The Two Faces of PCjr

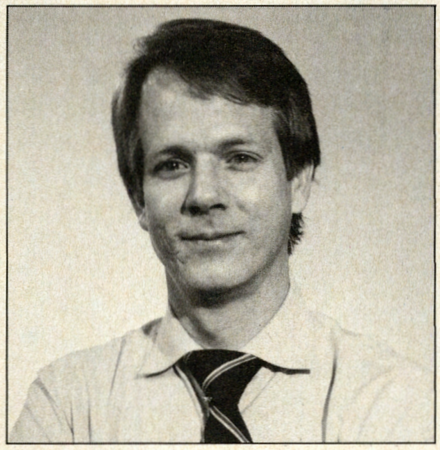
Like the Roman god Janus, IBM's newest computer stares in two directions: to the business and work environment of its big brother, the PC, and to the home setting.

When computer manufacturers like IBM work on new projects, they give them code names for working purposes. The code name for the original IBM PC reportedly was Chess. For the new IBM PC*jr*, IBM is said to have used a few code names, including Hercules, Pancake, Pigeon, and Sprite. Bringing out a new personal computer model calls for the help of outside companies, particularly software companies such as Sierra Online, which developed many of the first games for the new PC*jr*. To find out if any of these outside companies were leaking information, IBM gave each its own code name for the new computer.

I don't know what IBM's own internal code name for this wonderful new computer was, but I know what it should have been: Janus. Janus was the two-faced Roman god who stood guard over the new year with one face turned back to the old year and the other face turned forward to the new. To me, there isn't any better symbol of the meaning and significance of the PC*jr* than Janus.

Code Name: Janus

The PC*jr* is a computer that looks in two directions. It has two strong aspects that allow it to peer with a steady gaze into two quite different realms. One face looks into the world of the IBM PC as we have known it. The other looks into the world of home computing, a world that has previously belonged to computer companies like Atari, Commodore, Mattel, Coleco, and Timex/Sinclair.



Peter Norton

The "PC" face of the PC*jr* is thoroughly remarkable. When the PC first appeared, we all understood that it was a very powerful, capable, and versatile computer at what we thought was a modest price, between $3,000 and $5,000. When the XT appeared, extending the PC's capabilities for a price between $5,000 and $7,000, we were impressed by its cost-effectiveness too. At the time, the price/performance ratio of the PC and the XT seemed quite impressive. Yet the PC*jr* is priced at around $1,000 to $1,500 while giving us about 85 percent of the computing power of the PC and roughly the same proportion of its capability as well. That is completely remarkable.

(The reason the PC*jr* has less computing power than the PC is technically intricate: In the PC and XT, the display screen has its own dedicated memory, which can be read to refresh the display independent of the needs of the processor. The PC*jr*'s display, however, shares the same memory used by the computer's processor, and refreshing the display image steals memory cycles. This causes the processor's use of the memory to take about 50 percent longer than it would otherwise, and the extra time needed to use the memory adds, on the average, about 15 percent to the overall running time of each instruction the computer carries out.)

The other face of the PC*jr* is equally impressive. Home computers have traditionally provided a strong ability to play games. In the areas of home finance, education, and home-style word processing, however, the capabilities of home computers have been sadly lacking. Analysts of the home computing world have reported that a high proportion of old-style home computers have ended up on a closet shelf, simply because they weren't really very interesting or useful after the novelty wore off. The IBM PC*jr* shouldn't suffer from this problem.

The PC*jr* brings an unprecedented

amount of computing power and versatility into the home environment. Families that learn to make full use of the PC*jr* won't find it lacking in amazing abilities. It has enough computing power, graphic formats, and all-around flexibility to provide a dramatic improvement in home computing.

But the most important thing about the PC*jr*'s homeward-looking face is its connection to the face that looks toward the established world of the PC and XT. IBM's home game computer and IBM's cheapest personal computer are the *same* computer. This means that the computer user who works at home can run nearly all of the most important and useful programs that have been written for the other IBM personal computers.

The Volkswagen of PCs

Some practical factors need to be kept in mind when measuring the PC*jr*'s abilities against the PC's. Consider this analogy: A pickup truck can haul around more stuff than a Volkswagen. But if a person's main objective in owning a vehicle is to bring the groceries home, then a Volkswagen would serve just as well as a pickup truck could. The new PC*jr* can be compared to that Volkswagen and the PC to the pickup truck. An XT would be comparable to a moving van.

The truth is that the work most of us give our computers doesn't begin to stretch their capacity. If the work that you do with your computer really calls for an XT, you'll probably think of the new PC*jr* as nothing more than a toy. On the other hand, if you own a PC and the programs you use can get by in 88K of user memory, then the PC*jr* would probably be almost as useful to you as a PC.

The PC*jr* does have limitations when compared to the PC and XT, and two stand out: It has only one disk drive and only 128K of total memory. Most PCs have two disk drives, and an awful lot of PCs have 256K or more memory. Yet most programs for the PC have been designed to work with only one disk drive and only 64K. In the past, it may have seemed laughable that so many programs lived within these restrictions, but now we should be very grateful. Any program that followed a design guideline of 64K and one disk drive is likely to run beautifully on the PC*jr*.

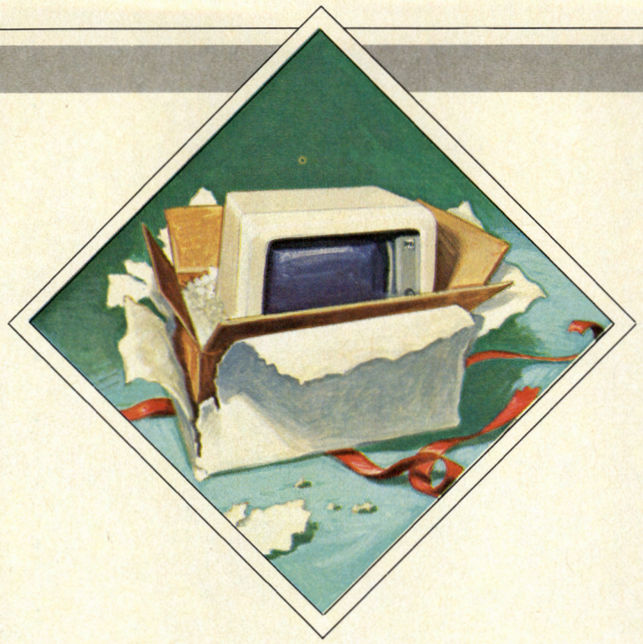
IBM's new PC*jr* can be used in the office and at home.

This means that the PC*jr* will run most of the PC's programs very nicely. Most of the programs that can't run on the PC*jr* are programs you probably wouldn't think of trying on the PC*jr* anyway. If you really need a moving van, you wouldn't even think of driving a Volkswagen. If you're considering getting a PC*jr*, then a PC*jr* is probably enough computer for your needs.

Looking at things this way, the new PC*jr* has all, or nearly all, the capacity and flexibility that most of us who are already using PCs need. For someone who is just starting out in personal computing with the IBM PC*jr*, things look even better. These new users can tailor their choices of programs and applications to fit the abilities of their computer.

All in all, the PC*jr* looks like a remarkable machine for its capabilities and an astounding one for its price. The microcomputer world has looked with amazement at the impact of the original PC on personal computing, it seems likely that the PC*jr* will make an even bigger splash, and that the waves from that splash will spread much further. There is great excitement ahead. Stay tuned. ■

MIKE EDELHART



Meet The New Members

This issue of PC Magazine explores the three latest additions to the IBM personal computer family: the XT-370 Workstation, the PCjr, and the 3270-PC.

First there was the PC.

When IBM produced its personal computer, the machine stood far apart from the rest of Big Blue's activities. It was a kind of stepchild, not unloved but not entirely in step with the rest of IBM's computing clan. Apart from building and marketing the machine, IBM did little to find it a role in the computing world. The PC's character and many of its capabilities were provided by outside vendors rather than by the computer's corporate parent.

Then IBM introduced the XT. It was as if PC had suddenly acquired a tough older brother. The introduction of this more powerful version of the IBM personal computer was an important manifestation of the company's belief that personal computers are serious machines with important business roles to fulfill. The PC-XT team could play a much larger role than the PC alone, and the addition of an "older brother" brought the PC closer to the family fold.

Now IBM has expanded the PC family even further with the introductions of the PC*jr*, the XT-370 workstation, and the 3270-PC. These new products represent the full integration of PC into the IBM scheme of things. No longer is the personal computer an odd offspring. Now, in fact, it appears as the heir apparent to the role of the IBM business workhorse machine. And the new additions to the PC family extend the impact of IBM personal computing all the way from the rec room at home to the data processing room at major corporations and institutions.

If the PC by itself represented the standard by which microcomputers were judged over the last two years, the PC family will be the standard by which integrated computer systems will be judged from now on. IBM will no longer merely be selling a good micro machine; it will be marketing a system of micros with such variable and flexible capabilities and applications that the possibilities almost defy description.

The possible combinations are mind

Illustrations: Joel Naprstek

From left: the IBM XT-370 workstation, the PCjr, and the 3270-PC.

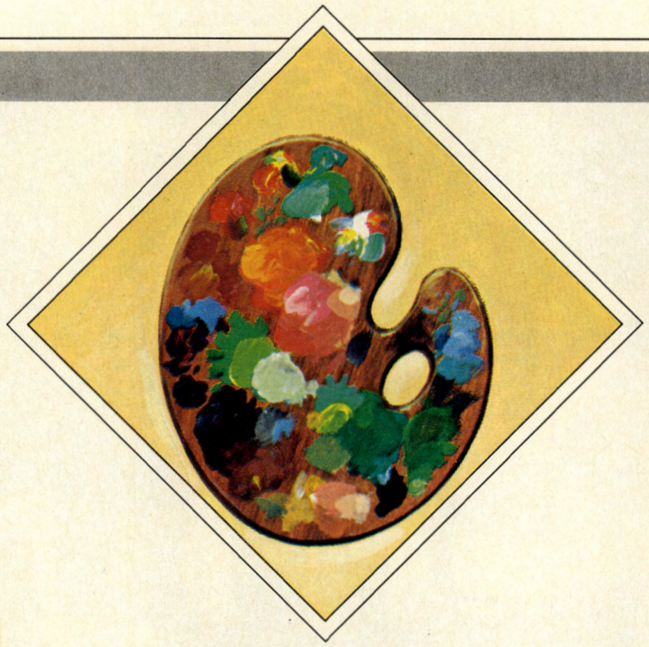
boggling: PC*jr*s at home or on secretaries' desks, with PCs and XTs in executive offices, laboratories, and sales stations, and 3270s at branch offices, communications posts, and overseas outlets, plus 370 workstations at key points in the processing chain of command. An individual or company can now have machines of different characteristics and power in different situations, all richly compatible with one another and which can trade information up as the information grows more complex or shift it down into smaller, easier-to-use units. Customers, if they so desire, can arrange to purchase an entire family of computers that can be serviced by one person and that require one basic set of training guidelines.

Beyond these straightforward commercial considerations, though, the new PC family is important because it is a first taste of the computing future. Today's individual, isolated, proprietary micros are destined to give way to networked machines that can speed information wherever it needs to go and can speak to one another and to other devices at different levels in the computing chain. The distinctions between micro, mini, and mainframe will continue to blur as machines become more flexible and powerful at all levels. Personal computing will shift from today's one-man, one-machine situation to a future in which the individual will be able to work through many machines, depending upon the best way to solve each problem.

By spreading IBM personal computing across the broad face of individual and business situations, the PC family is leading the way into this age of enhanced communications and capability.

In short, these machines are transforming the way we use and think about personal computers even more than the PC did on its own. That is why we at *PC Magazine* felt that the new PC family deserved special coverage. Meet the new members of the IBM clan; you'll be reading about them and using them often in the weeks ahead. ■

Photograph: Dennis Bettencourt



Screening The PC*jr*'s Color, Video, And Memory Options

Offering a choice of colors and screen memory sizes, PC*jr*'s video display screen is a rich and effective mixture of new and old technology.

Dennis Bettencourt

SCREEN  OPTIONS

The most remarkable thing about IBM's new personal computer, the PC*jr*, is how it manages to be both the same as and very different from its predecessors, the PC and XT. Nowhere in this remarkable new computer is this more strikingly apparent than in the video display screen. We'll take a quick tour through what is old and new in the PC*jr's* display screen and how the two are married so effectively.

The PC*jr's* display is rich and complicated with many parts to it. It offers new graphics modes, more flexible use of color, and an intriguing new way of using memory, which played a major part in keeping down the cost. PC*jr's* new circuitry differs from the PC's, and its new video modes have an effect on its computing power. Finally, there are all those video plugs on the back of the PC*jr*. We'll take a look at all these factors and explore what they do.

Let's begin where IBM began—doing its darnedest to imitate the PC. PC*jr* is designed to act like a PC with the color graphics board installed. The original PC design had eight different video modes, one specifically for the monochrome adapter and the other seven for the color graphics adapter. Programmers familiar with the PC's built-in ROM BIOS service programs know the color graphics modes as video modes 0 through 6 and the monochrome mode as 7. A color graphics adapter could be put into any of the PC's seven modes using the BASIC statements shown in Figure 1.

Since the PC*jr* is designed to mimic an ordinary color graphics board PC as much as possible, all the original color graphics modes work as they originally did, as do the BASIC statements that control them. But in addition, there are three new graphics screen modes, with accompanying new BASIC statements to control them: SCREEN, COLOR, and PALETTE.

To the PC*jr's* innards, there are three new video modes, referred to as modes 8, 9, and 10, respectively. Each of these new modes extends the PC*jr's* graphics capability beyond what the PC's original color graphics board provided. Mode 8 gives us *low*-resolution graphics, with 160 dots, or pixels across the standard 200 lines that fill the screen from top to bottom. By contrast, the other two resolutions, which are not new, given us 320 pixels across for medium resolution, and 640 dot positions on each line for high resolution.

Mode	BASIC Statements	Description
0	SCREEN 0,0 : WIDTH 40	Text, 40 column, black and white
1	SCREEN 0,1 : WIDTH 40	Text, 40 column, color
2	SCREEN 0,0 : WIDTH 80	Text, 80 column, black and white
3	SCREEN 0,1 : WIDTH 80	Text, 80 column, color
4	SCREEN 1,1	Medium-resolution graphics, color
5	SCREEN 1,0	Medium-resolution graphics, black and white
6	SCREEN 2	High-resolution graphics, black and white
7 (The monochrome mode, not used by the color graphics adapter or the PC*jr*)		

Figure 1: *the original eight color graphics modes in BASIC.*

A Choice of Colors

In the new low-resolution mode 8, all of the PC*jr's* 16 color options can be used at once. In the original PC, only four colors could be used at a time in the graphics modes, and we couldn't even choose our own four. But now, with low-resolution mode 8, we have a free and complete choice of colors, as well as a new resolution.

You might be wondering why it is useful to have a new lower-resolution mode for creating graphics. The point isn't simply to extend the amount of color we can use, as we'll see in a moment. The real reason is to make graphics work better with TV sets. TV sets have a fairly poor picture-making resolution compared with computer monitors. They are not very good at handling pictures made up of 80 columns of text nor those that use either medium or high graphics resolution; it's for TV sets, in fact, that IBM personal computers originally were equipped with the 40 column display mode. With the higher-priced PC and XT models, TV sets were seldom used for computer display screens—after all, anyone who could afford a PC or XT could probably afford a proper monitor. But now, with the low-priced PC*jr*, TV sets will be widely used—that's why a low-resolution graphics mode is so important.

Two other new modes, 9 and 10, add more color to the existing medium and high resolutions. Mode 9 is 320-by 200-pixel medium resolution, but with full use of 16 colors, whereas the old medium resolution mode 4 used only four. Mode 10, 640-by 200-pixel high resolution, selects four colors in a palette from the full complement of 16, whereas the old high-resolution mode gave us only two colors (black and white).

We have been describing these three new graphics modes in terms of their internal ROM BIOS video mode numbers. In BASIC, they are controlled by the SCREEN statement's mode parameter. Those who have dug into PC have learned that the computer looks a little different from the perspective of BASIC, and this is also true for these new video modes. In the BASIC SCREEN statement, there are four new modes, numbered 3 through 6. SCREEN 3 mode is the new low resolution (video mode 8); SCREEN 5 mode provides the new medium resolution with more color (video mode 9); SCREEN 6 corresponds to the new high resolution with more color (video mode 10). SCREEN 4, which we skipped over, is something of an oddity—it's the old four-color, medium resolution, which we always had with SCREEN 1, but the color is controlled a bit differently.

Color, in fact, figures in the PC*jr* in ways that are quite new. For one thing, the PC*jr* has more colors than the PC. As we've seen, there are now 16 instead of four in medium resolution mode and four instead of two in high. In addition, we can now select which colors to have at our command when we are using a 4-color mode. In the original circuitry, the colors in the PC's palettes were mostly fixed; we could chose one of the four colors used in the palettes, but the other three were preselected for us. The new color graphics circuitry, on the contrary, is completely flexible: We can load any color selection that we want to into the palettes.

Arranging the Palette

To control the new color possibilities, PALETTE and PALETTE USING statements and an extended COLOR statement were added to BASIC. The function of the PALETTE statements goes beyond defining which colors will be available in the four-color modes. In the 16-color modes, the PALETTE statements can be used to rearrange the colors, so that when a program asks for color 1 (which is normally blue) it might actually get color 4 (red), and so on. Changing PALETTE can instantly change the colors on the screen or make things appear and disappear. In fact the new color tricks are so extensive and interesting that they could use a whole article to themselves just to cover the basics of what we can now do with color.

As we hinted before, the PC*jr* differs from other IBM personal computers in the way it uses memory to support the display screen. All IBM personal computers have what is called a memory-mapped display, which means that what appears on the screen is stored in the computer's memory. The central processing unit doesn't "write" information onto the screen, it changes what is stored in memory. Meanwhile, the computer's display circuitry continually reads whatever information is stored in memory, and converts it into the signals needed for the screen.

In the PC, there is a special kind of memory dedicated to the display screen's memory mapping. What makes it special? First, it is dual-ported, which means that the display circuitry and the computer's processor each has its own independent access port to the memory, so that they will not get in each other's way; both the display and the processor can be working with the dual-ported memory at the same time, without either one slowing the other down. This arrangement works very well for memory-mapped displays but is more expensive than ordinary memory. To help hold down the cost of the PC*jr*, its display memory is *not* dual ported. To keep the display and the processor from butting heads, the memory circuitry for the PC*jr* gives a certain proportion of memory access cycles to the display screen, letting the processor use the remaining cycles when it needs them. The net result is that it takes the PC*jr's* programs six clock cycles each time it uses the memory, whereas the PC and XT do the same work in four. This shared use of memory cycles slows down the PC*jr* only when it is using the memory. The PC*jr* runs more slowly than the PC, primarily because of the extra time needed to use its memory. This is a small price to pay, however, particularly since most programs don't use much of the computer's power anyway. In many circumstances, the slower running time of the PC*jr* will not be noticeable.

The new color tricks are so extensive and interesting that they could use a whole article to themselves.

Display memory in the more expensive IBM personal computers is unique for a second reason: its location in the computer's address space. The memory addresses, or address space, of IBM personal computers have a range of 1 million bytes. About two thirds of this space is set aside for our programs to use; the rest is held off for certain dedicated uses, one of which is providing a fixed location for the display memory. For those familiar with hexadecimal memory addressing, the monochrome adapter places its display memory at segment B000, and the color graphics adapter places its display memory at B800, both toward the high end. In keeping with its remarkable nature, the PC*jr* sets up its display memory in a way that is both completely different and yet also functionally the same as the PC.

How can we explain this apparent contradiction? To start with, the PC*jr* doesn't use any special memory at all—the ordinary, general-purpose RAM that is used for programs is also used for the information that appears on the display screen (which is partly why the PC*jr's* display memory can't be dual ported). Next, to provide memory for the display, a portion of the PC*jr's* memory (64 or 128K) is set aside for the display screen. Usually 16K is set aside, which is the same amount that the original color graphics adapter had. However, the amount of memory set aside can be adjusted. However much memory is given to the display, it is at the top of the computer's 64K or 128K, and the rest of the memory is used in the ordinary way. If a PC*jr* with 128K of total memory has the display set to use 16K—the normal amount—then 112K is left over. In this case, the computer will act like a PC with 112K memory.

In the PC, the display memory is located at one of two fixed locations, depending upon whether a monochrome or color graphics adapter is used. Many programs have been written that make use of this fact by placing their output directly into the computer's display memory. When IBM was designing the PC*jr*, it knew that the actual display memory would not be located at the traditional places, but elsewhere, (and at a moveable "elsewhere" at that), since the actual location of the PC*jr's* display memory

changes depending upon how much memory there is and how much is given to the display. This creates a real problem for all those programs that try to use the display memory directly—they will be looking at the wrong location when they're run on a PC*jr*.

IBM's ingenious solution to this problem is a piece of circuitry known as a video gate array or VGA. The VGA is a sort of programmable supervisor of the use of memory. It is told when the PC*jr's* actual display memory location is set. As programs run on the PC*jr*, the VGA constantly checks any references to the traditional location of the color graphics display memory, located at hex segment B800. If the VGA finds that a program, any program, is trying to use the PC's display memory locations, the VGA invisibly reroutes things to the PC*jr's* true display memory. By this sleight of hand, the VGA makes the PC*jr* act as if it were an ordinary PC with the color graphics adapter. Even though there is no memory on the PC*jr* board actually located at B800, part of the PC*jr's* regular memory is treated as though it had been moved to that location.

Adjusting the Amount of Memory

As we've mentioned, the amount of memory used by the PC*jr's* display screen can be adjusted. This statement calls for some explanation. Depending upon what display mode the computer is in, different amounts of memory are needed to store the information that appears on the screen. In the original color graphics modes, this could be as much as 16K, or as little as 2K for the 40-column text modes. Accordingly, the original color graphics adapter board was built with exactly 16K of display memory on it. When the computer was in a graphics mode that needed 16K, the screen used the entire display memory. In any of the text modes, which used less than 16K, part of the memory was used for the information currently shown on the screen, and the rest was set aside for off-screen images, called pages. BASIC gave us control over these display pages with the visual-page and active-page parameters of the SCREEN statement.

PC*jr* takes this fundamental idea and develops it further. First, its display memory, carved out of ordinary main memory, is adjustable from a low of 2K to a high of 32K (which is required for using either of the new medium- and high-resolution modes because of their extra colors). Naturally, because the PC*jr* is set up to mimic the PC, the default for the display memory size is 16K, the same as the PC's. But PC*jr* can change the size of the display memory: expand it for more pages or for the new modes, or diminish it to recover working memory space. It's particularly useful to reduce the size of the display memory, since few programs make use of multiple display pages and the PC*jr* has less memory than members of the PC community are accustomed to using. If, for example, we're running a 128K PC*jr* with DOS 2.10 with the display memory set at the default value, our programs have 88K to themselves—a good-sized amount but not huge by PC standards. If we cut the display memory down to a reasonable minimum of 4K, then our programs have 100K to work with, over 13 percent more.

The innovation means that we can connect a TV set to our PC*jr* with an ordinary audio jack connector.

The final novelty in PC*jr's* video display magic is the number of plugs on the back of the system unit. There are no fewer than three different built-in video outputs. As expected, PC*jr* gives us exactly what a color-board PC would give us—a composite video outlet and an RGB outlet. The RGB outlet is designed to connect to a computer-quality RGB monitor, providing it with separate signals for the three components of display color, red, green and blue (R, G, and B). The composite outlet is designed to connect to what is called a composite monitor, of which there are two kinds. A color composite monitor will produce all the colors that the IBM PC generates, just like a high-quality RGB monitor; the only difference being that the picture quality may not be as good as that of an RGB. A monochrome composite monitor will produce a picture similar to IBM's monochrome monitor (which can't be used with the color board, or with the new PC*jr*) and may produce a green, amber, or white display image. Many PC users have found what PC*jr* users will discover—that this kind of display screen is one of the cheapest available and one of the easiest to read when you are working with written text for programming or word processing.

RGB and composite outlets are fine for connecting our computers to special computer monitors, but these outlets can't be directly connected to an ordinary TV set. In the old days, we needed another piece of equipment, called an RF modulator, to convert the computer's composite signal into a TV-style signal. The PC*jr* has a third video outlet that provides a signal our TV sets can use directly, thanks to an RF modulator built right into the PC*jr*'s system unit. The innovation means that we can connect a TV set to our PC*jr* with an ordinary audio jack connector. This direct TB signal provides one extra benefit previously unavailable on the other models of the IBM personal computer. When we use the TV outlet, the sounds generated by the PC*jr* are included with the picture image, so that our TV sets can produce both sound and picture together.

This article is just a quick run-through of what is new and exciting about the PC*jr*'s video display. It only begins to sketch out the main points, and there is much more to learn about the amazing details inside the PC's younger sibling. ■

Cartridge Magic: PC*jr*'s New Memory

The PCjr's two cartridge slots fit neatly into the memory design of the IBM personal computer family.

Of everything that's new in the IBM PC*jr*, one of the most innovative is the software cartridge. In this article, we'll take a look at how the new cartridges of the PC*jr* work and how they fit into the overall design of IBM personal computers.

A computer's memory can be compared to a playing field where the championship teams of computer instructions we call programs perform athletic feats. Plugging a software cartridge into the computer is like putting a team onto the playing field—it's an instant way to add excitement to the sport of computing.

To appreciate the magic of software cartridges, we must understand some key things about how IBM Personal Computers use their memory, since software cartridges are a part of that memory. IBM Personal Computers, actually have two different kinds of memory. Random Access Memory (RAM) is used by the computer to hold the programs and data that it is currently working with. This kind of memory doesn't have any permanent use; it's just the playing field where different teams, or programs, can play. When the program teams are done, they leave the RAM playing field free for the next team. When we talk about our computers having 64K or 128K of memory, we're talking about how much RAM it has, measured roughly in thousands of bytes.

Understanding RAM and ROM

Anything can be stored in RAM for as long as it is needed. This information can be inspected (or read out of memory) and

changed, or written into memory. For this reason, RAM is sometimes called read/write memory. When a new program begins working, it overwrites whatever was left behind by the last program; when the computer is turned off, everything that was stored in RAM is then erased.

Computers also use ROM, which is permanently recorded with programs and data, so it is able to provide us with programs that are always ready to be used and which can't be accidentally erased or lost the way that programs on diskette can be. Giving a computer programs recorded on ROM is like building a locker room next to the playing field. ROM doesn't increase the size of RAM's general playing field, but it adds extra facilities that can enrich the use of the computer.

Each IBM Personal Computer, whether it's a PC, an XT or a PC*jr*, has two distinct sets of ROM programs built into them: one is the cassette BASIC, which we see in action when we turn on our computers without a diskette, and the other is a set of fundamental control programs, called the ROM-BIOS, which assist the operation of the computer. These ROM programs are permanently recorded on memory chips, which are permanently installed in the computer.

PCjr's two program cartridge slots.

The program cartridges that the new PC*jr* can use add an extra degree of flexibility to ROM programs. The PC*jr*'s program cartridges are permanently recorded, so users don't have to worry about erasing or damaging them. But because they aren't permanently installed, they can be switched around; one cartridge can be plugged in for one use and another cartridge for a different use.

A cartridge plugs pre-recorded memory into the PC*jr*'s memory space. To understand how that works, here's a short tutorial on how the memory is organized.

Use of memory in the IBM personal computers is based on the architectural design of the Intel 8088 microprocessor. Every computer has what is called an address space, which is the definition of all of the memory that it could have. The specific memory that the computer actually has is located within that address space. IBM Personal Computers, and all computers which use the Intel 8088 chip, have an address space of slightly over 1 million bytes—1,048,576 bytes, to be exact, or 1,024 kilobytes.

Block Number	Use
0	Ordinary RAM for our programs and data
1	Ordinary RAM for our programs and data
2	Ordinary RAM for our programs and data
3	Ordinary RAM for our programs and data
4	Ordinary RAM for our programs and data = 640 K total
5	Ordinary RAM for our programs and data
6	Ordinary RAM for our programs and data
7	Ordinary RAM for our programs and data
8	Ordinary RAM for our programs and data
9	Ordinary RAM for our programs and data
A	Reserved (mysterious)
B	Display memory, monochrome and color graphics
C	Fixed disk and mysterious reserved
D	for use by 2nd cartridge slot
E	for use by 1st cartridge slot
F	ROM - BASIC and BIOS

Figure 1: *The 16 memory blocks in an IBM PC, XT, and PC*jr*, and their functions.*

Memory Architecture

When IBM designed the PC family of computers, it had to make some basic decisions about how that memory capacity would be used. In broad terms, two thirds of the address space is dedicated to general use by RAM memory; the other third is set aside for special uses such as the PC*jr*'s new software cartridges. To understand them in more detail and to see exactly how the software cartridges fit into the scheme of things, let's take a look at the working boundaries of the memory.

The entire 1,024K memory space of our computers doesn't have any real boundaries or divisions and every part of it can be used uniformly. But just as yard lines are drawn on a football field to mark off locations, so the computer's memory is divided into 16 blocks of 64K each. This division into 64K blocks helps us talk about different parts of the address space, and it helps show some of the structure that IBM designed into their use of the computer's memory. We can identify

each of these 16 blocks of 64K each by the first hexadecimal digit of their addresses, and we can refer to them as 0 through 9 and A through F.

Figure 1 shows an outline of all 16 blocks of memory. The first ten blocks together, a total of 640K, are set aside for ordinary RAM. This is almost two-thirds of the total space. The A-block that follows the RAM area is set aside for some future use. The B-block is used for the memory that the monochrome and color-graphics adapters need. The two original PC display adapters combined use only 20K of the 64K in the B-block; the rest of the space in this block could be used for expanded display modes.

The C-block of memory is partly used by the BIOS control programs for IBM's 10-megabyte fixed disk used in the XT. This BIOS program is stored in ROM. Curiously, this small amount of BIOS programming is placed right in the middle of the 64K C-block, dividing it neatly into half.

The main ROM programs for all of the IBM Personal Computers are located in the F-block, at the high end of the computer's address space. Both the cassette ROM-BASIC and the ROM-BIOS are located in this block.

One Mystery Solved

Before the introduction of the PC*jr*, both the D- and E-blocks of 64K memory were reserved for unspecified purposes. Now we know that they were set aside for use by the PC*jr*'s software cartridges. Before the introduction of the PC*jr*, both the D and E-block of 64K of memory were reserved for unknown purposes. Now, though, we know that they were set aside for use by the PC Jr's software cartridges. Each cartridge could use any of the memory space in this 128K area. The electronic coding in the cartridge determines exactly where in memory the cartridge's data will appear. For example, the BASIC cartridge places itself at the paragraph address E800, or exactly in the middle of the E-block of memory. This leaves the remaining 96K (all of the D-block and the other half of the E-block) for any other use. By convention, single cartridges plug themselves into the E-block of memory. Second cartridges (such as a BASIC program cartridge, which works with the BASIC language cartridge), normally use the D-block. If a program wants to look for cartridges all it has to do is to search through this 128K area of memory, to see what is there.

The information stored in a cartridge simply appears or disappears from the computer's address space whenever the cartridge is plugged in or removed. A sensor switch causes the computer to reboot when we plug a cartridge in, just as if we had pressed Ctrl-Alt-Del. This allows the computer to respond to the cartridge. In the booting process for the PC*jr*, the memory locations set aside for the cartridges are checked and allowed to indicate that they want to take control when the computer begins.

Before the introduction of the PC*jr*, both the D- and E-blocks were reserved for unspecified purposes.

Cartridges can contain anything that anyone might want to put on them, from exotic programs to ready-to-use data. In practice, though, they are intended for only a handful of things. One is standalone programs, usually games, that can be run without any use of the computer's disk drive. These are the cartridge equivalents of the self-loading diskette programs used without the help of DOS or any other operating system. Normally these cartridge programs would be plugged into the first slot; if a program is too big to fit into 64K—which is unlikely—it could have a companion cartridge for the second slot.

The second kind of cartridge is designed to work as a supplement to DOS. When DOS is running and given a command—for example, DIR or FORMAT—DOS first looks for the command program inside itself, checking for internal commands such as DIR. If the command isn't found internally, DOS looks to the disk for the external commands, like FORMAT. However, between looking to the internal command table and the external disk, DOS also checks to see if the command is on a cartridge. If so, then DOS carries it out with the program on the cartridge. The most obvious example of this kind of DOS program cartridge is the BASIC cartridge. When we have DOS running with the BASIC cartridge plugged in, and we give DOS the command BASIC or BASICA, DOS will run the version of BASIC that is stored in the cartridge.

The Second Slot

The third kind of cartridge is plugged into the second slot, with the BASIC cartridge in the first slot. Stored on these cartridges are BASIC programs that need to have BASIC running at the same time. The BASIC cartridge is designed to check the second cartridge memory block and run any BASIC program it finds there. This makes it possible to have game cartridges with programs that are written in BASIC.

Each cartridge is supposed to have a standard signature, which is 2 bytes coded in hexadecimal 55 and AA. This signature appears at the beginning of each cartridge on the extra ROM-BIOS code for the IBM fixed disk—oddly enough, at the end, not the beginning, of each diskette's boot record. Although the cartridges plug into two specific addresses, at the beginning of memory blocks D and F, DOS and other parts of the computer's system programming actually check at every 2K memory boundary for this special signature.

Figure 2 gives a short BASIC program that will search all of the computer's address space for these signatures. If you have a regular PC, this program will find nothing except for the fixed disk BIOS at location C800, if you have it. But on a PC*jr*, this search program will detect any cartridges that are plugged in.

The hex 55 AA signature shows that there is a cartridge present but doesn't tell anything about its contents. We find out about the contents from some special coded information that follows the cartridge's signature. The next byte, the third byte of the cartridge's memory, gives the size of the cartridge's ROM, in units of 512 bytes. This scheme allows for a size up to 128K. A single 128K cartridge theoretically could fill the space allotted to both cartridge slots.

Self-Starters

Following the length byte is a single machine language instruction, a 3-byte-long jump, which is intended to let the cartridge perform any start-up work that it wants to do. In the design of the IBM personal computers, interrupt 18 activates the computer's built-in cassette ROM-BASIC. When the computer is started or reset without a disk ready, the start-up programs perform an interrupt 18 to turn control over to the ROM-BASIC. This was the original scheme, in any case. Now, to accommodate self-starting cartridges, the PC*jr*'s start-up routines allow a cartridge to change the controlling vector for interrupt 18 to point to some part of the cartridge's programs. This way, the computer is given a standard place to look for the beginning of a cartridge program, while the cartridge can then jump to wherever the actual working program is located. This is part of the scheme that makes self-starting cartridges work, and it's also part of what makes it possible for the BASIC cartridge to work both as a self-starting program and as a program that can work under PC-DOS. Three bytes have been set aside on the beginning of a cartridge to let the cartridge participate in the computer's start-up activity.

Following the 6 bytes at the beginning of each cartridge, there is a table that DOS uses to identify the command programs on a cartridge. There is one entry in the list for each command program that DOS can use; the end of the list is indicated by a 0 byte after the last entry. Each entry in the list has three parts: a single byte that indicates the length of the name of the command, the command's name, and a 3-byte-long jump instruction that is used to activate the program.

Even cartridges that don't have any DOS command programs on them will have an abbreviated version of this command table, just to let DOS know that there aren't any commands on those cartridges. An empty command table consists of just a single 0 byte, the byte that is used to mark the end of the table.

The BASIC cartridge reveals what DOS command lists are like. There are two entries in the table on this cartridge, for the two commands BASIC and BASICA. BASIC and BASICA, of course, are the names of the two versions of BASIC that the original PC-DOS had. In PC-DOS, these are actually two distinct versions of BASIC, but on the cartridge there is only one program that provides all the features of BASIC. To keep completely compatible with the PC, the BASIC cartridge provides both names, even though they both use the same program.

Acting Dumb

There is a third program on the BASIC cartridge, known as TERM, which allows the PC*jr* to act like a "dumb" ASCII terminal. This TERM program, however, isn't included in the cartridge's table of contents as a third entry after BASIC and BASICA. Instead, TERM is activated through BASIC as a command, even though it isn't really a part of the language.

A complete PC*jr* with this TERM program on the BASIC cartridge costs less than many ASCII terminals. We may see lots of PC*jr*s sold to mainframe computer users just to serve as ASCII terminals; to these people, the PC*jr*'s computing ability is just a benefit. We've heard that IBM didn't plan to offer this program; it was a last minute idea that Microsoft thought up when it found out that there was enough room left over in the BASIC cartridge. Ironically, this one feature may end up selling lots, and lots of PC*jr*'s. ■

```
100 ' Check for 55 AA signature
110 '
120 CLS : KEY OFF
130 PRINT "Searching for signatures" : PRINT
140 FOR I = 0 TO 511        ' check every 2 K boundary
150   SEGMENT = I * 128     ' calculate memory segment address
160   DEF SEG = SEGMENT     ' get ready to PEEK
170   IF (PEEK (0) <> &H55) OR (PEEK (1) <> &HAA) THEN GOTO 190
180   PRINT "Signature found at hex paragraph "; HEX$(SEGMENT)
190 NEXT I
200 PRINT : PRINT "Done searching"
```

Figure 2: *This BASIC program searches the computer's address space for a cartridge's signature.*

Sound Abilities: The PC*jr*

The PC*jr* has distinguished sound-making skills that will enhance the noises in game programs. Norton believes that these capabilities reflect IBM's intentions for its new machine.

The new IBM PC*jr* incorporates many new features, but at the same time it closely matches the abilities of the PC. The most dramatic improvement over the PC is its ability to create music and sounds. We'll take a look at PC*jr's* remarkable new sound features and investigate both the technical details and the practical aspects of using them.

A Game Machine

Let me raise a question before we get started. If one of the main design goals for the PC*jr* was to make it thoroughly compatible with the PC and the XT, why did IBM add new features that would set the PC*jr* apart? Why give the new machine abilities that the PC doesn't have? The answer is games. Think about this question and the answer. It is significant to the growing community of PC users because it tells us a great deal about IBM's plans for its personal computers.

I have no doubt that IBM is serious about program compatibility among all the models of the growing PC family. However, it is evident that IBM is introducing a degree of what we might call "functional product differentiation." The company is producing various models of personal computers, which are different from one another not just in price and overall capability, but also according to the type of use. The PC*jr* is clearly targeted at home users, which means that games will become one of its largest applications. The PC is a most capable computer. It has always been able to play games well, but it lacks some of the features that the most

popular game computers have. Among them are two video features—additional colors and "sprites," which are parts of the screen picture that can be moved around quickly through hardware rather than software control. In addition, it lacked rich sounds. But, with the PC's high computing power, there isn't as much need for sprite hardware. The new video features enable the PC*jr* to make better use of color, though, in fact, it has more basic colors than the PC. One area where the PC falls completely short is in its capacity for sound variation, which is insufficient for most games. The PC*jr*, however, has distinguished sound-making skills.

But what about compatibility? Do the PC*jr's* sound capabilities sabotage the need for program compatibility among all the IBM Personal Computer models? Of course the PC and XT can't utilize the sound and video features programmed into a PC*jr* game, unless they are equipped with the necessary add-ons. However, most of the PC*jr's* new features, particularly the ones for sound, are oriented toward games, but, for the most part, XT users are not heavy game players. Worrying that certain game programs can't be run on an XT is like fretting because we can't put the XT's 10-megabyte fixed disk on our PC*jrs*. Some people may want to, but doing so is contrary to the very nature of each computer.

Entertainment and educational software programs need extra sound capability because we interact with them in a way that is more complex than interacting with word processors and spreadsheets. Games generate action sounds from chest pounding gorillas to exploding bombs. These programs play music in chords, sounding more than one note at a time, and, with the PC*jr's* new sound hardware, all of this is possible.

Technicalities

Now let's get down to the technical details. The PC*jr* retains all the PC's sound-making abilities, and adds some entirely new ones. While the new features for the display screen are natural extensions of the PC's abilities in this area, the sound-making abilities are accomplished through an entirely new means.

The sounds we hear are actually waves of pressure in the air. The frequency at which they oscillate determines the pitch of the sound, and the strength of the waves determines the volume. A computer creates sounds by generating electrical pulses at the correct frequency and strength to enable a speaker to produce the sound desired. In the PC's sound-producing scheme, the computer's clock pulse and a programmable counter were combined to generate the required frequency. Here is how it works.

> The PC*jr* retains all the PC's sound-making abilities and adds some entirely new ones.

Both old and new sound in the PC*jr* begin with a frequency of 1,193,180 cycles per second, or 1.19 Mhz (million cycles per second). For the PC's sounds, this frequency is fed into a programmable counter, known to electronics designers as an 8253-5 timer chip. Our programs, written in BASIC or any other programming language, "program" the 8253 timer chip by passing it a counting number—let's say the number 2280. The 8253 timer counts the 1.19-Mhz signals being fed into it, until they reach the count value. Then the 8253 sends out a single pulse and begins counting again. In effect, the 8253 chip divides the 1.19-Mhz signal by the count that it has been given. So, if we give the 8253 chip a count of 2280, it divides the 1.19-Mhz signal into 523 cycles per second, which is the pitch of the musical note middle C. After it is amplified to the right voltage level, the 8253 chip passes the signal onto the computer's built-in speaker.

When this signal is passed to the speaker, it produces a sound with the same frequency as 1.19 million divided by the number given to programmable counter. Nearly any sound frequency can be produced this way, within the limits of arithmetic. A divisor of 60 would give a frequency of 19,886 cycles per second, while a divisor of 59 would give a frequency of 20,223. But, we couldn't achieve any frequency in between (say, by dividing by 59.5). However, in practice, this isn't a limitation; we can produce nearly any frequency that human ears can hear.

While this scheme can produce a pure musical tone at nearly any pitch, it has quite a few limitations. For one thing, only one sound can be produced at a time. Furthermore, it doesn't provide any control over volume, and it produces only pure musical pitches, not the kind of noises needed for rocket liftoffs and bomb explosions.

PC*jr's* Four Voices

The PC*jr* overcomes these limitations. While it follows the same basic approach, it adds the power of a circuit chip: the Texas Instruments SN76489A sound generator (henceforth TI). The TI sound generator provides the PC*jr* with many additional features. It can produce four different sounds simultaneously with what we call "four voices." Three of the four voices produce pure tones, like the PC, while the fourth produces noise sounds. Each of the four voices has its own independent volume control, which is called "attenuation." And finally, the noise voice can produce two quite different types of sounds.

Here's how PC*jr's* four voices work. Each of the three ordinary voices of the TI sound generator accepts a number, which controls the frequency of the sound that is generated. The sound generator starts out with the computer's 1.19-Mhz frequency, divides it by 32, and then divides it again with the controlling count that a program has assigned to any of the three voices, which results in the sound frequency that

the voice will produce. For example, if we gave a controlling count of 100, the frequency produced would be 1,193,180 divided by 32 and divided again by our 100 count, giving us a pitch of 373 cycles per second, roughly a G flat note.

Each voice takes its own count and produces its own sound frequency. The controlling counts are binary numbers 10 bits long, so they can range from 1 up to 1,023. This provides a range of sound frequencies from a high of 37,287 cycles (with a divisor count of 1) to a low of 36 cycles (with a divisor of 1,023), which is roughly the range that the human ear can hear.

While the TI sound generator's three ordinary voices produce pure tones following the arithmetic we've outlined, the fourth noise voice works entirely differently. It produces a scrambled pattern of electrical pulses, using what is called a shift register with an XOR feedback network to produce an irregular series of electrical pulses, which we hear as a noise sound. The noise voice can produce an even pattern, which we'll hear as a pure "white" noise, or a periodically varying pattern, which causes a more ragged sound. Each noise pattern has particular uses in game programs. The noise voice needs a base frequency from which to build its scrambled sound pattern, but, unlike the other three voices, this voice doesn't have its own full range of frequencies. Instead, we are given four choices—either use one of three fixed frequencies or borrow the frequency being generated by the third of the three pure tone voices. The three fixed frequencies are 2,330 cycles (a high-pitched even hissing sound), 1,165 cycles (a medium hiss), and 582 cycles (a lower, rougher hiss). If we don't need all three of the other voices for pure tones, then we can base our noise on any frequency we choose, which greatly extends the range of noises that we can produce.

Volume Control

Each of the four voices in the TI sound generator has its own independent volume control, known as attenuation. While the noise voice doesn't have its own full frequency range, it does have its own separate volume range, just as the other three voices do. The volume is controlled by four attenuation bits, each of which reduces the level of the sound by a fixed proportion, measured in decibels (dB). Each successive bit deadens the sound by about half as much as the bit before, so that the bits can be combined to produce a fairly even range of sound levels. The bits are referred to in the TI generator as A0 through A3, and here is the measurement in decibels of how much each of them reduces the sound level:

```
A0   2 dB (least attenuation)
A1   4 dB
A2   7 dB
A3 15.5 dB (most attenuation)
```

By combining the bits, we can achieve different levels of attenuation; for example, if A1 and A2 were set, the sound would be attenuated 4 dB plus 7 dB, for a total of 11 dB. If all four bits were set, the sound would shut off completely. With four bits, there are 16 combinations, giving 15 distinct sound levels, and no sound at all.

> The noise voice can produce a periodically varying pattern, which causes a more ragged sound.

So far we've discussed the two ways that the PC*jr* can create sounds—from the original clock-counter combination and from the new TI sound chip. The PC*jr* is also capable of using these sounds in two ways. The PC and XT can use only the frequencies they create to produce actual sounds on the speaker built into the computer's system unit. The PC*jr* has this internal speaker, too, but, in addition, it can pass its sound signals to other electronic equipment. Among the many plug sockets on the back of the PC*jr's* system unit are two that can be used for sound.

One is an ordinary audio jack, which can be connected to a hi-fi set. Plugging it into a hi-fi results in a high-quality reproduction of the sound signals created by PC*jr*. As you probably know, the speaker built into PC and XT isn't designed for high fidelity—it was designed to produce sounds crudely and inexpensively. But now, with the PC*jr*, we can pass the computer's sound signals on to a decent sound system to achieve top-quality sound.

The other destination for the PC*jr's* sound signal is the video TV outlet on the back of the PC*jr's* system unit. This outlet is designed to connect directly to a TV set, and it provides the computers display image, already converted into the format needed by a TV set. This TV outlet is something new to IBM personal computers; the PC and XT models, provided they are equipped with a color-graphics adapter board, can be connected to a TV. But the connection is not direct; the signal must be converted by another piece of equipment called an RF modulator. One of the PC*jr's* features is a built-in RF modulator, so that we can connect it to a TV without fuss. Since the PC*jr* was set up to create a TV-type signal for its picture, it's only natural to include the computer's sound signals, too. After all, TV sets give us sound as well as pictures.

Here's a quick summary of PC*jr's* sound features. It can create sounds in two ways—either the same way the PC does or through its new TI sound generator. The PC*jr* can also use these sound signals two ways—either passing them to its internal speaker, like the PC, or sending them to external parts. The external signal is available either as part of the TV signal or independently in a signal connected to a hi-fi set. With the TI sound generator, we can create three independent pure tones and a noise sound. For each of the three pure voices and the noise voice, we can control the volume.

All these new sound capabilities require programming support, and, once again, BASIC comes through for us.

BASIC Support

Some programming language critics, myself included, like to throw bricks at BASIC, but when it comes to providing support for the features built into IBM personal computers, it compares to none. Let's take a short tour through the new BASIC commands that support these sound features.

The question of where sounds come from and where they go to is a good place to begin. The BEEP and SOUND statements control these functions. Figuring out what does what can be confusing, and the BASIC manual isn't completely clear, either. Here's how it works. First, there are four new statements:

```
SOUND ON    and    SOUND OFF
BEEP  ON    and    BEEP  OFF
```

If sound is on, then the new TI sound generator is active; when sound is off, then the old sound-generating circuitry does the job. This very simply takes care of where sounds come from—SOUND ON, new method; SOUND OFF, old method. But the explanation of where sound goes to is more complicated.

The sound signals can go to the internal speaker, to the external plugs, or to both—three combinations. (It cannot be turned off so that it doesn't go to either place.) Various combinations of BEEP and SOUND statements determine which one of these three possibilities is taking place. Figure 1 shows how the four combinations of BEEP and SOUND statements and ON and OFF statements control what's happening.

If you study Figure 1, you'll understand how it works. When the TI sound generator is working (SOUND ON), then the computer's internal speaker is turned off—because the internal speaker and its circuitry can't handle the TI generator's sound signals. So the SOUND ON statement, which activates the sound generator, always sets the internal speaker off and the external connection on. Then, BEEP ON also turns on the external connection, if it isn't already on, and BEEP OFF turns it off, unless SOUND ON is also turning it on—a little too complicated, but that's the way that it works.

Unless otherwise informed, BASIC will use the default settings, BEEP ON and SOUND OFF. The sounds come from the original sound circuitry the PC uses, while the sounds go out everywhere.

> With the PC*jr*, we can pass the computer's sound signals on to a decent system to achieve top-quality sound.

To make use of the three pure-toned voices and the volume controls, both the regular SOUND statement and the PLAY statement have been extended. For the SOUND statement, the two original parameters of frequency and duration continue to be used, and two new parameters are added: volume and voice. The volume parameter controls the loudness. Volume is set with a number from 0 to 15; 0 is silent; 15 is full volume; the numbers in between set various attenuation levels. If you are familiar with binary arithmetic, you'll recognize that the inverse of the BASIC volume value gives us the bit settings for the TI sound generator's four attenuation bits, A0 to A3.

The new voice parameter of the SOUND statement controls which of the pure tones voice will be used. The voices are numbered 0, 1, and 2, respectively. The last voice, number 2, is the one whose frequency can be borrowed by the noise voice. If we set a SOUND frequency for voice 2 with a volume of 0, then this voice will be silent. But it still will generate a frequency ready for the noise voice to use.

The PLAY statement has also been extended to use the new features, but in a different way. While it takes a separate SOUND statement to activate each of the three voices, a single PLAY statement can control any or all of the voices. The new version of PLAY will take three separate music definition strings, one for each voice. If a string is left off, then that voice isn't affected by the PLAY statement. For example, a statement like this:

```
PLAY , NOTES.1.$, NOTES.2.$
```

would skip voice zero and use the other two voices.

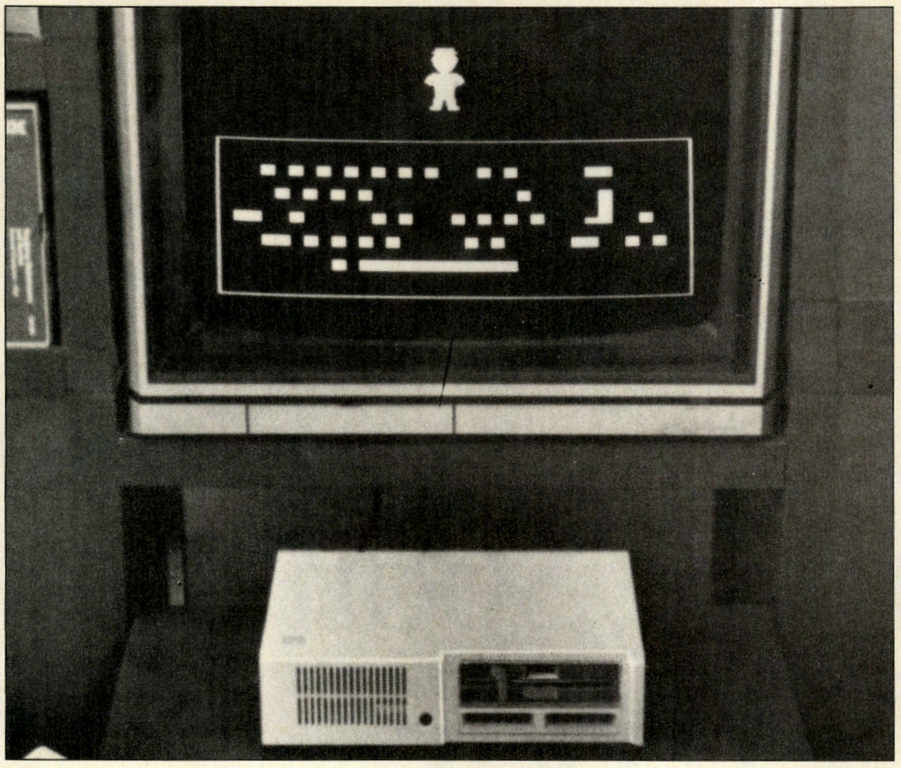
To let the PLAY statement control the volume of the music that it is playing, an additional command letter has been added to the music definition language, V, which sets the volume level between 0 and 15 in the same way that the SOUND statement sets the volume. The default volume level is 8, so we can increase or decrease the volume from the default setting.

The noise voice in the TI sound generator has quite special abilities and so BASIC handles it with a special NOISE statement. The NOISE statement looks like this:

```
NOISE source, volume,
duration
```

The duration parameter is the usual count of clock ticks, which is familiar to anyone who has used the SOUND statement with the PC. The volume is the new, but now also familiar value from 0 through 15, with the higher number creating the loudest noise. The source parameter controls what kind of noise sound is made. The source is a number from 0 through 7, and it works in a way that is similar to BASIC's color numbers, actually controlling more than one thing at the same time.

The NOISE source number determines the base frequency of the noise. And it also controls whether the noise will be "white" or periodic. Figure 2 shows how this works.



BEEP	SOUND	TI Active?	Speaker	Connections	Comment
OFF	OFF	No	Yes	No	
OFF	ON	Yes	No	Yes	
ON	OFF	No	Yes	Yes	default setting
ON	ON	Yes	No	Yes	

Figure 1: *BEEP and SOUND control in PCjr's BASIC.*

PERIODIC Noise	WHITE Noise	Frequency used for the noise
0	4	2330 cycles (high)
1	5	1165 cycles (medium)
2	6	582 cycles (low)
3	7	Borrowed from voice number 2

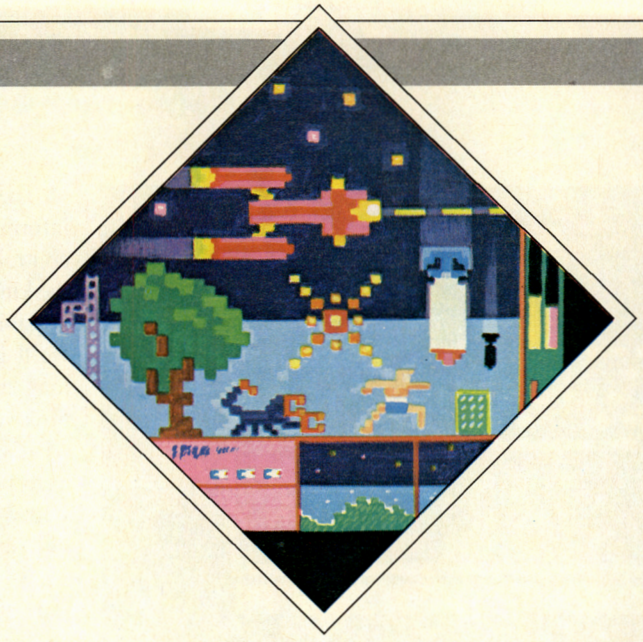
Figure 2: *PCjr's BASIC's NOISE statement controls base frequency.*

There is, of course, quite a bit more information to explore about how BASIC controls the PC*jr's* new sound abilities and the kinds of sounds they can produce. The possibilities are so great that I'm sure that we'll be trying new combinations for a long time and making some remarkable discoveries. Just to give you a taste of what's in store, let's consider these few examples.

- If we want our computer to play music like an ensemble of two or three instruments, we can use the PLAY statement to start playing three quite distinct music strings, each of which has the part for one of the instruments.
- If we want to make our personal computer play music to sound like a single instrument that plays chords, we can set up three music strings that have an identical time pattern, with the same tempo and duration of notes and rests, but with the differently pitched notes that we need to create our chords.
- We can have our computer play a "round" such as "Row, Row, Row Your Boat," by setting up one music string, shifting and copying it into two other strings, using the MID$ statement, and then PLAYing the same music in three rotated parts.
- And finally, one of the most bizarre things we can do with the sound features is to have the PLAY statement silently create various frequencies in voice number 2, by using MB (music in the background) and V0 (zero volume, silent) in the music string. Then, we can use the NOISE statement to borrow the frequencies from voice 2, which will allow it to create varying noises. Here is an example of how we could do this.

```
10 SOUND ON
20 PLAY "", "", "MB;V0;L1;
ABCDEFGFEDCBABCDEFGFEDCBA"
30 NOISE 3,8,400
```

If these new sound possibilities excite you, get your hands on a PC*jr*, plug in the BASIC cartridge, connect it to a hi-fi or TV set, and have fun. ■



Coming Soon: Games For The PC*jr*

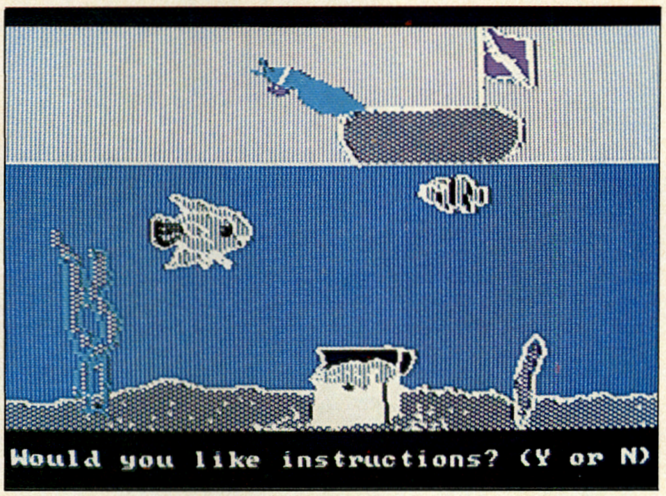
Software developers of computer games are converting their existing products or developing new ones for the IBM PCjr.

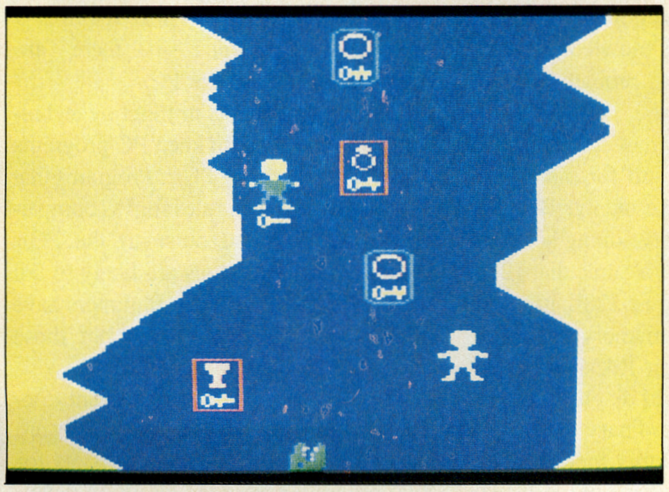
When IBM debuted its two new PC*jr* models, it quickly became obvious that the computer games industry had been eavesdropping. Most of the major software developers of games acknowledged that they are hard at work converting their games to the new format or creating new ones with exciting graphics. Other software houses are waiting to get their hands on a development machine.

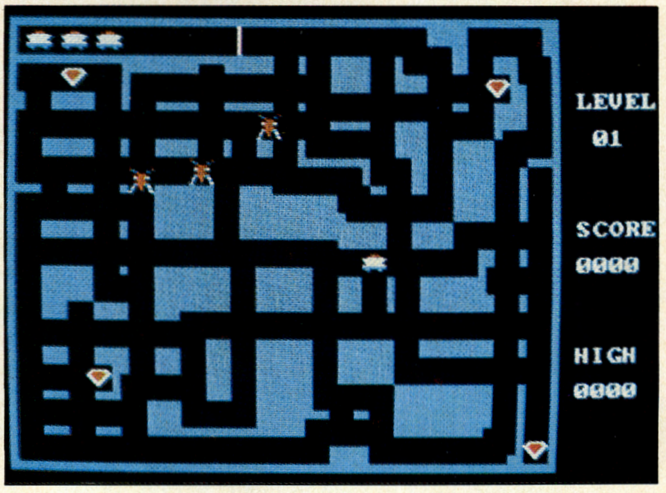
Indeed, four well-known California game design firms—Sierra On-Line, Gebelli Software, Microsoft, and The Learning Company—had been contacted in advance by IBM to create an initial line of game products for the PC*jr*. All were represented by finished products at the November 1 PC*jr* announcement.

Sierra On-Line developed *Adventure in Serenia,* a disk package for the PC*jr* that was not on display, and two ROM cartridges, *Crossfire* and *Mine Shaft,* both of which are available for other computers. (Sierra is one of the largest publishers of home computer software.) *Crossfire* is a captivating arcade-style shoot-'em-up, in which the player moves a character about a grid of alleyways in four directions, shooting at anything that moves. This game is one of the popular, best-known titles from Sierra. *Mine Shaft* is a version of *Creepy Corridors*—a game in which your computer draws mazes with diamonds in the four corners. Once you make your way to all four, a doorway to the next maze appears.

We're going to back the PC*jr* with software as we have backed the PC," said Ken Williams, president of Sierra On-Line. "We think it's an important machine that will do much for the industry







Screens depicting four different games now available for the PCjr.

in general, and we'll be converting most of our games line to the PC*jr* format."

Roberta Williams, wife of Ken and designer of many high-resolution text adventures for Sierra On-Line (among them *The Wizard* and *The Princess, Time Zone,* and *The Dark Crystal*), has just finished working on an original game designed specifically for the PC*jr* called *King's Quest*. It's an adventure game along the same line as her others, but it features something brand new to computer games—animation.

"I'd always wanted to do an animated adventure game," says Roberta, "but the game I foresaw couldn't be written for an existing home computer. I was happy to discover that the PC*jr* has animation capabilities."

It's almost impossible to use animation of high-resolution graphics with the artificial colors used with other computers, according to Roberta. But with solid colors, you can create beautiful animation. The PC*jr* has 16 solid colors available to the game designer. And the 128K model has enough memory to handle an animated adventure such as *King's Quest*.

Gebelli Software, founded by Nasir Gebelli, has been a driving force in the Apple game software arena. Two of his games, *Mouser* and *ScubaVenture,* have already been written in cartridge form and were on display at the IBM announcement. *Mouser* is a connecting series of nine rooms through which mice and a cat run rampant. The player's task is to position the movable maze walls to trap all the mice. Color graphics and sound effects are nicely used, and the game will probably appeal most to younger players. The same is true of *ScubaVenture,* a game in which

Photographs: Dennis Bettencourt

Developing PC*jr* Software

The PC*jr* opens doors for software developers and users but doesn't create any real problems.

What does the introduction of the new IBM PC*jr* mean to all the software developers who have been busily writing programs for its predecessors, the PC and XT?

The news for developers and users of PC software is good. The PC*jr* opens some new doors but it doesn't create any real problems for software developers or users. Existing PC software doesn't need to be redesigned or rewritten.

Because the PC*jr* is nearly 100 percent software compatible with the original PC and XT models, we don't need a new set of programs to make the PC*jr* sing and dance. We don't even need any special revisions to adapt existing programs to the PC*jr*.

No Cheating

Software developers can rest assured that their programs will work as well on the PC*jr* as they do on the PC and XT. However, they must understand the PC*jr*'s basic limitations and promise not to cheat. Of the basic limitations that software developers need to keep in mind, the most obvious are the single drive and the 128K memory.

There are two important issues concerning the disk drive. First, the PC*jr* computers will have only one. Even when industry's magicians give us ways to add on more drives, most of us will have only the standard one. The obvious lesson here is that programs for the PC*jr* must require only one drive. Relatively few PCs have only one drive (unless, like the XT, they also have a hard disk), and some program developers have become sloppy about supporting one-drive systems. The PC*jr* provides the motivation to create programs that will run with one drive.

Second, the PC*jr* disk drive is controlled and operated quite differently, with BIOS programs doing much of the diskette control work that the PC and XT's disk drive adapter does with hardware. The exact performance of the disk drive and the load that it puts on the computer's computing power are very different. Many copy-protection schemes rely on close and exacting knowledge of how the diskettes work, so problems in this area are likely to arise.

The most popular model of the PC*jr* has 128K memory, and for many programs that sounds like plenty. But programmers who are developing software for the PC*jr* shouldn't assume that 128K in a PC*jr* is the same as 128K in a PC. The PC*jr*'s display memory needs are taken out of the main RAM. (The PC and XT have dedicated display memory that isn't a part of ordinary RAM.) In most circumstances, this means that a 128K PC*jr* has 16K taken out for display use. (The amount of memory used by the display can be adjusted up or down, but 16K is normal.) When figuring how much memory is available for a program to use, we also have to take DOS into account. The PC-DOS 1.1 version use up about 12K of memory, and DOS 2.0 takes up twice that much. Fortunately, the PC*jr*'s DOS 2.10 doesn't need any more than DOS 2.0.

We find that a 128K PC*jr* has 88K of user memory for our programs and their data. This sets a new standard, or design point, for programs. A program that barely fits into 128K on a PC will probably not work on a PC*jr*, but any 128K program that has spare room should run without difficulty on the PC*jr*.

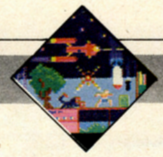
And Bear In Mind

There are also some minor things that software developers need to be aware of. The PC*jr* acts like a PC with a color-graphics display adapter, but it doesn't even try to imitate the PC's monochrome adapter. So a program that requires a monochrome adapter—if there are any—is useless on the PC*jr*.

There are also lots of niggling technical details that might keep PC programs from working on the PC*jr*. Programs that break the unwritten rules for PC compatibility by trying to outsmart the PC's BIOS control programs are likely to fail on the PC*jr*. Few PC programmers have written programs this way, but those who have may now regret it.

The PC*jr* opens one significant door for software developers: that of software cartridges. Since the PC*jr* has slots for two program cartridges, program developers can now consider either developing new cartridge programs or adapting existing diskette-based programs for the cartridge. This creates exciting new possibilities for bringing software to the PC*jr*—and also to the PC, when IBM or another manufacturer gives us a cartridge adapter for PCs and XTs. Cartridges are the main thing that sets the PC*jr* apart from the rest of the IBM Personal Computer family. The more integrated and compatible the family is, the better off we all are.

—PETER NORTON



the player moves against a scrolling screen and attempts to pick up treasures while avoiding nasty sea creatures.

Another program available for the PC*jr* is Microsoft's *Adventure,* which comes on a disk, not cartridge. However, it was not on display. More PC*jr* software is expected from this company. And though *Bumble Plot* and *Juggles Butterfly* (both on disk) from The Learning Company, were also completed they were not displayed at the product demonstration, either. The Learning Company, publishers of the highly acclaimed Apple product *Rocky's Boots,* has concentrated on software for the young user. Most of its programs are aimed toward an educational goal. Most strive for computer literacy, achieved through a fun activity. For example, *Bumble Plot* is a series of six games that use an animated creature, *Bumble,* to teach children ranging in age from 6 to 10, how to draw computer pictures. It also illustrates the concepts of greater than and less than. *Juggles Butterfly* is for the very young, ages 3 to 6. It uses colorful clowns to teach children spatial relationships and to develop simple math skills.

Other software developers are looking toward the PC*jr,* too. When I spoke to Mike Cullum, director of software development for Avalon Hill Microcomputer Games, he said he was "sitting here now trying to decide how much of our software is compatible with the new machine." According to Cullum, the company (which has published 2 dozen war games for the computer) plans to release a number of games for the PC*jr*. Among them are *Diplomacy,* the classic world power struggle, *TAC,* a game of tactical armored combat during World War II, *Panzers East,* the invasion of Russia, *Gulf Strike,* a simulation of a modern Persian Gulf war, and *Exercise Rhine,* a naval action game of close tactical warfare based in the North Atlantic during World War II. Most of these games are already available from Avalon Hill for the PC, Apple, and Atari computers. "These are the games that will definitely be available for the PC*jr* during the first half of 1984, and we're working on more," said Cullum.

"The company does not view the conversion as a big deal. Special graphics options are available in the new DOS 2.10 that are not available in DOS 1.1, which is what Avalon Hill's existing PC software uses. "So it will only enhance the games," said Cullum. "And the only reason we haven't gone to DOS 2.0 yet is because of all the users with DOS 1.1. We try to make our games as compatible as possible with the largest group of users," he continued.

Creature Creator is a program that lets you make a strange creature and then teach it to dance.

Broderbund software, another big name for Apple, Atari, and Commodore computers, currently has no public plans for the PC*jr*. However, the company's director of marketing, Cathy Carlston, said, "The company will develop some PC*jr* products in 1984." In all likelihood, we can expect conversions of Broderbund's top hits such as *Lode Runner, Choplifter, Serpentine,* and *David's Midnight Magic*. Carlston hinted that we might see an original title or two for the PC*jr* specifically. Broderbund's PC*jr* software will probably first be published on disk, which is standard operating procedure for this company.

Spinnaker Software

Spinnaker Software, the largest publisher of computer learning games in the world, also has plans for the PC*jr*. Initially it will publish *Facemaker,* a drawing composition program, *Fraction Fever,* a fast-paced action game in which the player must match correct fractions and zap incorrect ones, and *Kindercomp,* a series of games based on drawing, matching shapes and letters, writing words and names, and filling in missing numbers—all in time for the arrival of the PC*jr* in the stores. The plan at Spinnaker is to support the PC*jr* with the company's entire line of children's learning software, including the highly acclaimed detective games *Snooper Troops I* and *Snooper Troops II*. Currently, Spinnaker's products are available for Apple, Atari, IBM PC, and Commodore 64 home computers.

DesignWare creates learning programs for ages 4 through adult for the Atari, Apple, and IBM PC and plans to convert its entire line of seven games for the PC*jr*. *Math Maze, Spellakazam, Spellagraph, Trap-A-Zoid,* and *Spellicopter* are all journeys into word and number play. *Crypto Cube* is a three-dimensional word game with a built-in puzzle generator that allows you to create your own games. And *Creature Creator* is just what it sounds like—a program that lets you make a strange creature and then teach it to dance.

Epyx, a large publisher of computer games for Atari, Apple, IBM, Commodore, and Radio Shack computers, also plans to convert most of its successful products to the PC*jr* format in 1984, but the company hasn't made any announcements yet. However, since much of Epyx's software includes fantasy role playing and adventure games, you can expect similar programs for the PC*jr*.

Human Engineered Software and CBS Software, both with extensive lists of software titles, have carefully sectioned their product lines into about 30 percent entertainment, 30 percent home business, 30 percent productivity software, and some utilities, which means that although both companies plan to support the PC*jr* with the same enthusiasm they have shown for the PC, neither will offer many games.

The PC*jr* is a relatively inexpensive machine with plenty of memory and graphics capabilities. Only time will tell more about this new machine. What is certain, however, even at this early stage, is that the PC*jr*, if nothing else, might be the best game machine ever designed. ■