is compatible with the IBM Personal Computer Asynchronous Communications Adapter at the Card Two address (2F8H). The serial port will support asynchronous communications at the following transmission speeds:

  - 4800 bits per second for transmitting
  - 1200 bits per second for receiving with overlapped keyboard operation
  - 4800 bits per second for receiving with no keyboard overlap

  The transmission rate divisors in the PC*jr* are different from those on the IBM Personal Computer Asynchronous Communications Adapter because the PC*jr* uses an input clock of 1.7895 MHz.

- Keyboard Interface

  The keyboard interface to the PC*jr* system unit consists of an infrared receiver card and a cable interface. The infrared receiver converts the infrared light transmitted from the keyboard into an electrical signal. This signal causes a nonmaskable interrupt (NMI) to the 8088, which then deserializes the data to obtain the keyboard scan code data.

- Modem and Diskette Interface

  The internal modem in the PC*jr* system unit is a 300 bits per second Bell 103 compatible modem, with auto answer and auto dial capability. The internal modem connects directly to the phone line and does not require an acoustic coupler. The PC*jr* communicates with the modem via a serial port on the modem adapter card. This serial port is compatible with the IBM Personal Computer Asynchronous Communications Adapter at the Card One address (3F8H).

  A single diskette drive can be installed in the PC*jr*. This drive is a double sided, double density, slimline 5-1/4" diskette drive. Because the IBM PC*jr* has no Direct Memory Access (DMA) controller, diskette operations cannot be overlapped with keyboard or asynchronous communication operations.

- Nine Level Interrupt

  The PC*jr* provides nine levels of interrupts. All but three of these interrupts are used on the system board for diskette, modem, serial port, vertical retrace, timer and keyboard operations. The remaining three interrupts are available on the I/O expansion bus.

- I/O Expansion Bus

  The PC*jr* has an I/O Expansion Bus which contains many of the same lines found on the I/O channel of the IBM Personal Computer and IBM Personal Computer XT.

Video Graphics Subsystem

Unlike the IBM Personal Computer and the IBM Personal Computer XT, the IBM PC*jr* includes the video subsystem on the system board. This Video Graphics Subsystem is an enhanced version of the IBM Personal Computer Color/Graphics Adapter and is compatible with it.

The video subsystem operates in two modes, alphanumeric and graphic. Figure 1 shows the alphanumeric and graphic modes available on the IBM PC*jr* and which modes are compatible with the Color/Graphics Adapter.

The 64KB of Random Access Memory (RAM) on the PC*jr* system board is shared between the processor and the video every 1.1 μs. Two fetches are made for the display, and one fetch is allotted to the processor. Each fetch handles 8 bits of data. This is sufficient bandwidth to support 40-column alpha, 160 x 200 16-color graphics, 320 x 200 4-color graphics, and 640 x 200 2-color graphics. Figure 3 shows how this is done.

If a 64KB Memory and Display Expansion Option is installed on the PC*jr* system unit, the memory is reconfigured so that all the even bytes reside on the system board and all the odd bytes reside on the expansion card. This allows the video adapter to fetch 16 bits of data for each fetch. This bandwidth supports 80-column alpha, 320 x 200 16-color graphics, and 640 x 200 4-color graphics. Figure 4 shows how this is done.

Logic has been added to the PC*jr* so that reads and writes that are directed to the IBM Personal Computer Color/Graphics Adapter at B8000 will be compatible on the PC*jr*. Figure 5 shows how reads and writes to B8000H are redirected to the main storage area by the Processor Page Register. The Processor Page Register selects which 16KB pages are to be redirected.

The PC*jr* also has a CRT Page Register which allows any 16KB region of main storage to be mapped to the display. This will aid in animation, since a program can simultaneously update a page while displaying a different page. Then, at vertical retrace time, the CRT and processor pages can be swapped. The ROM BIOS normally sets the CRT and processor pages to the top 16KB of RAM available, and accordingly adjusts the pointer to the top of RAM.

The PC*jr* has a custom video chip which shares the memory and also takes the data from memory and formats it for the display. The Video Gate Array (VGA) contains a 16 x 4 palette. This palette assigns color to the data coming from memory. This allows a program to select, for instance, any four colors out of a possible 16 to be displayed on the screen in a four color mode (320 x 200 4 color). The palette also allows the program to quickly change the colors of the screen image by changing the palette instead of the memory data.

The VGA also contains a border color register which is independent of the palette. This allows the program to select one out of a possible 16 colors for the border.

In all low bandwidth graphics modes (160 x 200 16 colors, 320 x 200 4 colors, and 640 x 200 2 colors) the VGA receives the same amount of data from memory but formats it differently to give the resolution and colors for the mode selected. These modes can be switched at any time since the 6845 does not have to be programmed and the VGA just interprets the data differently. The two high bandwidth graphics modes can be switched in a similar fashion for the same reason.

The video subsystem on PC*jr* has three output types. RGBI direct drive video, which can drive the IBM Color Monitor; a composite video output for color or black and white composite monitors and a connector which will attach the IBM Connector for TV (RF modulator).

Sound Subsystem

The nucleus of the sound subsystem is a analog multiplexer which allows one of four different sound sources to be selected, amplified, and sent to the audio outputs. The multiplexer and the amplifier are configured so the amplifier's gain is unique and consistent with each sound source. This provides a consistent level of output with any of the sound sources. One of the sound sources is a Complex Sound Generator chip the SN76496N. This chip has three programmable frequencies which can be mixed to form chords and a white noise generator which may also be mixed for special effects. Each of the three channels, as well as the noise channel can be independently attenuated up to 16 levels.

The 8253 timer channel 2 is the second sound source available in the sound subsystem. This sound source is compatible with the sound available on the IBM PC and the PCXT.

The other two sound sources are the cassette audio and an audio input from the I/O channel. The cassette audio input allows the audio information from the cassette deck to be routed to the audio outputs. This will allow both data and audio information to be mixed on the cassette recorder. The I/O channel audio input will allow easy addition of speech and other audio features to the system. This is because the feature can use the amplifier and the speaker already attached to the system.

The sound subsystem output is routed to the RGBI Direct Drive Video connector for use with direct drive monitors with amplifiers and speakers, the IBM Connector for TV for using the amplifier and speaker in the TV, and an external audio jack for connection of an external amplifier and speaker.

Figure 6 is a block diagram of the sound subsystem.

ROM Cartridge Interface

The ROM Cartridge Interface allows programs on ROM to be inserted and run on the PC*jr* system. Up to 128K of ROM can be accessed by the Cartridge Interface which consists of two cartridge slots. These two slots are identical and a cartridge can be inserted into either slot. The cartridge itself configures where in the memory map it will reside by wiring the desired chip select to the ROM modules inside. Figure 9 shows the chip selects and where they appear in the memory map. Two cartridges can be inserted into the system as long as they select different addresses in which to reside. This is useful when one cartridge contains a program, and the other a data base which occupies several cartridges.

Another feature of the cartridge interface is the ability to replace the PC*jr* base system ROM on a cartridge. Two signals on the cartridge interface called -BASE 1 ROM IN CARTRIDGE and -BASE 2 ROM IN CARTRIDGE are used to implement this feature. When one of these signals is pulled low, the corresponding ROM on the PC*jr* base system board is disabled. The cartridge can then use the correct chip select line (CS6 or CS7, see Figure 9) to provide the code in the cartridge interface. This feature can be used to gain control of the system at power on time. It can also be used to replace the system BIOS, Cassette Basic, etc.

A system reset is generated whenever a cartridge is inserted or removed from the system. This is accomplished with the cartridge reset tab. This tab is shown in Figure 7, and the cartridge interface is shown in Figure 8. Notice that the A1 pin on the interface is ground and A2 is the cartridge reset tab. The L shaped pattern shown in Figure 7 causes A2 to be momentarily grounded when a cartridge is inserted or removed. This causes the system reset.

Input and Output Channel

The PC*jr* I/O Channel Expansion connector is available on the right-hand side of the system unit. The connector is normally covered by a removable cover. To add an expansion module to the system the cover is removed, the expansion module is plugged into the connector and is then screwed to the system unit with four screws. The expansion module provides a feed through connector which will allow additional expansion modules to be added in a similar fashion. When the last expansion module has been attached the cover is replaced on the end. This allows the expansion modules to be added to the PC*jr* and still look as if they are integrated into the system unit.

The PC*jr* Channel Expansion connector contains many of the same signals found on the Personal Computer and PC*jr* expansion slot. It is feasible that just about any feature which can be added to the IBM Personal Computer and the IBM Personal Computer XT can be added to the IBM PC*jr*. Of course there are restrictions on the size, power and implementation of these features, but most of these can be overcome. The major differences between the PC*jr* I/O Channel Expansion and the IBM Personal Computer / IBM Personal Computer XT expansion slots are explained below, as are, the additional capabilities of the PC*jr* I/O Channel Expansion.

The main difference between the PC*jr* expansion bus and the IBM Personal Computers' expansion bus is the lack of Direct Memory Access (DMA) and some interrupts. The PC*jr* has only three interrupts available on the I/O Channel because the rest of the interrupts have been used on the system board for such things as serial port, modem port, keyboard, diskette, vertical retrace and timer interrupt. Although the PC*jr* does not have DMA, it does have the ability to add it with an external expansion. Also, two lines have been added to the I/O Channel to allow the diskette to be run under DMA. These lines are the DACK0 and DRQ0 lines.

The HRQ, HLDA and IO/-M allow the addition of DMA, alternate processors or any other bus masters to the PC*jr* system. The HRQ line is used by the bus master to gain control of the bus. The PC*jr* responds with an HLDA when it is ready to give up the bus, indicating it has placed in a high impedance mode all the address, data and control lines. The IO/-M line is placed in a high impedance mode and controlled by the bus master to indicate the type of address on the bus. A DMA expansion module would hold this line low, indicating memory address on the bus, since I/O is selected by the DACK line. Each I/O device should include the IO/-M line in its decode. This line replaces the AEN on the Personal Computer and Personal Computer XT Expansion Bus. The READY line is an open collector input, which is usually used to indicate that an I/O or memory transfer to the 8088 requires wait states. This line is also brought low by the system board on access to RAM and the Complex Sound Generator. This allows the alternate bus master to interface with these devices.

The PC*jr* also has a -CARD SLCTD line on the I/O expansion bus. This is an open collector line that is pulled by the device which is selected for the operation currently on the bus. This line allows easy interface to an I/O Expansion Box over a cable. It can be used by the expansion interface card to drive the data bus transceiver and to insert wait states in accesses to the I/O Expansion Box. The PC*jr* system board also pulls this line low whenever the device being accessed is contained on the system board.

Another additional feature on the PC*jr* I/O Channel Expansion bus is the AUDIO IN feature, which was discussed in the sound subsystem section. This input will allow for easy addition of speech and other audio devices to the PC*jr* system.

Figure 10 shows the I/O Channel Expansion connection specifications.

There are other differences between the PC*jr* system and the IBM Personal Computer and Personal Computer XT. All the technical information is contained in the PC*jr* Technical Reference manual.

Mode	Number of Colors	Compatible With Color/Graphics Adapter	Requires 64KB Memory and Display Expansion Option	Buffer Size
40-column alpha	16	Yes	No	2KB
160 x 200	16	No	No	16KB
320 x 200	4	Yes	No	16KB
640 x 200	2	Yes	No	16KB
80-column alpha	16	Yes	Yes	4KB
320 x 200	16	No	Yes	32KB
640 x 200	4	No	Yes	32KB

Figure 1. PCjr Video Modes

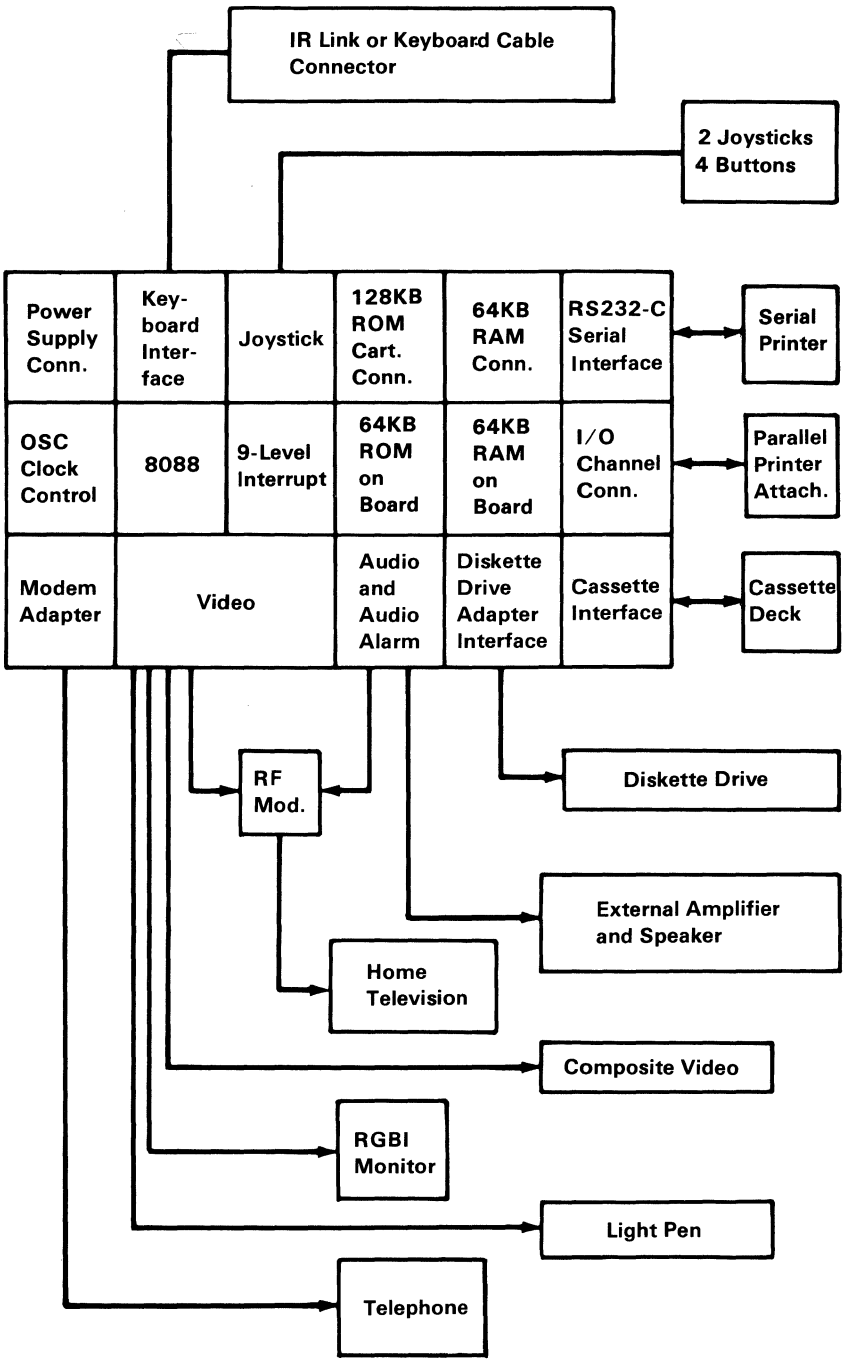


Figure 2. **System Board Block Diagram**

Main storage is shared between the video and the processor.

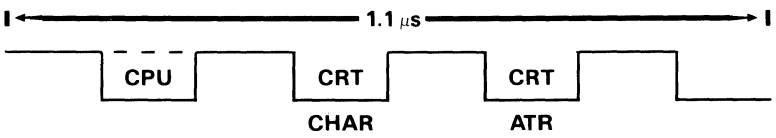


Figure 3. 40 Column Alpha Operation

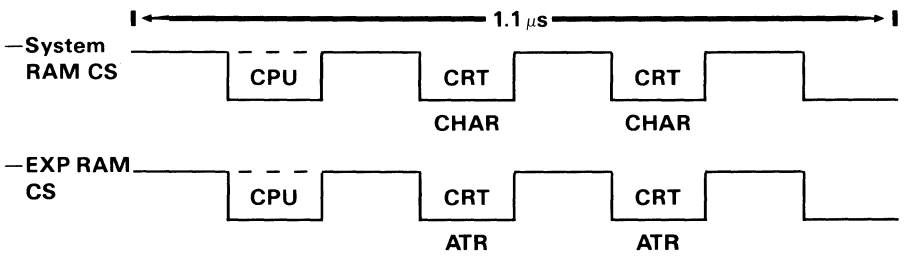


Figure 4. 80 Column Alpha Operation

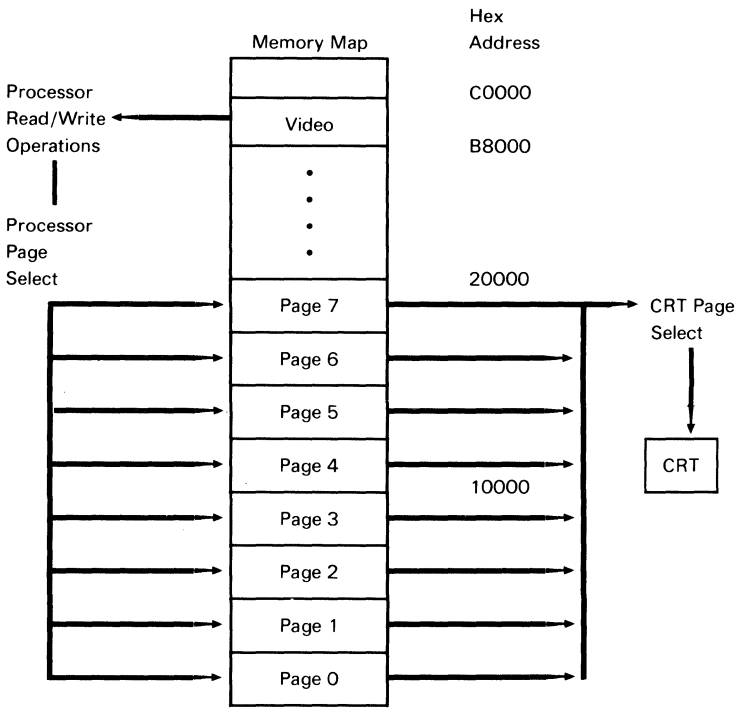


Figure 5. Video Color/Graphics Subsystem Memory Map

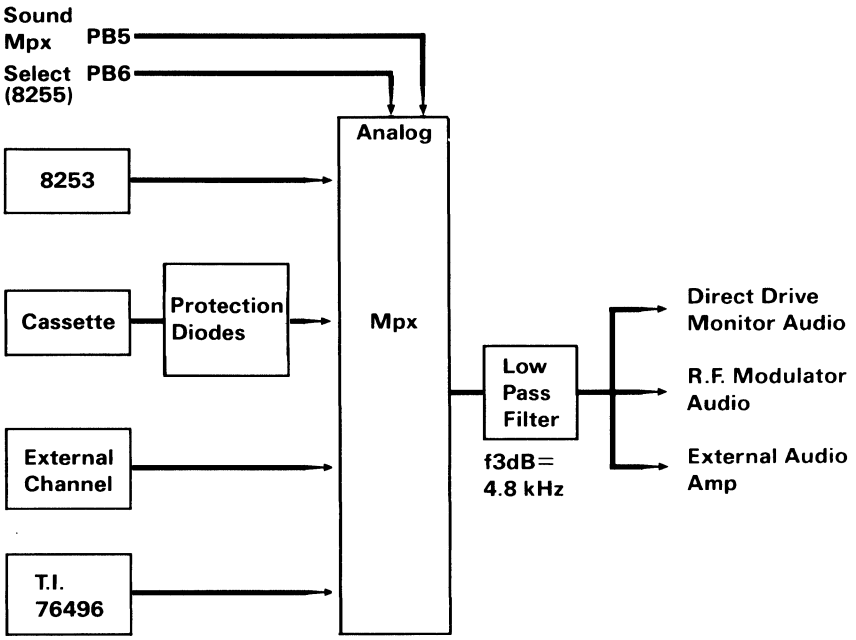


Figure 6. **Sound Block Diagram**

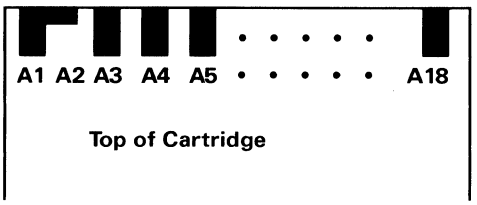


Figure 7. **Momentary Reset Land**

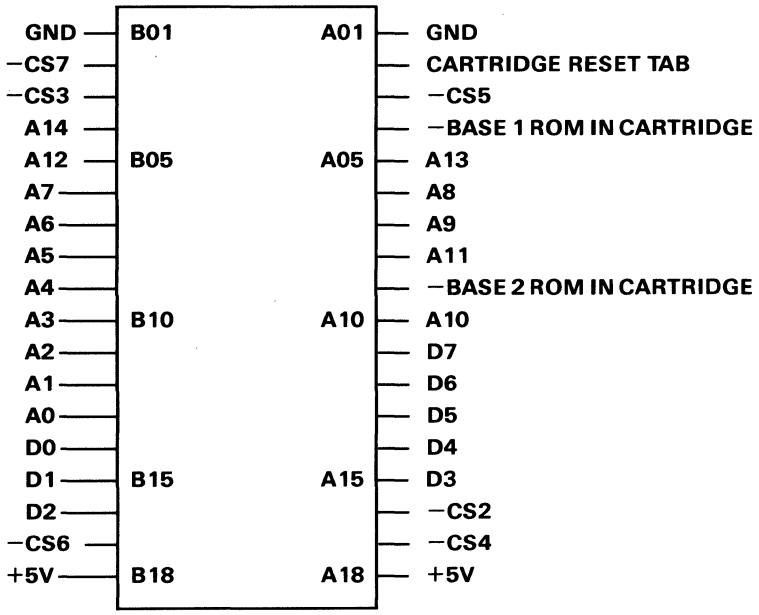


Figure 8. **Connector Specification**

ROM Chip Select	Hex Address Space	Typical Use
CS0	X	Not Used
CS1	X	Not Used
CS2	D0000-D7FFF	Optional Cartridge ROM #2
CS3	D8000-DFFFF	Optional Cartridge ROM #1
CS4	E0000-E7FFF	Standard Cartridge ROM #2
CS5	E8000-EFFFF	Standard Cartridge ROM #1
CS6	F0000-F7FFF	System Board ROM #2
CS7	F8000-FFFFF	System Board ROM #1

Figure 9. **ROM Cartridge Chip Select**

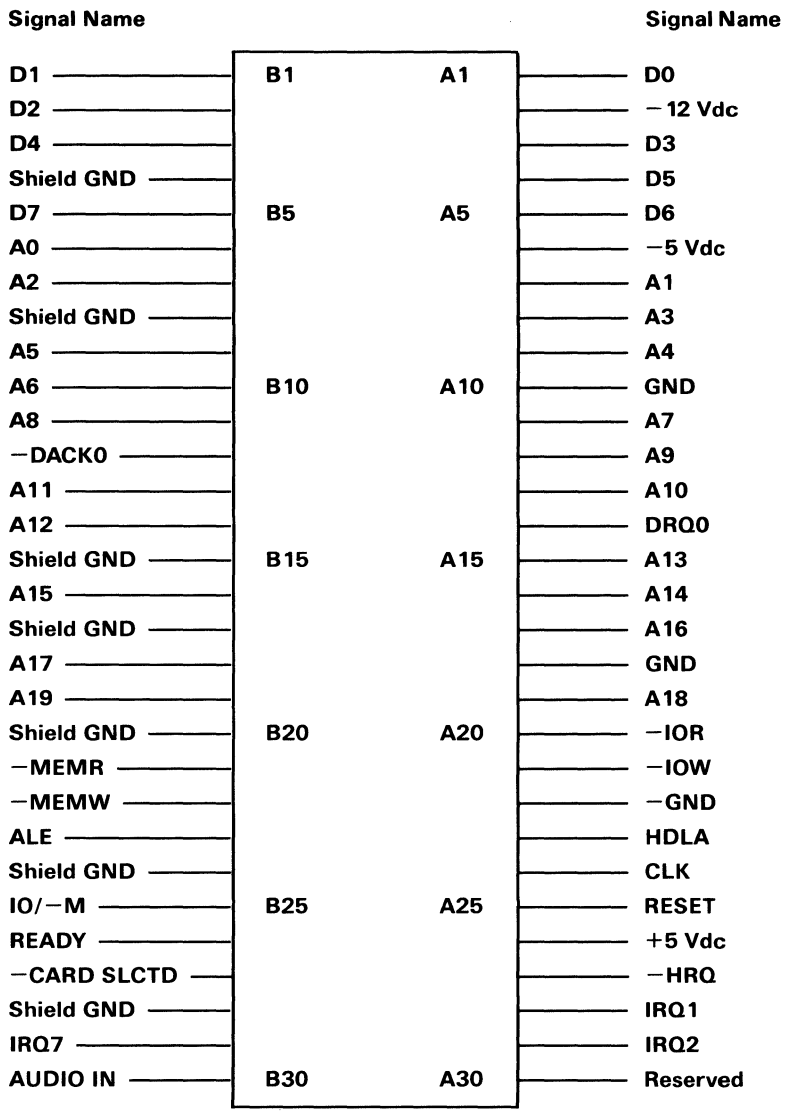


Figure 10. I/O Channel Expansion Connector Specification

IBM PC*jr* Compatibility Overview

The IBM PC*jr* differs from the IBM Personal Computer and IBM Personal Computer XT. Even though it is different, the IBM PC*jr* has a high level of programming compatibility with the IBM Personal Computers. It is possible to create PC*jr* software applications that can run without modification on other IBM Personal Computers. In order to create such programs or to assess if a current program is compatible, you must understand the differences among the Personal Computers in the IBM family and know the proper way to communicate with them.

Normally, it would be impossible for a program written for one computer to run on a different computer since the microprocessors would be different, and the language of the application could not be executed by different processors. In this case, the application would have to be re-written entirely in the language of the other processor. Since the IBM PC*jr* and the other IBM Personal Computers use exactly the same microprocessors (Intel 8088), most assembler language programs need not be modified.

This alone is not enough, since applications normally take advantage of the device services of a computer (BIOS) and the operating system (DOS 2.1). In order to allow for maximum program compatibility, the IBM PC*jr* has maintained all BIOS system interrupts and utilizes the same IBM DOS. This means that applications which use the BIOS and the IBM DOS interrupts on the IBM Personal Computers operate the same on the IBM PC*jr*.

Note: The BIOS micro-code of the IBM PC*jr* is not identical to that of the IBM Personal Computers. If an application bypasses the BIOS interrupt calls and directly accesses routines and/or storage locations in one system, it may not run in the other system. Some routines may be similar and some BIOS storage locations may be the same. It is strongly recommended that applications use only the BIOS and DOS interrupt interfaces in order to achieve compatibility in the IBM Personal Computer family.

Using the same language and the BIOS and DOS interfaces go a long way in achieving application compatibility. However, there are still several factors which need to be taken into consideration:

- Timing Dependencies
- Unequal Configurations
- Hardware Differences

Timing Dependencies

Programs running in user read/write memory normally run slower on the PC*jr* than on the IBM Personal Computers. Programs running in read-only memory (ROM) normally run a little faster on the PC*jr* than on the IBM Personal Computers. This may or may not cause a difference in execution time depending upon the application. Most applications are very I/O dependent in which case the execution time is not the critical factor and may not be noticeable. In other cases, the application runs the same but merely take a different amount of time.

If an application has critical timing dependencies, any timing differences (faster or slower) may adversely affect its usability. Using an application's program execution speed to achieve a desired timing can affect the application. In these cases, the application may need to be modified.

Note: It is strongly recommended not to depend on instruction execution speed to achieve specific application timing. The system timer can provide short interval timing for assembly language programs. Similar timing functions are available in BASIC.

Performance of specific I/O devices (such as diskette or printer) may also differ between the PC*jr* and the other IBM Personal Computers. You should also avoid using timing of any I/O device as a dependency for the application.

Unequal Configurations

In designing an application to run on both the IBM PC*jr* and the IBM Personal Computers, you need to make sure that the required hardware configuration is available on all machines. This means the application's minimum requirements are met by all IBM Personal Computers.

Hardware Differences

To be able to run on either computer without change, an application utilizing a specific I/O device must have access to identical devices (or devices with identical operating characteristics and interfaces). The IBM PC*jr* and the IBM Personal Computers have compatible I/O device capabilities.

Figure 11 and Figure 12 list the hardware features and I/O devices supported by the IBM PC*jr* and the IBM Personal Computers and summarizes the differences:

Device	Personal Computer	Personal Computer XT	PCjr	PCjr Comments
Maximum User Memory	640KB	640KB	128KB	Shares user RAM with Video Buffer
Cordless Keyboard	No	No	Yes	Scan codes compatible and full 83 key capability
83 Key Keyboard	Yes	Yes	No	
Diskette Drive	Yes	Yes	Yes	Compatible, but different addresses and no DMA support
Hard Disk	No	Yes	No	
Parallel Printer	Yes	Yes	Yes	Compatible
RS232 Serial Port	Yes	Yes	Yes	Compatible, hex2F8 address Interrupt Level 3, Baud-Rate-Frequency divisor difference
Game Control	Yes	Yes	Yes	Compatible interface with potential timing differences
Cassette	Yes	No	Yes	Compatible
Internal Modem	No	No	Yes	Compatible to PC Serial Port hex 3F8 address, Interrupt Level 4, frequency divisor difference
IBM Monochrome Display	Yes	Yes	No	

Figure 11. IBM PCjr and IBM Personal Computers Comparison(Part 1 of 2)

Device	Personal Computer	Personal Computer XT	PCjr	PCjr Comments
Color Graphics and Display	Yes	Yes	Yes	Compatible, with some register differences and enhancements
Light Pen	Yes	Yes	Yes	Compatible
Attachable Joystick	Yes	Yes	Yes	Compatible
8253 Timer (time of day)	Yes	Yes	Yes	Compatible
8259 Interrupt	Yes	Yes	Yes	Some difference in interrupt levels
Internal Sound	Yes	Yes	Yes	Compatible but less frequency response
TI 76496 Sound	No	No	Yes	
ROM Cartridge Interface	No	No	Yes	
Future I/O ROM Architecture	Yes	Yes	Yes	Compatible

Figure 12. IBM PCjr and IBM Personal Computers Comparison(Part 2 of 2)

The hardware differences between the IBM PC*jr* and the IBM Personal Computers may lead to incompatibilities depending upon the specific application. Once again, if your application maintains an interface to the Personal Computer family at the BIOS and DOS interrupt levels, then all hardware differences are handled transparently to your application. If your application goes below the BIOS level and directly addresses the hardware, then there could be an incompatibility.

User Read and Write Memory

Memory difference can be a problem even with programs written for the same computer, if the available memory is not the same from one machine to the next. Thus, the deciding factor is to state what the minimum memory requirement is for the application, and require that amount on the computer in question.

It is important to understand the memory aspects of the IBM PC*jr* in relationship to that of the IBM Personal Computers. The IBM PC*jr* can be configured for 64K bytes or 128K bytes (with memory expansion). However, this user memory is not all available to the application. The IBM PC*jr* video architecture utilizes a minimum of 16K bytes (in graphics mode) and 2K bytes (in alpha numeric mode) for the screen buffer. Therefore (in graphics mode), the IBM PC*jr* really has 48K bytes or 112K bytes (with memory expansion) available for system and application software. This is not the case with the IBM Personal Computers, since the color graphics adapter contains a separate 16K byte screen buffer. Thus, a 64K byte Personal Computer with color graphics (extra 16K bytes) is an 80K byte system compared to a 64K byte IBM PC*jr*. The IBM PC*jr* also has graphic enhancements which allow more than the 16K bytes to be utilized for

video screen buffers. If these enhanced features are used in an application, then even less is available for user memory.

Another aspect of available memory is the amount taken away by operating systems and language interpreters. In the case of the IBM DOS, both the IBM PC*jr* and the IBM Personal Computers support the same DOS. If your application requires the BASIC interpreter, then there may be a difference. The IBM Personal Computer Cassette BASIC resides entirely in the system ROM, taking no user memory. However, Disk BASIC or Advanced BASIC utilizes 11K to 26K bytes respectively (depending on DOS version), of user memory. In the IBM PC*jr*, Advanced BASIC capabilities (cartridge BASIC) reside in ROM, taking no user memory.

As you can see, many factors enter into the amount of memory available to a user. The most frequent comparison is that of the assembler language or compiled application using a 16K-byte screen buffer operating under DOS 2.1. In this case, an application requiring 64K bytes of user memory on an IBM Personal Computer cannot run on the IBM PC*jr* due to the video buffer usage of 16K bytes. Also, any application requiring more than 112K bytes of user memory with DOS 2.1 on the IBM Personal Computers cannot run on an IBM PC*jr*.

Diskette Capacity and Operation

Since the IBM PC*jr* maximum stand-alone configuration is one diskette drive with a maximum capacity of 360K byte diskette storage, an IBM PC*jr* application is either limited by this diskette capacity or is impacted by the user having to change diskettes more frequently. The IBM Personal Computers can have multiple diskette drives with a capacity of 360K byte diskette storage each or even possess hard files with a much larger disk storage capacity. This capacity difference may or may not be a concern depending upon the specific application.

In terms of diskette interfacing, the IBM PC*jr* and the IBM Personal Computers both utilize the NEC μPD765 floppy diskette controller, but with different hardware addresses, and the IBM PC*jr* does not operate through direct memory access (DMA). Since the IBM PC*jr* does not have DMA capability, application programs cannot overlap diskette I/O operations. When diskette I/O takes place, the entire system is masked (operator keystrokes and asynchronous communications cannot take place). Therefore, the application must insure that asynchronous operations do not take place while diskette I/O is active.

IBM PC*jr* Cordless Keyboard

The Cordless Keyboard is unique to the IBM PC*jr*. Even though it does not possess all 83 keys of the IBM Personal Computers' keyboards, it does have the capability to generate all of the scan codes of the 83-key keyboard.

The additional functions available on the PC*jr* are shown in Figure 13.

PCjr Special Functions	Required Key Combinations
Shift screen to the left	Alt+Ctrl+cursor left
Shift screen to the right	Alt+Ctrl+cursor right
Audio Feedback (System clicks when a key is pressed.)	Alt+Ctrl+Caps Lock
Customer Diagnostics	Alt+Ctrl+Ins

Figure 13. IBM PCjr Special Functions

Since all scan codes can be generated, any special application requirements can be met on the Cordless Keyboard.

The highest level of compatibility to interface to keyboards is through BIOS Interrupt hex 16 (read keystroke). Below that level is risky since there are hardware differences between the PC*jr* keyboard and the IBM Personal Computers' keyboards. The PC*jr* system utilizes the Nonmaskable Interrupt (NMI) to deserialize the scan codes and pass them to Interrupt hex 48 for compatible mapping to 83-key format. Interrupt level 9 remains a compatible interface for 83-key scan code handling. It is not recommended to replace interrupt level 9 even though a high degree of compatibility is maintained. If necessary, analyze this architecture carefully.

Color Graphics Capability

The IBM PC*jr* color graphics architecture is quite different from that of the IBM Personal Computers. The main difference (as previously discussed) is that the video buffer is taken from main user memory rather than having separate memory for video (as in the IBM Personal Computers). Normally, this would be an incompatibility since applications directly address the color graphics buffer at hex B8000. However, the IBM PC*jr* has special hardware to redirect hex B8000 addressing to any specific 16K-byte of its user memory. The IBM PC*jr* defaults the video buffer to the high end 16K-byte block of user memory and applications can continue to address the video buffer at hex B8000. In addition, all IBM Personal Computers' color graphics adapter modes are BIOS compatible and memory structure (bit map) compatible. These modes are shown in Figure 14 and Figure 15.

Modes	Requirements
Alphanumeric:	
40x25 BW	None
40x25 Color	None
80x25 Color	Note
80x25 BW	None
Graphics:	
320x200 4 Color	None
320x200 BW	None
640x200 BW	None
Note: PC*jr* requires the 64KB Memory and Display Expansion.	

Figure 14. Modes Available on the IBM Personal Computers and IBM PCjr

In addition the IBM PC*jr* provides some new enhanced graphic modes which are not available to the IBM Personal Computers.

Modes	Requirements
Graphics:	
320x200 16 Color	Note
640x200 4 Color	Note
160x200 16 Color	None
Note: PC*jr* requires the 64KB Memory and Display Expansion.	

Figure 15. Modes Available Only on IBM PCjr

The IBM PC*jr* and IBM Personal Computers utilize the 6845 controller, but the hardware interface is not completely the same. Hardware addresses hex 3D8 and hex 3D9 are not supported by the IBM PC*jr* video interface. Requests using these two addresses are not honored.

Also there are differences in the actual video used by the hardware. BIOS maintains compatibility by using the appropriate PC*jr* video parameters (addressed through Interrupt hex 1D) and maintains all video calls (through Interrupt hex 10). Applications can still specify video parameter overrides by modifying Interrupt hex 1D to address their own parameters; however, since there are hardware differences the recommended approach is as follows:

1. Copy the original parameters from the BIOS of the system.
2. Change only those parameters desired.
3. Consider the specific video differences between systems.

Other differences to be aware of are:

- The IBM PC*jr* defaults the colorburst mode to off, whereas the IBM Personal Computers default colorburst to on. Thus applications should not assume either default but set colorburst mode (through BIOS call) to the desired setting.
- The IBM PC*jr* video supports a full gray scale capability which the IBM Personal Computers do not.
- There can be some differences between the IBM Personal Computers and the IBM PC*jr*; especially when color mixing techniques are used.
- The IBM PC*jr* defaults the screen width to forty columns, whereas the IBM Personal Computer defaults the screen width based on switch settings. Thus applications should not assume that screen width is eighty, but set the desired width.

Black and White Monochrome Display

The IBM PC*jr* does not support the IBM Personal Computers black and white monochrome display. Programs which directly address the IBM Personal Computers monochrome display are not compatible. For example, any direct addressing of the B&W video buffer at hex B0000 is not redirected by the IBM PC*jr*. Applications should support Personal Computer video capabilities through BIOS, and the video buffer address should be either transparent to the application or the address is provided indirectly in the BIOS data area.

RS232 Serial Port and IBM PC*jr* Internal Modem

The IBM PC*jr* serial port address is hex 2F8 and is associated with hardware Interrupt level 3. This is compatible with a second Asynchronous Communications Adapter on the IBM Personal Computers. The Internal Modem address is hex 3F8 and is associated with Interrupt level 4. This is compatible with the first Asynchronous Communications Adapter on the IBM Personal Computers. It is important to note that when the IBM PC*jr* has the Internal Modem installed it is logically COM1 and the RS232 serial port is logically COM2 in BIOS, DOS, and BASIC even though its address is still hex 2F8 using Interrupt level 3. Other hardware differences on the PC*jr* serial devices are:

- A different frequency divisor is needed to generate baud rate. This is transparent to applications using BIOS to initialize the devices (Interrupt Hex 14).
- No ring indicate capability on the RS232 serial port.
- Asynchronous communications input cannot be overlapped with IBM PC*jr* diskette I/O. Since diskette I/O operates in a non-DMA mode any asynchronous data received during diskette activity may be overun (and lost). DOS applications must insure that no diskette activity is active while receiving asynchronous communication data. This can be done by pacing the asynchronous device (tell it to hold from sending). The ASCII characters XOFF and XON are frequently used by some host computers for this purpose.

Summary

In summary, the IBM PC*jr* is a member of the IBM Personal Computer family by way of its strong architecture compatibility. The highest degree of application compatibility can be achieved by using a common high level language, and/or accessing the system only through BIOS and DOS interrupts. It's not recommended to go below the BIOS level even though there are other hardware compatibilities. When it is necessary to design for particular computer differences, the application should determine at execution time which particular computer it is using. This can be done by inspecting the ROM memory location at segment address hex F000 and offset hex FFFE for the following values:

hex FF = the IBM Personal Computer
hex FE = the IBM Personal Computer XT
hex FD = the IBM PC*jr*

Once determined, dual paths would handle any differences.

Cartridge BASIC

Cartridge BASIC is a Superset of BASICA 2.1

BASIC is an interpretive programming language that allows direct interaction of commands and language statements. The IBM PC*jr* BASIC interpreter is structured in two functional levels: Cassette and Cartridge.

Cassette BASIC not only provides the necessary cassette tape recorder input/output instructions, but also a high level of support for display, keyboard, printer, light pen and joysticks, and a full complement of editing, logic, math and string functions. Cassette BASIC is provided with each computer in the form of built-in Read Only Memory, and is operational when the system is first turned on.

Cartridge BASIC is optional in the form of Read Only Memory (ROM) built into a cartridge. Cartridge BASIC extensions to the language provide a full set of instructions, commands and built-in functions. Most of these can be used with or without DOS present. The cartridge does support DOS, the structured directories, date and time.

Cartridge BASIC also supports the following new PC*jr* facilities:

Enhanced Graphics Support

- Four additional screen modes
  - 160x200, 16 colors
  - 320x200, 4 colors, variable palette
  - 320x200, 16 colors
  - 640x200, 4 colors
- PALETTE and PALETTE USING statements
  - New statements allow control of the hardware palette
- CLEAR
  - Enhancements to CLEAR statement allow you to increase/decrease the size of your memory (default is 16K)
- All graphics statements support the new screen modes

Enhanced Sound Support

- Multi-voice sound
  - SOUND and PLAY statement support up to three voice sound
  - Must issue SOUND ON command to use sound enhancements
- Volume control
  - Optional parameters on SOUND and PLAY enhancements
- Noise
  - New statement allows you to generate periodic and white noise
  - Good for games and special effects

Terminal Emulation

- Supports simple RS232 asynchronous communications
- Supports IBM internal modem or other external modem via serial port

Compatibility Guidelines for Application Development

This article summarizes the compatibility guidelines for application development. IBM suggests that you use these guidelines in your program development. These are guidelines only. They are not guarantees to ensure compatibility nor are they prohibitions of any kind. You may write a program in any manner you wish. However, if you follow these guidelines, your program will have a much better chance of running correctly on the different systems in the IBM Personal Computer family.

This article emphasizes how the IBM PC*jr* affects application programs.

For the purpose of simpler notation, all references to the IBM Personal Computer in this article also apply to the IBM Personal Computer XT. One distinction between the Personal Computer and the Personal Computer XT appears at the end of the article.

1. High Level Languages

   - IBM-supported languages — BASIC, FORTRAN, COBOL, and Pascal — are the best choice for writing compatible programs. IBM will make every effort to ensure that these high level languages continue to work on all systems in the Personal Computer family.

     The Pascal and FORTRAN compilers will not run on the IBM PC*jr* because there is not enough memory. However, programs that were compiled will run on the PC*jr*, as long as they do not exceed the memory capacity of PC*jr*. This means that you can do program development on an IBM Personal Computer, and then transfer the resulting program to an IBM PC*jr*.

   - A high level language program must be well-behaved. If a program directly uses features of the hardware, that program may not be transportable. For example, if you use hardware specific commands such as IN, OUT, PEEK and POKE, or if you use assembly language subroutines, you should follow the assembly language rules given below.

   - Any program that requires precise timing information should obtain it through a DOS or language interface (e.g., TIME$ or TIMER in BASIC). If the program requires greater precision, it should use the assembly techniques shown below. Using program loops for timing may prevent a program from migrating to an otherwise compatible system with different performance.

     Because system memory is shared with display memory in the IBM PC*jr*, a program in PC*jr* RAM will probably run slower than the same program running on the IBM Personal Computer. This means that all timing dependent programs intended for both the IBM Personal Computer and the IBM PC*jr* must use external timing information.

2. Assembly Language Programming

   - Assembly language programs should perform I/O through the ROM BIOS or through DOS function calls.

     - A program should use software interrupts to access these functions. This removes absolute addressing from the program, and only the interrupt number is required.

       The IBM PC*jr* has retained the physical addresses of most of the IBM Personal Computer's ROM BIOS entry points, but your program should not rely on them.

     - The BIOS and the operating system can hide hardware differences. As new devices are added to the system, IBM may modify the ROM BIOS to maintain a consistent programming environment.

       For example, the IBM PC*jr* does not have a Direct Memory Access (DMA) controller. If your program directly controls the diskette adapter, and with it the DMA controller, it won't work on the IBM PC*jr*. However, the diskette BIOS routine (accessed through interrupt 13H) has the same calling interface. A program that uses the BIOS interrupt for the diskette will still run correctly.

     - At system initialization, the ROM BIOS builds I/O parameter tables for some of the devices. (The diskette and video adapters are two examples.) If you want to modify a BIOS table for use by your

program, you should first make a copy of the table, make your changes in the copy, and then change the vector table to point to the modified table. When your program finishes, you should restore the vector table to its original value. This will maintain a consistent structure for BIOS, and will allow your program to restore the system correctly before it terminates.

IBM has changed the BIOS tables in support of the PC*jr*. For example, some of the BIOS video parameters dealing with horizontal and vertical timings have changed. Unless you copy the existing table before making your changes, you may unintentionally change other portions of the table.

- ROM BIOS and DOS do not provide for all devices and functions. Following is a list of the allowable I/O operations, that IBM will attempt to keep compatible in future systems.

  - A program can control the sound by using port 61H and the sound channel of the Timer/Counter (see below).

  - A program can control the 8253 Timer/Counter channels 0 and 2, ports 40H, 42H and 43H. Under no circumstances should a program change the value in port 41H of the IBM Personal Computer, since this channel of the timer controls the dynamic memory refresh.

  - The IBM PC*jr* uses a different method of refreshing memory, and channel 1 is used for extended timing and keyboard deserialization. Channel 0 provides the time-of-day interrupt, and can also be used for timing short intervals. Channel 2 of the timer/counter is the output for the speaker and cassette ports. This channel may also be used for timing short intervals, although it can't interrupt at the end of the period. The PC*jr* uses the same input clock frequency to these timer channels as does the IBM Personal Computer. Assembly language programs that require the measurement of short intervals should use these timers.

  - A program can use port 201H to control the game control adapter. This applies to both the IBM Personal Computer and the IBM PC*jr*.

    Note: Programs should use the timer, rather than a program loop, to determine the delay on a resistive input.

  - You can use the video status port (3BAH or 3DAH) to determine horizontal and vertical retrace times.

    The PC*jr* video controller is significantly different in its implementation than the IBM Personal Computer Color/Graphics Adapter. This means that the programming for special display effects, such as animation, will be different for PC*jr*. In most cases, PC*jr* has extended capabilities to help the programmer. These facilities are unique to the PC*jr*, and are not available on other IBM Personal Computer.

  - You can program the Asynchronous Communications Adapter and the 8250 chip for interrupt driven communication. You should use ROM BIOS for the initialization of the 8250 — baud rate, parity, stop bits and data bits.

    The input clock on the IBM PC*jr* is not identical to the input clock on the IBM Personal Computer. If you use BIOS for initialization, you are guaranteed that the correct baud rate will be set. If you set it yourself, it may be incorrect on the IBM PC*jr*. All other functions are identical on PC*jr*.

  - A program can use the Interrupt Mask Register (IMR), port 21H, to selectively mask and unmask the hardware features.

    The IBM PC*jr* has maintained all of these hardware compatibility requirements.

- Absolute memory locations

  - Interrupt vectors (0H) — A program may change the interrupt vectors to point at different processing routines. When a program changes an interrupt vector, it should retain the original value. If the interrupt (hardware or software) is not directed toward this device handler, the request should be passed on to the original interrupt handler. You can build a chain of interrupt handlers in this fashion.

  - Video display buffers (B0000H and B8000H) — For each mode of operation defined by the video display BIOS, the

memory map will remain the same. For example, the bit map for 320x200 medium addressability graphics mode of the Color/Graphics Adapter will be retained on any future adapter card that supports that mode. If the bit map is modified, a different mode number will be used.

All IBM PC*jr* video modes that are identical to the Color/Graphics modes have an identical bit map and an identical BIOS interface. The new PC*jr* display modes have new mode values assigned to them (refer to the discussion of the COLOR statement in the IBM PC*jr* BASIC manual, page 4-63). Notice that the PC*jr* hardware provides a display paging mechanism that lets your program write display information into segment B8000H, even though the refresh buffer is located in the system memory. This feature lets you do direct display control in your program identically on both the IBM Personal Computer and the IBM PC*jr*.

- ROM BIOS data area (40:0) — Any variables in this area will retain their current definitions as long as it is reasonable to do so. However, it is possible that someday a variable may no longer have meaning in the system, and IBM may use the data area it occupies for other purposes. Therefore you should access ROM BIOS data variables through BIOS calls and not by direct addressing.

  The ROM BIOS data area has changed in the IBM PC*jr* to include some new variables. All old variables have retained their previous meanings.

- Timing Dependencies

  A program that requires timing information should use either the time-of-day clock or the timing channels of the 8253 timer. The input frequency to the timer will be maintained at 1.19 MHz, providing a constant time reference. Program loops should be avoided.

  The IBM PC*jr* timer/counter has identical characteristics to those of the IBM Personal Computer.

- Copy Protection

  Programs which use copy protection schemes should use the ROM BIOS diskette calls to read and verify the diskette.

  You can use any method to create the diskette. However, you should also consider how you are going to modify a large number of replicated diskettes if you have to add a copy protection methodology.

  The verifying program can look at the diskette controller status bytes in the ROM BIOS data area for further information about any embedded errors.

  The IBM PC*jr* uses a diskette controller similar to the one used in the IBM Personal Computer. However, since the PC*jr* has no DMA, the diskette programming is done differently. ROM BIOS effectively masks these differences. Notice also that a copy protection scheme that relies on instruction timing (rather than diskette rotation timing) will not work correctly on the IBM PC*jr*.

- Program Relocatability

  DOS programs must be relocatable. They should be insensitive to the size of DOS or their own load address.

  A program's dynamic memory requirement should be known, and should be contiguous with the load module. Don't assume that all of memory is available to your program — DOS 2.1 has memory allocation function calls that you can use. Also you should verify that your load module has sufficient memory in which to operate.

3. Machine Sensitive Code

   - Programs may use machine specific features, with the understanding that they will then be machine dependent. This may be a very reasonable choice for a program that desires the ultimate in performance or special effects. However, such a program will no longer be transportable to other machines in the IBM Personal Computer family.

   - You can write a program that uses machine specific features and still retains its compatibility. You can do this by writing the program for all types of machines that you intend to use it on. The program can test to determine which machine it is running on, and then configure itself appropriately. IBM has embedded a machine identification value in the ROM BIOS which is different for each machine:

0FFH for the IBM Personal Computer
0FEH for the IBM Personal Computer XT
0FDH for the IBM PC*jr*

- You can also use this same technique for I/O adapters in the system. The BIOS call that determines machine type can also tell you what devices are attached to the system.

In the future, if IBM adds new machines and I/O devices to the family, IBM will continue to provide methods for determining the specific machine type or I/O feature.

Personal Communications Manager- Modem Drivers

Application Description

The Personal Communications Manager is an easy to use communications application that enables users of any IBM Personal Computer, including the new PC*jr* to exchange information over standard telephone lines.

The product consists of three functional parts:

- Terminal emulation
- Electronic mail
- IBM PC word processor/editor access

The four goals which guided the design of the Personal Communications Manager are:

- Provide ease of use
- Provide a wide range of user functions
- Allow easy editing using all IBM Personal Computer editors
- Allow easy modification to use different modems

Easy modification to use different modems is the topic of this presentation. The application provides a well defined modular interface which allows definition and modification of six generic modem functions. Together with their associated subroutines and data, these six functions comprise a modem driver. Drivers are unique to a specific modem or a compatible group of modems and are accessed by the Personal Communications Manager as a .COM file, which is loaded when the application enters terminal emulation or electronic mail. A source program file for the driver of the IBM PC*jr* internal modem is provided as an example on the master copy of the program diskette. The name of this file is EXPM__MD.ASM.

Personal Communications Manager Supported Modems

The modems supported by the application as delivered are:

- IBM PC*jr* internal modem
- Microcom PCS or RX modems
- Hayes Smartmodems™

While the program and the description is easy to follow and to use, writing a modem driver to support an auto-dial modem requires a thorough knowledge of communications protocol, the modem to be supported and assembler language. Therefore this discussion is directed to programmers with these skills.

Required Modem Functions

The modem functions required by the Personal Communications Manager consists of three groups:

Minimum Modem Functions

All functions and options must be accessible under program control.

- Auto-dial
- Auto answer
- Data transparency (support sending/receiving of 8-bit binary data)
- Return connection speed

Note: If the modem is BELL 212A compatible, the type of connection established must be returned when the phone is answered, to allow the program to adjust the speed of the serial port.

- Set carrier detect lead (pin 8)

Note: The Personal Communications Manager tests the carrier detect lead to determine if it is connected. The value of the carrier detect lead is assumed to follow actual Carrier.

- 8250 compatibility (internal modems)

Note: The serial port of an internal modem must be compatible with the IBM Personal Computer Asynchronous Communications Adapter.

Personal Communications Manager Full Function Utilization

The functions and options in the second group are required to make full use of the functions in the Personal Communications Manager and are as follows:

- Auto-dial using both tone and pulse (tone and rotary dialing)
- Change speed and parity while connected
- Carrier timeout greater than 1.5 seconds

Note: If call waiting (a telephone feature available in some areas) has been installed on a phone line, a carrier detect timeout of less than 1.5 seconds will cause a modem to hang up when a second call attempts to access the line. Call waiting generates a pair of tones to inform the user a call is waiting. These tones interrupt carrier and garble the data going across the line. The electronic mail facility of the Personal Communications Manager detects these garbled data and resends them. The IBM PC*jr* internal modem has a two second carrier detect timeout which allows the program to wait out loss of carrier; it then continues resending the garbled data.

Programming Ease Requirements

The functions and options of the third group make the writing of the modem drivers easier.

- Terse mode

Note: If the modem supports a concise reply mode, it is recommended that this function be utilized. This will keep the driver from waiting for long modem responses. The driver must clear the ring buffers before returning.

- Change dial mode within the dial command

Note: If the modem supports mixed mode dialing, it will not be necessary to restructure the dial code.

Driver Description

The modem driver shown in the example is organized as follows:

Specification Documentation

Physical Control Block (PCB)

The PCB contains the variables used to pass communications parameters to the modem drivers from the main program.

Driver Functions

Description of six driver functions.

Serial Port Interface

The serial port interface is an expanded BIOS (basic input/output device system of the IBM Personal Computer) interrupt 14H handler. It has been expanded beyond sending/receiving and status. All serial port functions can now be accessed through this interface.

Definitions and Equates

- Macro definitions
- Public definitions
- Local definitions

Data Storage

- Public storage - Physical control block
- Local storage - Local flags and temporary data storage
- Modem commands - The command strings used to control the modem

Entry and Exit Codes

- Entry and exit codes for the driver including a dispatch table pointing to each function of the driver

Driver Functions

- The six functions of the modem driver

Driver Functions Subroutines

- Utility functions subroutines

Driver Functions

The six Personal Communications Manager modem driver functions are described below and are organized as follows:

- What is done by each function
- The implementation of the function in the driver example provided
- How the example implementation may be different for other modems
- Coding hints

It is recommended that the terminal emulation function of the Personal Communications Manager be used to debug a driver. The terminal emulation function can invoke each driver function, on demand, usually by a simple keystroke sequence.

P__INIT

P__INIT is used to initialize the modem and the serial port. In the example provided for the IBM PC*jr* internal modem, it performs the following using function 4 of the expanded interrupt 14H handler by initializing in the sequence shown:

- serial port
- ring buffers
- communications interrupt

Note: Notice that the WAIT subroutine is called whenever changes are made to the serial port. This allows any signal line changes to settle before allowing the program to continue.

P__INIT is also used to determine the speed and parity to be used for sending subsequent commands. This also serves to test if the modem is functional. If the test fails the carry flag is set indicating an error condition and a message Communications Device Failure is displayed. WAIT is used here again to allow the modem enough time to process commands.

In addition when the Personal Communications Manager is being used with an other modem, P__INIT should check the PCB for invalid communications parameters.

This will ensure that the selected parameters are correct. P__INIT is called when Terminal Emulation is entered.

P__CTRL

P__CTRL is used to reset the serial port and/or the modem to the values supplied in the PCB. The example program shows two functions of P__CTRL. The first builds the speed and format commands for the modem with the information specified in the PCB. The second builds a new RS232 line control byte for the serial port.

P__CTRL is called every time the program enters Interactive Mode.

P__CALL

P__CALL is used to dial the requested number and wait 30 seconds for carrier. When writing a P__CALL for use with another modem, if the modem does not accept spaces, dashes, or parentheses then P__CALL has to strip these characters out. If the modem does not automatically switch back to tone dialing after completing a dial, P__CALL can not assume a default and should set the mode explicitly at each dial. If the modem does not allow imbedded dialing mode commands within a dial command, P__CALL must be restructured to first look for a dial character and then set the dial mode before it sends any dial prefix.

P__CALL is called by issuing an Alt-d in Interactive Mode.

P__WAIT

P__WAIT is used to perform the following functions:

- Place the modem in an auto answer state
- Test for carrier and indicate a connection
- Test for 300/1200 connection and adjust the serial port line speed if necessary

In the example given for the IBM PC*jr* internal modem, P__WAIT only performs the first two functions described above. This is because the IBM PC*jr* internal modem does not run at 1200 baud.

P__WAIT is called by issuing the Alt-a command in Interactive Mode.

P__DISC

P__DISC performs the following functions:

- Places the modem on hook
- Cancel any pending modem command

In the example given for the IBM PC*jr* internal modem, both of these functions are accomplished by negating the Data Terminal Ready (DTR). Depending on your modem, lowering DTR may or may not provide both of these functions.

P__DISC is called three different ways.

- Issuing an Alt-h in interactive mode
- Escaping out of a dial command or user function
- Exiting Terminal Emulation to the Personal Communications Manager main menu

P__BRK

P__BRK is used to send a break out on the phone line. In the example given for the IBM PC*jr* internal modem, this is accomplished by sending the break command to the modem. If the modem doesn't have a break command, a 300 ms break can be generated by using function 9 in the interrupt 14H handler.

P__BRK is called by issuing an Alt-b in Interactive Mode.

IBM Corporation
Editor, IBM Personal Computer Seminar Proceedings
5Q9 / 237-3
Post Office Box 1328
Boca Raton FL 33432