THE NEW *jr*:

A GUIDE TO IBM'S PC*jr*

WINN L. ROSCH

BANTAM COMPUTER BOOKS

12.95

THE NEW *jr:*

A Guide to IBM's PC*jr*

THE NEW *jr:*

A Guide to IBM's PC*jr*

WINN L. ROSCH

A HARD/SOFT PRESS BOOK

BANTAM BOOKS
TORONTO • NEW YORK • LONDON • SYDNEY • AUCKLAND

to mother

THE NEW *jr*: A GUIDE TO IBM'S PC*jr*
A Bantam Book / January 1985

ISBN 0-553-34161-8

Published simultaneously in the United States and Canada

Bantam Books are published by Bantam Books, Inc. Its trademark, consisting of the words "Bantam Books" and the portrayal of a rooster, is Registered in U.S. Patent and Trademark Office and in other countries. Marca Registrada. Bantam Books, Inc., 666 Fifth Avenue, New York, New York 10103.

PRINTED IN THE UNITED STATES OF AMERICA

HL 0 9 8 7 6 5 4 3 2 1

Contents

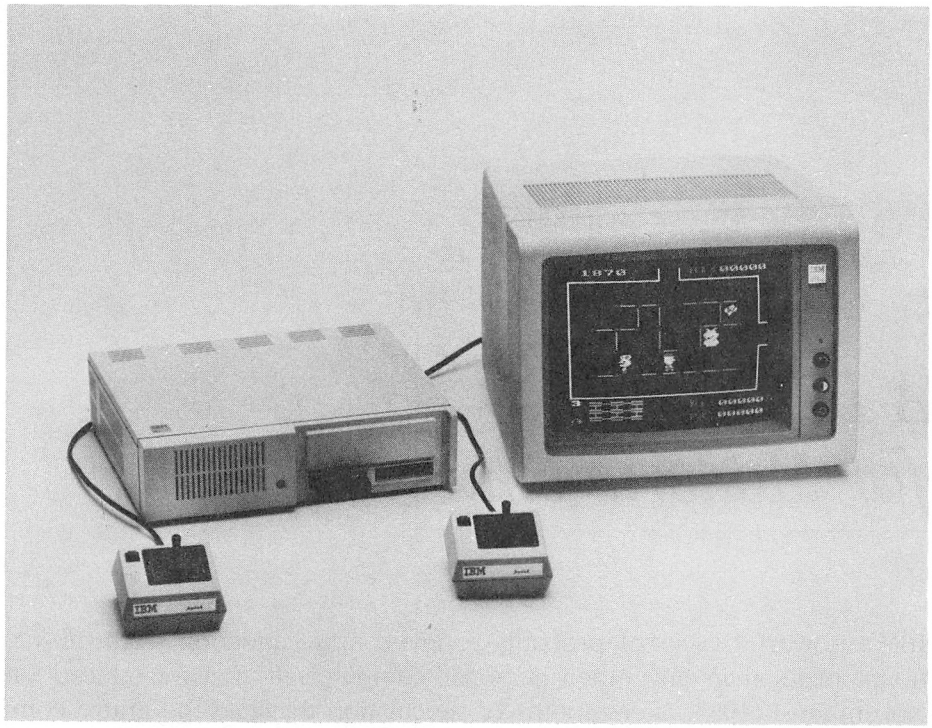
CHAPTER ONE

A Better Tool for Modern Times

IBM's powerful PC*jr* has probably received more attention from the national press than any other personal computer. It was introduced on November 1, 1983, after months of speculation that had the entire computer industry—and much of the public—breathlessly awaiting word on what kind of machine IBM was going to unveil.

The reaction was nearly universal, and entirely unexpected, especially to IBM, which had entered the personal computer market late but had quickly become the industry leader with its popular PC. Virtually everyone who tried the new computer complained about its unusual keyboard. IBM had carved itself a deserved reputation as the maker of the world's finest typewriter keyboards, and the PC*jr*'s was clearly not up to this standard. Other critics lashed out at the PC*jr*'s memory limitations and lack of a second disk drive.

What a difference a day makes. From the moment of its release, the PC*jr* had been regarded as IBM's biggest mistake ever. But with IBM's announcement, on July 31, 1984, of a slate of important enhancements, this humble little machine became the unquestioned best buy in the computing world, putting all the power and potential of the best-selling IBM PC—and much more—into a small desktop package. In fact, the new improved PC*jr* handles many programs far better than its more expensive cousin. For instance, the preeminent business spreadsheet, *1-2-3* from Lotus, runs *faster* in the PC*jr*'s cartridge than on the IBM PC. And, when combined with the speed and power of the chip inside, the PC*jr*'s



The standard PC*jr* is designed to run many games and other programs without a costly disk drive.

graphics and sound capabilities are nothing short of extraordinary, putting virtually all its competitors to shame.

The new PC*jr* keyboard is actually more friendly and familiar than that of the PC, built with all the improvements that most typists want on computer keyboards—a big Enter key, Shift keys in the right places, and a light, full-travel touch that positively lets you know when a key has been struck. In an unprecedented about-face, IBM admitted its mistake in the original keyboard design and now offers to *give* the owners of the old "Chiclet" keyboard a free new one. No longer must slippery little plastic keys stand between you and affordable-but-powerful personal computing.

Moreover, memory is no longer an issue. The PC*jr* is no longer crippled by having "only" 128 kilobytes of memory available (which was quite a bit more than was available in most other popular home computers). You can add as many extra bytes of memory to the PC*jr* as you can to the PC. And the PC*jr*'s cartridge slots allow more of its memory to be available for your programs than you would have in an IBM PC with the same amount of memory.

The new PC*jr* has better graphics potential than the PC. You get more colors at less cost—you don't have to buy a $250 adapter to put color on the PC*jr*'s monitor (as you must with the PC). You can't even run IBM's PC*jr ColorPaint* on the company's earlier personal computers. And there's no easier way of preparing colorful computer graphics on any personal computer at even twice the price than with *ColorPaint.*

Older computers don't even *sound* as good as the new PC*jr.* Not only does the PC*jr* have a built-in three-voice sound synthesizer, but you can add a voice synthesizer, as well. Or you can use the PC*jr* with voice attachment to record *your own voice* on computer diskette and add your voice to your programs.

Internally and philosophically, however, the PC*jr* is unchanged. It has the same microprocessor as before—the same powerful single-chip microbrain used in the bigger IBM PC and PC-XT—and it still has enough connectors on the back so that you can add just about anything to the basic system. It is still a versatile, open computer, designed not for a single purpose, but to be capable of doing *anything* that you want a computer to do. Although the PC*jr* makes an excellent desktop work station for business, particularly when connected to IBM's Cluster system, it is affordable enough for the home, easy enough to use for young children in school, and friendly enough to go by the name Junior rather than any bossy or foreboding initials.

An Identity Crisis

The keyboard wasn't the PC*jr*'s only problem when it was first introduced. IBM wasn't sure whether the new machine should be a computer for the home, school, or business. But today you'll discover that the machine is remarkably well adapted to applications in all three major areas. The reason for the amazing flexibility is based on IBM's philosophy of the computer.

IBM believes that all computers should hold the potential of doing almost anything, almost anywhere. The PC*jr* is uniquely adaptable, a mechanical Renaissance man ready and able to excel in nearly any endeavor. All the little machine needs to really show its stuff is the right software and a proper application—a job suited to its manifold talents.

When IBM proudly announced the release of the PC*jr*'s new keyboard, memory, and associated hardware and software, it focused attention on business applications to show off the machine's improved capabilities

because that's where the computer was the most maligned. According to IBM, the new PC*jr* is the perfect machine for the small businessperson or young executive who doesn't want to spend his or her future eating technological dust, adrift behind the wheels of progress.

But IBM hasn't forgotten the other major computer applications. The company's new "Writing to Read" program focuses on using the low-cost machine, together with speech synthesis and advanced graphics, as a classroom tool that will help your daughters and sons learn faster and better.

With Andrew Tobias's *Managing Your Money* program, newly published by IBM, Junior becomes the one right machine to take over your home finances. More than just another program to balance your checkbook—which you can probably do faster with a pocket calculator—*Managing Your Money* helps you plan investment and tax strategies. Rather than just keeping track of your cash, as other programs do, it and the Junior help you manage your assets and make them *grow.* In your off hours at home, you can still play your favorite games on Junior (games like *King's Quest*, which give Junior sharper and more colorful displays than are available on other computers) or even put the finishing touches on that novel you've been prying out of your soul for the last fifteen years, with one of the dozens of word processors the PC*jr* will let you use.

The Trouble with Junior

It's tempting to believe IBM's version of unbridled optimism. Junior has both the power to handle some of the toughest and most demanding business work, and the ability to run most of the programs in one of the largest software libraries in the world—nearly the entire catalog of IBM personal computer programs. Yet Junior is affordable enough to happily nestle atop a living-room television set and play games faster and with a sharper picture than is possible with lesser machines. This little computer is so easy to use that even preschool children whose alphabets stretch only as far as *C* can use the PC*jr*'s new keyboard, or Freeboard, and learn to read by writing, yet Junior is powerful enough to instruct high-school and college students with exactly the same programs that they might use later in life on more powerful business computers such as the IBM PC, PC-XT, or PC AT. With Junior, they will suffer no confusion of changing operating systems, controls, or commands when leaving their classroom Juniors to put their hands on the IBM computers in offices and executive

IBM's "Writing to Read" series of programs combines the PC*jr*'s superior graphics and sound power to make the PC*jr* the premier educational computer available.

suites—a problem they would have when shifting from other "educational" computers to business machines.

If IBM is right about its view of Junior, this small but mighty com-

puter will be joyously purring away on desks in millions of businesses, schools, and homes around the world. But like all personal computers, Junior has shortcomings as well as advantages. The machine is not the best computer in the world, nor is Junior the best computer for everyone. Some other computers claim to be easier to use. Several are cheaper. Many are more powerful and faster.

Although Junior cannot be the one perfect computer, it does give one of the best combinations of features—ease of use, power, speed, and low cost. It's too expensive to be a toy, but give it a job and it's well worth the price. Junior is not a machine to buy just to have a computer; it's not a box to watch and "gee-whiz" at; it's a machine to use, take advantage of, and enjoy.

Learning to Use Junior

Perhaps the only major problem you'll have with Junior is getting adjusted to its particular needs and learning how to use it to its best advantage. When IBM's television advertisements brag that their little Junior can do nearly everything, they somehow neglect to say exactly how and what you need to do anything. Perhaps they assume that if you're smart enough to choose a computer made by IBM, you're smart enough to figure out how to use it. Fortunately, if you have patience and the right help—like this book—learning to use Junior or any personal computer won't be difficult.

Learning to use Junior, even for business, should be enjoyable; that's the computer's big advantage in education. The right computer can make learning anything fun—math, spelling, accountancy, and even computers. As a happy medium between the frivolous game-playing machines and bigger, more expensive gray-flannel business computers, Junior doesn't force you to work under an evil green stare—rather, you can use it to make either a rainbow of colors on a regular television set or columns of figures straight as an Army roll call on a high-resolution computer monitor. Junior gives *you* the choice.

Junior will give your entire family an incentive to learn about computers. Most adults involved in the business world have good reasons to learn to use the computer. Working with a computer can be a goal in itself for adults who can use the machines to handle bookkeeping, manage files, or type letters. But running a business or balancing a budget will hardly steal your kid's time from video games or Saturday-morning television. Junior, however, can challenge them, encourage them, even make video games

worthwhile by letting your children write their own games and other programs. Along the way, they'll learn all the fundamentals of computer programming and maybe even how to think more clearly by organizing their approaches to problems.

Don't think of Junior as only a home computer, however. When IBM's engineers turned from designing business computers and created Junior, they added new capabilities and special features to make Junior particularly adept at handling graphics for business and games, as well as the sounds necessary to make it a fun machine to play—and work—with. In effect, they combined the best of both the work and play worlds.

Locked inside Junior's chassis is a microcomputer-on-a-circuit chip armed with more memory than most $10,000 business minicomputers had a few years ago. Junior can put that power to good use, handling most of the programs that the more expensive IBM PC and PC-XT regularly run. With IBM's blazingly fast cartridge BASIC language or IBM's easier-to-use and even more powerful Logo language, that computer power can easily be turned to almost any purpose you want.

The Mac Attack

Compared to other computers, the new PC*jr* is a big winner. In days past, critics unfairly pitted the PC*jr* against Apple's Macintosh, if only because the two were made available at about the same time. The comparison was unfair because the PC*jr* costs less than half as much as Mac and is probably the *better* computer.

If you had any doubt that IBM took this comparison to heart, you need only look at the *ColorPaint* software which seems aimed straight at stealing customers away from Mac. And price is not the only advantage; the PC*jr* is much more versatile than the competition from Apple. You're not limited to one tiny, black-and-white display as you are with the Mac. You can get yourself a big, businesslike green or amber monitor or take advantage of full color computing—new hues you can't get on Mac at any price. You're not limited to using a mouse to run the Junior, but if that's what you want, you have your choice of at least two.

If you want more than one disk drive, you can add another industry-standard, 5¼-inch double-sided disk drive to the Junior's native endowment or boost both speed and capacity by an order of magnitude with a hard disk. And you can easily transform some of the PC*jr*'s massive amount of add-on memory to work like a second disk drive at astonish-

ingly quick electronic speeds. Although a modem is an afterthought for Mac, the Junior gives you an internal slot reserved for one (and even the standard IBM internal modem is reasonably priced).

In addition, the Junior can grow. The Mac has only a pseudo slot, a connector on its back panel that allows you to connect accessories. But because the pseudo slot is *not* a real extension of the machine's internal spinal column, called the "bus" by computer engineers, any accessories you add to it suffer a speed limit—one that slows high-speed devices, like Winchester hard-disk drives, down by a factor of five. The Junior's expansion connector on its side is a real bus extension, and that means *no speed limits.* Anything that you add to the Junior becomes a true part of the machine.

Most importantly, the Junior has "open" architecture, which means that anyone can buy all the technical data on its design to build accessories and write programs for it (and many have and will). Mac is a "closed" system. Only the privileged are allowed to add on. That means you'll likely see many, many more programs and accessories made for the Junior than any closed-architecture machine. (That means many more than the thousands of programs that *already* run on the Junior!)

Unlike Mac, which was designed with many high-level functions built into its permanent read only memory (ROM), the Junior is designed to be as versatile and adaptable as possible. Which philosophy works better is hard to gauge, but note that the functions of the *MacPaint* program which took three or four years to build into Mac were added to the Junior in less than six months—with the added benefit of full color.

Of Bits, Bytes, and Bragging

Some Apple fans claim that Mac is a 32-bit computer, but it really isn't. Even the manufacturer of Mac's 68000 central microprocessor refers to the Mac's brain as a 16-bit chip. It has a 16-bit arithmetic-logic unit (or ALU), where the processor's calculations are actually done. While the Mac has a few 32-bit registers, its data bus is only 16 bits wide. (In fact, Apple never comes right out and says that Mac *is* a 32-bit computer. The advertisements only hint at this with words like "32-bit architecture.")

Both the PC and PC*jr* are built around 8088 microprocessors that have 16-bit ALUs, but data buses that are only 8 bits wide. In many ways, therefore, the PC and PC*jr* are only 8-bit computers.

In the overall scheme of computing, bits really play a minor role. As a

What's a Byte?

The computer term *byte* is just the computerphile's shorthand word for 8 bits of digital data—a convenient number in computing because 8 bits are enough to digitally code 256 different characters. One byte in the digital computer's language is, therefore, sufficient to represent all the upper- and lowercase letters of the alphabet, all the numbers from 0 to 9 and well over 100 special symbols that can be used to control the computer. Because 1 byte of memory can hold 1 letter of the alphabet, you can consider a byte to be 1 character of memory.

measure of computer power, the bits by themselves don't matter—it's their effect that is important. More bits should mean faster, but that's not necessarily the case. Although for straight number crunching, Mac will rush through digits faster than the PC*jr*, in true day-to-day work the PC*jr* is much more of a speed demon, with faster-loading programs and faster access to data on disks. You'll probably wait a lot less with the PC*jr*.

Bits are very important in terms of memory capabilities. Mac and the PC*jr* are about equally matched in their most popular configurations, both with 128 kilobytes of memory. Mac, however, has to wait for expensive new circuits to be designed and made to increase its memory capacity—and then only to 512 kilobytes. You can expand your PC*jr* right now all the way to 640 kilobytes. (Although the PC*jr* was once called "crippled" because some of the features of the more expensive PC had been removed from its design, the Mac's design really cripples its microprocessor, which could, in better surroundings, make use of 16,000 kilobytes of memory!)

The Silicon Crystal Ball

Computers are very much part of our fast-changing world. Not only have they made problem solving easier, but they are also helping us think, even changing the way we think. They help us to know more and give us the capability of exploring ideas deeper than has ever been done before. They can help our intuitions by representing complex ideas graphically. They can help us find ideas and solutions by accessing huge data networks. They can help us record our ideas, and arrange and rearrange them for emphasis and clarity.

Junior is part of that computer revolution. More than that, Junior is *leading* the computer revolution. By examining Junior, you can see one vision of the future, as predicted by the giant company that has been responsible for shaping much of the world's computer industry. Peering inside Junior's case, you can see the direction in which IBM believes that small computers are going (and have so far gone).

Look closely at Junior and you'll see that it is something more than merely a scaled-down business computer. The circuits locked deep within Junior's system unit show that it is not just another number cruncher and text manipulator like the massive mainframes and powerful personal computers that have carried the IBM banner before it. IBM now believes that on-screen color graphics, not just ordinary green-on-black text, is a necessary part of this kind of computer. The circuitry inside the PC*jr* shows that instead of requiring an add-on adapter card for color and graphics as does its big brother IBM PC, Junior's multihued drawing abilities are built into even the least expensive model.

No longer is a personal computer complete in itself. It is part of a vast, intercontinental communications system that can share data at the speed of light. The modem is a designed-in feature that allows the PC*jr* to use ordinary telephone lines to search through on-line databases like CompuServe, or even operate other computers by remote control. Junior is even designed to join in business networks—the first networklike product announced by IBM included Junior from the start.

No longer is the minimal memory that used to be sufficient for playing games enough for a small computer. Junior starts where older machines leave off, then doubles that with an instant and inexpensive expansion device.

And Junior shows that personal computers are now for people—ordinary people—and not just experienced programmers. Although the Junior is a technologically advanced computer, it makes computing easy for nontechnical people to understand. Built right into the Junior's circuitry is a "game" that helps new owners learn how the computer works and how to use it—almost without instructions. Its built-in "diagnostics" makes finding problems easy if something goes wrong.

No longer are powerful personal computers reserved for a select few—Junior is here—*now*, and designed to be produced by the millions in factories. At an affordable price. The Junior is not riding the tide of the personal computer revolution: it's generating its own mighty waves.

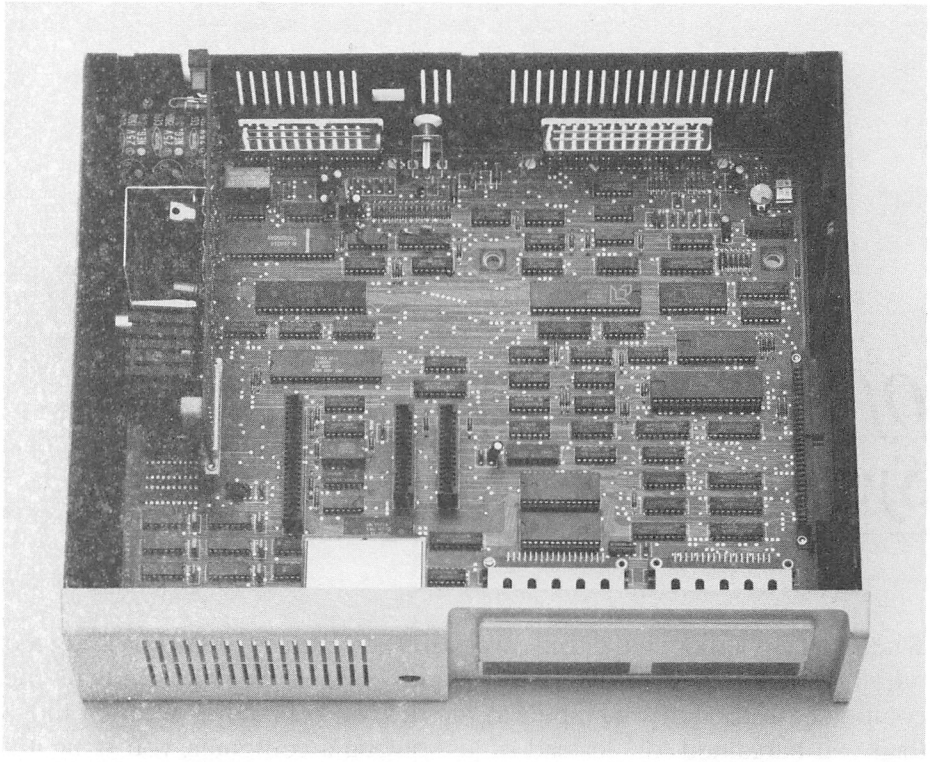
Of Systems and System Units

When you ripped open that big cardboard box labeled "IBM PC*jr*," tugged out the block of Styrofoam inside, and burst it apart to find all the computer components wrapped in plastic, you probably got much more than you bargained for. Instead of just buying a computer, you bought a very solid foundation for a complete computer *system.*

Although Junior has more computing power stuffed into its plastic case than any other calculating engine in its price range, its greatest strengths are the tasks it can accomplish when other devices are connected to it. The devices that will carry out the instructions that Junior sends to them are called *peripherals.* Printers and monitors are two important examples of peripherals.

Because you can see, touch, and feel these peripherals (like Junior itself), they and the computer are often termed *hardware.* The programs that control the hardware and put it to use—instructions that exist only as electronic signals—are called *software.* Together, hardware and software make up a complete *system.*

The difference between a computer and a computer system is much more than a tiny variation in terminology. Thinking of Junior as just a computer box makes it a mere blot on the top of your desk, to be discarded when something better comes along. Thinking of Junior as a system, however, means that it is more like an organism. It can gradually grow and develop, adapting to the applications you need. In a sense, Junior becomes something that is constantly changing—and becoming better.



The interior of the standard PC*jr*. Power supply is at the upper left-hand corner. Video circuitry is directly to the right of the power supply. The Intel 8088 microprocessor which runs the entire computer is the largest chip at the extreme right edge of the machine.

Certainly Junior is the most important and most necessary part of your computer system, but by itself Junior is just a machine that changes numbers and words into different forms. You can put letters and numbers into Junior and work miracles with them, but what good do they do locked inside its circuitry? You'll need to see what you've done, and show your results to the world. A printer is an essential peripheral device that lets Junior show you the results of its calculations.

When you expand Junior into a true computer system by connecting it to the right software programs, it suddenly can do things: print letters, mailing labels, and manuscripts, play games on a color video screen, balance the books of a small business, teach fractions and fractals, and even talk to other computer systems over telephone lines.

The power of any computer system is determined by the combination of its hardware and software, which always play complementary roles. The software is in second-by-second control, its instructions telling every

part of the system what to do. The hardware, on the other hand, brings the software to life and connects it with reality. Software can never exceed the limitations inherent in the hardware, so the hardware ultimately sets the capabilities of the computer system—what it is *possible* for the system to do. Therefore, the major factor determining its power is the hardware.

The System Unit—The Organizer and Controller

The heart of your computer system is the rectangular box that holds the microprocessor (the tiny electronic brain) and the digital integrated circuits (ICs). The microprocessor and the ICs control the whole system. The box that houses them serves as the main communications center, the switchboard that routes signals throughout the system. It's also the memory unit that keeps track of details, and the clock that makes sure everything happens at the right time. It's an artist that can paint glowing pictures across the monitor screen, and, of course, it's the central command center where all the data processing of the computer system is done.

Because this box is where all these centralized processing functions take place, it's often called the *central processing unit*, or *CPU.* IBM calls the box a *system unit*, because it is the foundation on which the whole computer system is built.

Junior's system unit was designed with the same adaptability and versatility that IBM put into its largest, most complex computers. In IBM's eye, even the smallest computer is not meant to be complete in itself like a handheld calculator, but must be capable of linking up with much larger, more inclusive systems. In fact, from the very beginning, when IBM offered its first personal computer networklike system, called the Cluster, it featured *ports* ready for Junior to plug into and play its part (see chapter 13).

Not only is Junior similar in concept to other IBM personal computers, it's very much alike on the inside, as well. Although the circuit board inside Junior's system unit looks quite unlike that in the PC or PC-XT, it's actually a rearrangement of many of the same parts and components rather than a totally new creation. Junior's system unit's main circuit board incorporates many of the PC's optional parts as standard equipment, like the color-display adapter and built-in serial port, along with Junior-only goodies like a special internal slot just waiting for a modem that will allow you to communicate with other computers over the telephone lines.

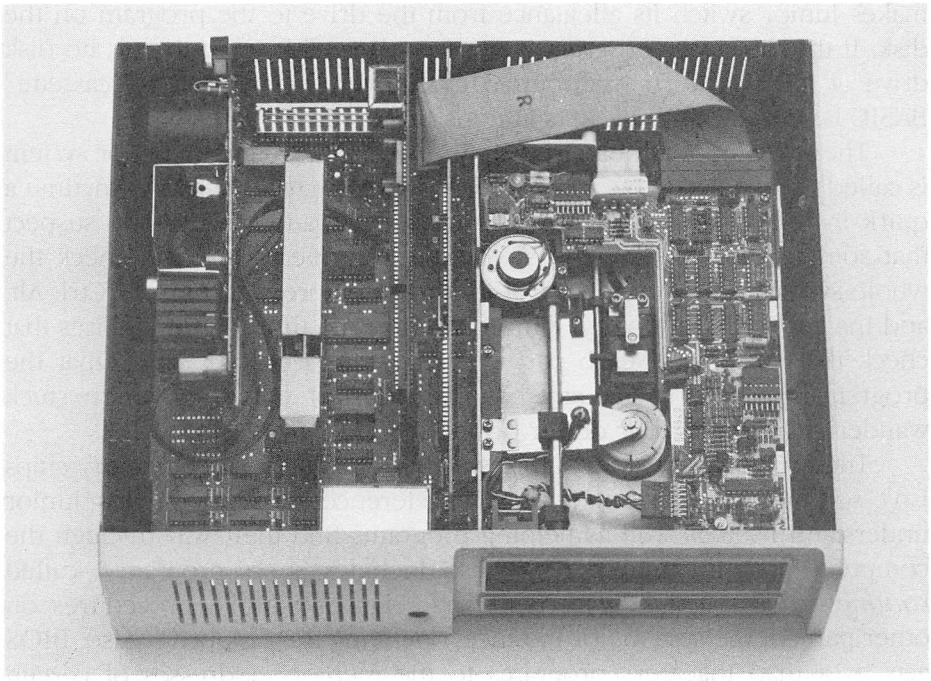
Junior's built-in memory is quite impressive. Like most other microcomputers, Junior stores data—measured in thousands of bytes, or *kilobytes*—in an area of temporary memory called *random access memory* (RAM). Even the less expensive "entry-model" Junior boasts a full 64 kilobytes (referred to as *K*) RAM soldered to the main circuit board, a minimum that matches the maximum of most earlier small computers.

The electrical arrangement of Junior's memory differs extensively from that of the other IBM personal computers. Because Junior's RAM is not parity checked, as the memories of other IBM personal computers are, Junior won't shut down if it detects something wrong with its memory but will continue to function if the errors aren't too severe.* That can lead to surprises when what Junior remembers and what you do aren't the same, but it also allows you an opportunity to correct things that other IBM computers do not, although the need for such corrections will be very, very rare.

All the necessary electronic functions to operate a video display are incorporated onto Junior's main circuit board. Junior uses its RAM memory to keep track of every dot on its display screen instead of requiring a separate add-on board with 16K of additional RAM, as older IBM personal computers do. This saves IBM the price of putting an extra 16K of RAM into Junior, but it cuts down Junior's useful memory by an equal amount and creates minor problems in adding memory to Junior. Too, some programs designed to work only on the IBM PC may not know about Junior's unique memory design and might try to use the display area of memory for thinking—with colorful but disastrous on-screen results. Fortunately, properly designed programs are smart enough to avoid the problem.

Because Junior uses RAM memory for the on-screen display, the memory that it has available for its thinking shrinks to only 48 kilobytes without adding IBM's memory enhancement, and to only 112 kilobytes of useful memory when you add the memory/display enhancement and run IBM PC-compatible programs. Even less memory (down to 96 kilobytes) may be available to programs when you use one of Junior's three new graphics modes, which eat up more memory and are not currently available on other IBM computers. (In fact, special graphics software can eat up to 96 kilobytes of Junior's memory—if the computer has enough extra memory installed.) That memory shrinkage can keep programs that squeeze

*Parity checking is a method used by relatively advanced computers, such as the IBM PC, to count the binary 1's and 0's in your data *twice*—and make sure the count is the same both times. If it's not, odds are that the data is being stored or transmitted improperly.



The interior of the enhanced PC*jr*. The PC*jr* 64K Memory and Display Expansion is directly to the right of the power supply. The 360K disk drive fills the entire right side of the chassis and is connected by the large flat ribbon cable to its disk-controller board.

tightly into the 64K or 128K memory of another IBM computer from fitting into Junior.

Besides random access memory, the contents of which change with each passing program, the system unit also holds a form of permanent memory—read only memory chips, or ROM—totaling 64K. Inside these permanent memory microchips are the programs that control the running of Junior's turn-on diagnostic and setup procedure (a program helpful in locating failures anywhere in your computer system), Junior's Basic Input/Output System (BIOS), and a short training program called *Keyboard Adventure*.

Every time you turn Junior on, its microprocessor automatically starts to read the instructions coded inside its ROM chips. Step by step, it checks the entire system, recording what expansion cards you've plugged in inside the system unit. The instructions in ROM then tell Junior to try running the *disk drive*—if it has one—to see if it contains a floppy disk with the *disk operating system* (DOS) recorded on it. If the drive is there and has a disk with the right program file on it, the turn-on program

makes Junior switch its allegiance from the drive to the program on the disk. If the necessary program is not found on disk (or if there's no disk drive at all), the turn-on program gives over control to the "cassette" BASIC language also built into Junior's ROM chips.

The program that helps you find problems inside your Junior system is called "Diagnostics" because it alerts you to the trouble spots. Anytime a quirk in Junior's normally smooth-running operation makes you suspect that something in the system is working improperly, you can check the whole system (or just one part of it) simply by pressing Junior's Ctrl, Alt, and Ins keys simultaneously. Junior will then run through procedures that check the function of each part of the system. The only thing that the program won't do is fix any bad parts that it finds, so you're stuck wandering down to your dealer or repair shop.

The BIOS—basic input/output system—inside the ROM memory chips isn't so much a program as it is a reference book that helps Junior understand itself as well as helping programs find their way through the computer system. Junior's BIOS also includes short programs—called *routines*—that tell it how to do some relatively simple procedures on other parts of the system, for instance, operating the disk drive. Also, BIOS acts as a map that keys programs to the memory addresses of certain computer functions, cross-referencing general labels of parts (like "serial port") to specific hardware locations inside Junior. BIOS helps programs be more compatible among a variety of different computers because the *generic* names of computer components that the program needs to find can be common among a whole range of machines. With BIOS pointing the way, the program can easily find any specific hardware function—like the memory byte for one dot on the display screen.

Cassette BASIC is a language program that makes your job of communicating with Junior a little easier. It translates your simple commands—made of numbers, common symbols, and words that look somewhat like English—into the strange and esoteric commands that Junior knows how to carry out. The version of cassette BASIC locked into Junior's ROM memory is almost exactly the same as the version used by the more expensive IBM personal computers. It's a whole computer language in itself, but it can be extended by plugging IBM's BASIC cartridge into one of Junior's slots. The cartridge and Junior's built-in BASIC then work together, with part of the program in each place.

The BASIC in Junior is just different enough from the BASIC inside the ROM of bigger IBM computers that the BASIC language extension program designed for the bigger computers (and recorded on the disks sold with IBM's disk operating system, PC-DOS) won't work with Junior.

Junior's BASIC offers the same functions and suffers the same limitations as many other BASICs because it is part of a larger family of BASIC languages published by Microsoft, the maker of PC-DOS.

Keyboard Adventure, the program locked inside Junior's ROM, is intended to be a fun lesson in how to use the system's Freeboard, but if you're already familiar with a typewriter keyboard, you probably won't be challenged. *Keyboard Adventure* may actually be meant to serve as an overall introduction to the computer for those who know little about how it works.

Inherent in the concept of a system is expandability, the ability to change the computer system and increase its power as new needs arise. IBM designed Junior's system unit to make it easier to enhance than any other personal computer. When you change or expand the circuits inside its system unit, you don't have to make any adjustments like throwing switches or moving jumper wires. Rather, Junior checks out what you've provided it with every time you turn it on.

IBM also has made expansion surgery inside the system unit safer by using an external low-voltage power supply, provided through the heavy, black plastic "brick" you found packed in the box with Junior. The brick is nothing more than a power transformer that reduces the 120 volts of AC electricity your electric utility supplies as "house current" into the safe, low level of about 16 volts. A power-supply circuit card inside the system unit not only changes this low AC voltage to 12 volts DC (as well as positive and negative supplies of 5 volts DC), but it constantly monitors the current Junior uses and instantly switches off the supply if a malfunction causes it to demand too much power. Because of the low voltages and protection circuitry inside Junior, changing enhancement circuit boards is fairly safe.

The external power supply also helps the circuitry inside Junior's system unit last longer. The biggest enemy of an electronic device is heat: too much heat can drastically shorten the life of solid-state semiconductors. Making Junior's power transformer a separate unit instead of building it into the system unit removes the greatest single source of heat.

Junior's external power supply is not perfect, however. Although you think you're turning the power off when you throw the switch on Junior's back panel, you don't in fact stop the flow of electricity through the transformer. Even when the power switch is off, the transformer is still drawing power from the wall outlet. Feel the brick when it's plugged in and you'll notice that it's warm, which means that it's using electricity whether you're computing or not. So play it safe and unplug Junior's power supply whenever you're not going to be using it for a while.

Another problem is that if the transformer is defective (or fails in operation), it might cause a fire, notwithstanding its built-in protective circuitry.

Expanding Your System

There are several ways to add to your Junior to make it more powerful. You can plug up to 128 kilobytes of read only memory (ROM) into Junior's dual cartridge slots to increase its BASIC vocabulary or to give it new program capabilities.

Junior's cartridges are little more than convenient plastic cases for additional ROM chips. If you take one apart, you'll find nothing but a small circuit board, a ROM chip or two, and some electronic components.

Junior's memory cartridges are more convenient than those of some other computers. The manufacturers of many machines that use cartridges warn against plugging or unplugging the cartridges while the power to the computer is on. Plug or unplug a cartridge while the power is on—even unintentionally—and the result can be a dead computer. Not with Junior. You can plug and unplug IBM cartridges whenever you want.

You probably won't want to insert or remove a cartridge while you have important data in your Junior's memory, however. When you push a cartridge into Junior, or pull one out, memory is reset. (This also happens when you press the Freeboard's Ctrl, Alt, and Del keys simultaneously.) In that way, plugging in a game cartridge, for instance, can automatically start the program running. But it also destroys the old program that was running—and all the data you had typed into it—and everything else that was previously in your computer's memory.

Your computer is automatically reset by the clever design of IBM cartridges. Look at the gold contact fingers on the front edge of any IBM cartridge and you'll notice that the last finger on one side is formed into an *L* shape. The part of the L parallel to the front edge of the cartridge temporarily shorts out a contact whenever you slide that cartridge into the slot, closing a circuit, which tells Junior to reset its memory.

You cannot add to Junior's RAM memory through the cartridge slots (as many engineers hoped you could when Junior was first introduced). Junior's cartridge slots are one-way streets for electrical signals; data can be taken from them, but it cannot be sent to them.

But you can add more RAM memory to Junior in two ways. For a modest memory increase, adding 64 kilobytes to the entry-model Junior's native endowment to push it up to 128K, you can plug IBM's official

memory/display adapter module into the system unit. Only one such module can be added to an entry-model Junior. Enhanced Juniors are sold with this module already installed, so you cannot add this module to an enhanced Junior to get more memory. The memory/display module is necessary to use Junior's disk drive and to show off its high-resolution graphics abilities. Needless to say, you'll probably also want to have a disk drive and/or a high-resolution monitor to put those abilities to good use.

IBM has made it easy to add memory and other enhancements to Junior externally by providing an input/output bus-expansion connector on the right side of the computer, hidden behind a plastic trim panel. You can pry this thin panel off and see the connector. By plugging one or more "sidecar" expansion modules into this connector, nearly any amount of additional memory can be added to Junior's factory endowment. IBM sells additional memory in add-on blocks of 128K, and you can add several of these to Junior.

But there are a couple of limits. Junior's 8088 microprocessor can address only about 1 million bytes (a megabyte) of memory, and IBM has designed *all* of its personal computers to allocate only 640 kilobytes of that capacity for normal program memory.

Another limit on the number of expansion memory modules that can be installed on one Junior is the small extra amount of electricity available from Junior's internal power supply. That power is just sufficient for adding one printer port *or* one module of extra memory—and no more. Quantitatively it amounts to about 33 watts, roughly half that of the IBM PC's 63.5 watts.

If you want to add more memory beyond a single expansion module (or other accessories), you'll need to give your Junior a power boost. IBM sells an add-on power supply that plugs into Junior just like other expansion modules. IBM recommends one additional power supply for every three expansion modules you add. (You must be careful where in your system you put the add-on power supply. You mount the power supply between the modules you want to power and the Junior system unit.)

Junior's design, which assigns RAM memory to the video display, also requires that you run a special configuration program to put any additional memory beyond 128 kilobytes to work. Such programs are usually provided with the memory enhancements that you buy from IBM or outside suppliers.

Junior's side-mounted expansion slot can be used for more than merely memory. When Junior was introduced, IBM offered only one peripheral to plug into this slot—an adapter for a parallel printer. Although you could stack several printer adapters onto your Junior (the

design IBM chose for Junior's expansion modules allows any number of sidecar modules to be "stacked" sideways on a single computer), only one IBM parallel printer adapter can be used at a time because each official IBM adapter automatically assumes that it is the only parallel printer being used in the system.

Other suppliers sell add-on modules that look very much like those made by IBM, but they overcome the limitations of the standard products. Some have their own internal power supplies. Many have multiple functions—memory, power, clocks that keep track of the time of day even when your Junior is switched off. As time goes on, more and more of these expansion products will become available from outside suppliers.

Beyond the System Unit

Although the system unit is the heart of your Junior, by itself Junior's system unit is not very useful. To bring Junior to life, you must add a variety of peripherals to its system unit. The Freeboard is your link to the computer and is so important that it comes with every Junior system that's sold. A monitor or television set shows you what Junior is doing and has done. A cassette recorder or disk drive gives Junior a more long-lasting memory. A printer gives you a permanent copy of your work, and a modem lets you reach out and get in touch with the rest of the world. And if you want Junior to do anything worthwhile at all, you'll probably have to add the right software. This book will explore in turn all the additions that you can make to your system to increase its power and usefulness.

The Freeboard

On Junior, as on most computers, your personal command center is the keyboard. By pressing its keys, you, a mere human, can order the sophisticated computer to do exactly what you want—at least if you keep your demands within the humble electronic device's capabilities.

The keyboard is your computer's primary *input device.* You use it both to enter data and to control the computer's own internal operations. Every time you press down on a key, the keyboard detects what you do and creates an electronic signal in a special digital code called a *scan code.* Each key has its own scan code assigned to it, so that it can signal to your computer which key you pressed. In effect, the keyboard translates your mechanical selection of a letter into digital data. The translation is necessary because to your computer, everything, including the words it prints and controlling commands, must be expressed internally as digital data.

Even among the wide variety of computer keyboards that are available, Junior's is both unique and novel. It is so different, even revolutionary, that IBM coined a new word to describe it: "Freeboard," the first wireless keyboard. It's cleverly designed, versatile, hardworking, and able to withstand most punishments without flinching, from spilled coffee to being trod upon by a two-year-old.

Okay, to be honest the Freeboard you'll get with your Junior is actually the computer industry's *second* wireless design. The first one was Junior's original Freeboard, now mostly a bad memory, particularly for

The new PC*jr*, with its attractive, full-travel typewriterlike keyboard and its 360K disk drive.

IBM. Because of the original Freeboard design, Junior's first few months were marked by teething pains.

The original Freeboard used a mechanism virtually identical to that of the current model except for two tiny differences—the shape of the keys and the force required to press them. The combination of high force and slippery little keys made the first Freeboard difficult to type on, and Junior difficult to sell. Supposedly the small keys gave all sorts of positive benefits like the ability to use "overlays," plastic or cardboard frames which could relabel the keys for different applications. The concept was called "revolutionary," but like many revolutions it was not successful. The old Freeboard and keyboard overlays are no longer manufactured.

The new Freeboard is a typist's dream. The touch is light, lighter than that of a standard Selectric typewriter or even the standard IBM PC keyboard. (It's one-third lighter than the original Freeboard.) The keys push down exactly the same distance as on the PC keyboard, so you get the feel of really typing. Unlike the PC keyboard, which is laid out for a compromise between programming and typing, Junior's Freeboard has a good, typewriterlike keyboard arrangement. The PC keyboard puts some of the computer programmer's favorite keys in the way of high-speed

IBM redesigned its PC*jr* keyboard to resemble that of the PC and PC-XT. Both this keyboard and the original Freeboard can transmit keystrokes to the PC*jr* using infrared signals rather than a cord.

typing. Junior doesn't. The Freeboard has a big, friendly Enter key on the right and Shift keys in their more normal typing positions.

If you were among the few who couldn't wait and rushed right out and bought a Junior despite the slippery little keys of the original Freeboard, IBM has not forgotten you. Admitting its mistake, IBM has made the unprecedented move of giving a new Freeboard to anyone who bought a Junior with the old one (from an authorized IBM dealer). Just take the sales receipt from your Junior to the dealer from whom you bought it, and you'll get a new Freeboard. It's not an exchange—you can keep your old Freeboard as a spare or as a collectors' item.

Underneath the new keytops, Junior's Freeboard is an innovation that will probably set a trend for the computer industry. Not only is it wireless, but it seems to be nearly indestructible. At the time Junior was introduced, no other keyboard in the world used its remarkable technology. Now other giant companies are eyeing it for their own machines.

The wireless Freeboard cleverly conquers a major problem suffered by other computer systems—once all the accessories are plugged into them, most computers turn into octopuses, with cables running off in every direction, dooming your every move with the keyboard to end in a tangle. Junior itself is not immune to such a confusion of cables: it has

over a dozen connectors to link it to necessities and accessories such as printers and high-speed modems. But no matter how much of a snarl Junior's wiring becomes, the Freeboard allows you to wander wherever you want, computing along the way, completely untethered, untangled, and unbothered.

Of course there's more to the Freeboard than merely a lack of wires, but first let's see how to use it. Then we'll investigate the real revolution inside it.

About Batteries

The first thing that you need to do to use Junior's Freeboard is stuff it full of batteries. Junior is probably the most expensive product sold with the famous legend "Batteries not included." After paying hundreds of dollars for your new Junior, you're stuck having to buy batteries for it!

Junior's battery demands are both pedestrian and particular. The Freeboard uses four AA-size batteries. You should buy alkaline or carbon-zinc (ordinary "heavy-duty") batteries only, not the more expensive rechargeable nickel-cadmium type. Alkaline cells are best; they will last three months (or longer) under normal use. Ordinary carbon-zinc batteries will work fine, too, but not as long.

Although you may be tempted to choose those expensive, rechargeable nickel-cadmium batteries for the Freeboard, that's not a good idea. Certainly they will work, but you'll be wasting money on their extra cost. Junior draws only a minute amount of electricity and may not drain enough power from "ni-cads" to allow them their proper charge-recharge cycle. Their capacity may be severely lessened—enough so that using them could become bothersome. In other words, stick with alkaline cells.

Wireless Operation

As free as the Freeboard makes you, you cannot wander all over Creation with it. The tiny LEDs in the Freeboard which actually transmit the scan codes to Junior's system unit have a limited range of about 20 feet for complete reliability. Even within that range, the Freeboard has its limitations. Because the infrared light that carries the scan codes behaves like any other kind of light and travels, for the most part, only in straight lines,

Junior's system unit must be able to see the Freeboard to be able to receive its signals.

Like the human eye, the eye with which Junior detects the infrared light has peripheral vision, but it has its limits, roughly in the range of about 55 degrees. If Junior's system unit is angled more than about 55 degrees from facing you straight on, it won't be able to see the Freeboard's signals. Nor will the system unit pick up the Freeboard's signals if there is an object blocking the light path.

If Junior can almost but not quite make out what the Freeboard is sending, it beeps. That's a warning that it saw the light but was not sure what it was. Rather than risking coming to a mistaken conclusion, Junior does nothing when the signal is not good enough. So if you hear a beep, type your keystroke again.

Better yet, as soon as Junior has run through its turn-on procedure and you're ready to compute, press the Ctrl (control), Alt (alternate), and Caps Lock keys on the Freeboard at the same time. That will tell Junior to give you audible feedback—it will make a clicking sound every time it sees an acceptable code signal from the Freeboard. You'll be reassured after each key press that Junior understood what you did.

If the clicking noise gets on your nerves or disturbs people around you, you can turn it off by pressing the same key combination again.

Sometime you may discover that Junior beeps for no apparent reason. That means it's received a signal from somewhere that it has interpreted as being very much like a Freeboard code. Other wireless remote-control signals might cause such a reaction from Junior. Or you might have put Junior too close to a monitor—either atop, underneath, or right next to it. If Junior is too near a monitor, some of the signals inside the monitor can leak into Junior and confuse it. If this happens, simply move your Junior away from the monitor.

You can increase the angle over which Junior recognizes your keystrokes in two ways. Probably the easiest way may be to turn Junior's system unit carefully on its side, making certain you put the side *without* cooling slots on the bottom. In many cases, Junior's detector eye will accept a wide horizontal range of signals when it's turned in that position. (Of course, you'll be trading less vertical range for the gain in the horizontal.)

Another trick is to sacrifice some of Junior's range in distance for increased angular perception by placing a "diffuser" over its eye. A piece of frosted tape works fine as a diffuser. Just cover up the round hole in the front of Junior's system unit. You can make the installation of your diffuser more appealing by putting the tape on the *back* of the hole on the inside

of the system unit. In this way, you may be able to increase angular coverage by about 10 to 15 degrees. Putting a diffuser over the two LED eyes on the Freeboard may also help to spread out their beams and increase angular coverage.

If you have a large, blank wall you can try another trick. Position Junior so that instead of facing you it faces the wall at an oblique angle. Then, instead of pointing the Freeboard at the system unit, point it at the wall so that the invisible infrared light beams bounce off the wall and back to Junior. Although your range will decrease, you'll have much more freedom in the direction you point the Freeboard.

The Un-Freeboard

If you try to use two Juniors simultaneously in the same room, you will hopelessly befuddle both machines. Junior's infrared transmission method cannot distinguish the signals of one Freeboard from those of another. A single Junior can become confused, too, when bright light blinds it; the infrared sensor can become so swamped with light that it is unable to make sense out of anything. You'll find that infrared-rich direct sunlight makes wireless operation of your Junior out-of-doors an unrewarding and almost impossible proposition.

For such trouble-making conditions, you can buy a keyboard cord from IBM to connect the Freeboard to Junior's system unit. The end of the cable with the silvery metal connector gets plugged into the *K* connector on the back of your system unit. The other end plugs into the telephonelike modular connector on the back of the Freeboard.

The Freeboard connector is actually a special design (which is patented) that automatically switches off battery power when the cable is plugged in. Similarly, when your Junior detects a cable plugged into the *K* jack, it automatically turns off its infrared eye to prevent getting confused by possible conflicts of wire-connected and wireless signals.

Because the cable and Freeboard connector are almost identical to those used in most modular telephone systems, you can easily (and cheaply!) extend your Freeboard cable using telephone wires and connectors. All you need is an extra length of modular telephone wire—even the coiled kind—and a cable-extender adapter (two female modular telephone jacks in a small plastic boxlike case).

Other Beeps

One Freeboard problem not so easily conquered is that your keystrokes will be lost if you type when Junior's optional disk drive is running (and its red warning light is on). If your disk drive is running at the same time you type at the Freeboard—no matter whether it's connected by infrared link or cable—Junior will obstinately beep and the characters you type will wend their ways to oblivion without gracing your monitor screen. That's not a flaw in the Freeboard, however. You lose keystrokes because Junior can't do two things at once, and when the disk drive is running, your Junior system unit cannot properly receive signals from the Freeboard. Junior is designed with the philosophy that a mistake in the data read from a diskette is potentially more dangerous than a missed keystroke. (After all, you only need to retype the character, but there's no telling what damage an errant byte read from disk can do.)

You might think that the clever Freeboard could save up your keystrokes while the disk drive is running, then blast them all away at Junior as soon as it is ready, but the keyboard has no way of knowing what is going on inside Junior. The wireless infrared link is a one-way affair, and Junior cannot send a request to the keyboard to wait up for it. Adding the cable won't help because the Freeboard circuitry is designed around the one-way wireless link.

The only solution to this problem is to add extra memory to your Junior using the memory modules sold by IBM or other suppliers and use some of that memory for an *electronic disk emulator*. Most add-on memory accessories (including those sold by IBM) include special programs which will make part of the additional memory act like a disk drive—but much faster. Further, the Freeboard takes priority over Junior's memory manipulations so Junior won't drop keystrokes (and beep) if you type at the same time the electronic disk emulator is being used.

Keyboard Comfort

The Freeboard is designed with your typing comfort in mind. To match the German industry standard (the DIN, or *Deutsche Industrie-Normen*), the Freeboard's keytops are designed to be about 1¼ inches above the surface the Freeboard rests upon. If that position doesn't feel right to you, the Freeboard allows you to set it to an alternate angle and height.

Underneath the Freeboard, near its rear edge, are two pivoting plastic legs, one on each side. Just pry them up with a finger and swing them around until they snap in place. Your Freeboard will then rest at a tilt 15 degrees greater.

P.C.'s Keyboard Adventure

Because Junior's Freeboard was apparently designed for people unfamiliar with typing, IBM came up with a unique way to teach the somewhat arcane arrangement of its keys—the *Keyboard Adventure* "game" built into Junior's ROM (read only memory). IBM recommends that you try *Keyboard Adventure* first, before you attempt doing anything else with your brand-new computer. It's the first assignment given to you in Junior's *Guide to Operations.*

Although the amount of fun in the game is dubious, *Keyboard Adventure* is a reasonable introduction to your new Freeboard and computer—requiring no prior knowledge of the workings of computers.

To start the game, all you need to do is press the Esc key in the upper left-hand corner of the Freeboard, immediately after you see the BASIC copyright message on the screen, and before you press any other key. However, a couple of things that IBM does *not* tell you at this point may hide the *Keyboard Adventure* game from you, seemingly forever. If you put a disk in the disk drive (if your Junior has one), you won't get the BASIC message on the screen, so you won't have a ticket to the adventure. Further, if you put the IBM BASIC cartridge—or nearly any other cartridge—into one of the cartridge slots, the game will not work. To run *Keyboard Adventure*, therefore, make sure that you do *not* have a disk in the disk drive or a cartridge in either cartridge slot.

Another fact that IBM doesn't tell you is that much of *Keyboard Adventure* is actually another, more important part of what your Junior calls "Diagnostics." Diagnostics are special programs that help you find problems with your computer system. You can take a shortcut to the heart of *Keyboard Adventure* by running Junior's Diagnostics program. Hold down the Ctrl and Alt keys on the Freeboard and then press Ins (insert). First you'll see a big IBM logo on the screen and about ten seconds later the display will change to pictures that represent parts of the system that you can test. Note the white box that flashes alternately with the number 1 under the upper left-hand symbol (which represents a floppy disk). By pressing almost any key on the Freeboard besides Enter—Fn (function),

Ctrl, Alt, Esc (escape), Shift, or Caps Lock—you can make this box step through the numbers and letters under the symbols on the screen. Try it using the spacebar.

Move the box to the letter *J*, then press Enter. You'll be greeted by the little person P.C. If you try using the letter *K* instead, you'll find P.C. atop a different keyboard, one with more keys, much like the keyboard supplied with the bigger computers in Junior's family.

In truth, *Keyboard Adventure* won't appeal to everyone. If you or your kids have ever tried a real computer game, you're not likely to find P.C.'s antics very mind boggling. For those who prefer a more conventional introduction to the Freeboard or who want to know more of what's inside of it than P.C. will let on, let's take a closer look at the Freeboard: how to work it and why it works (and looks) the way it does.

Typing on the Junior: The Freeboard Cursor Commands

Overall, the Freeboard bears a strong resemblance to a typewriter keyboard. In fact, the arrangement of Freeboard keys is almost identical to that of a standard typewriter; you'll find the familiar "QWERTY" format. It's much more like the almost universally praised Selectric keyboard than the almost universally disparaged IBM PC keyboard. That makes Junior's new big-key keyboard a top choice for experienced typists who want a word processor with a key layout like their old Selectrics. And if you don't know how to type, working and playing with Junior will help teach you where the letters are.

The obvious difference between typing on Junior and on a typewriter is that Junior displays your work on a video screen instead of on paper. When you type on a typewriter, the letters always appear in exactly the same place, and the paper moves. When you type on a computer, the screen always stays in the same place (unless you move it!), and the letters can appear almost anywhere on its surface. So that you don't have to keep guessing, computer displays and Junior use a marker called a *cursor* to show you where the next character you type will appear on your monitor screen.

On Junior the cursor is usually a thin flashing line, although it does take different forms depending on the program you are using. Other computers may also shape the cursor differently—sometimes as a rectangle that may or may not flash. No matter what it looks like, the cursor appears exactly where the next character you type will appear on the screen.

A typewriter has a knob or pushbuttons that you can use to move the typing paper around and position characters anywhere on the sheet. Junior lets you move the cursor around so that you can put characters anywhere on the screen. To control the cursor, use the diamond-shaped block of four keys in the lower right-hand corner of the Freeboard. These keys, marked with arrows pointing north, south, east, and west, are collectively called *cursor controls.* You can demonstrate to yourself how they work either by playing the *Keyboard Adventure* game (it's one of the primary points of the game), or just by exiting the keyboard demonstration and entering BASIC—and pressing them to your heart's content.

Note that each press moves the cursor one "step," and if you hold down a cursor control for more than about half a second, the control function begins to repeat at the rate of about fifteen steps every second. This automatic repeating function is called *typematic* and is a property of most of the Freeboard's keys.

Try filling Junior's display with row upon row of characters by holding down the key corresponding to your favorite letter of the alphabet. Notice how when you fill up a line and the cursor bumps against the right edge of the screen, it automatically flies back to the left side as if you pounded on a typewriter's Return key. This automatic return to the left side is called *wrapping,* or sometimes *word wrap.* It looks as if the letters have moved behind the screen from the right to the left, like string being wound around a package.

A few other keys have cursor-control-like abilities. The Backspace key acts like the backward-pointing arrow cursor-control key but with an important difference (in BASIC): as the cursor moves backward when you press Backspace, it erases any characters on the screen that the cursor moves under. You can also erase characters by moving the cursor under them individually using the regular cursor controls, then pressing the Del (delete) key next to the cursor diamond.

When you're using the BASIC language, the Ins key lets you squeeze in extra characters. Normally, when you move the cursor to the middle of a word and type a new character, the new character replaces the old one that was located over the cursor. Try it. Now move the cursor to some place where you'd like to squeeze in an extra letter, press Ins and then the characters that you want to add. Note that all the characters to the right of the cursor get pushed further right. Press Ins once more and the next letter you type will again replace the character over the cursor.

Shifty Junior

Besides the cursor controls, Junior's Freeboard features a number of other keys that normal typewriters lack. Several of these keys act like a typewriter's Shift key and multiply the number of different symbols (and functions) you can elicit with simple keystrokes.

As with most typewriters, use of Junior's Shift keys allows each of nearly all the other keys to control two (or more) different characters, one shifted (in the "uppercase" position) and one unshifted (in the "lowercase" position). Instead of forcing letters against paper, however, Junior's keys send the characters as signals to its system unit. Each character or signal tells Junior to do something different—most commonly, to add a letter or a number to a file and show it on the monitor screen. The typewriterlike Shift keys double the number of signals you can send to Junior through the Freeboard.

Tucked near the lower right-hand corner of the Freeboard is a Caps Lock key. It does not function like the Shift Lock key of a typewriter. Press it, and although it pops back up immediately, Junior acts as if it were locked down. Press it again, and Junior thinks that it has popped back up. The on/off function of the key is called a "toggle," because it electrically switches between two positions like a toggle switch. When Caps Lock has been toggled on, Junior acts as if you're holding down the Shift key all the time—with one important difference. Caps Lock shifts only letters, and does *not* shift the number and special function keys.

In addition to the normal shifted characters that you get with ordinary typewriters, Junior also has Ctrl and Alt keys that perform shiftlike functions by making nearly every key on the Freeboard perform double duty. Ctrl pressed at the same time as any other key tells Junior something different from the instructions it gets when the key is pressed alone. Junior's three shifts, Shift, Ctrl, and Alt, nearly quadruple the number of different signals you can send to Junior.

The Fn, or *function*, key in the upper right-hand corner of the Freeboard and the Esc, or *escape*, key in the upper left-hand corner can also multiply the effective number of keys. However, they work a bit differently. Instead of pressing them at the same time as you press another key, you must press Fn or Esc *before* you press the key. For instance, pressing Fn and then the up-arrow cursor control will move the cursor to its "home" position at the upper left-hand corner of the screen. That's why the green legend *Home* is printed under the arrow on that cursor-control key.

Note the green bar underneath the label on the Fn key. The green color coding lets you know which other keys work in conjunction with the Fn key. If you have a printer connected to your Junior, for instance, you can print out the image that's on your monitor screen by pressing the Fn key and then the *P* key, which has the green *Prt Sc* legend on it. Or press Fn then the letter *B*, which is also embellished with the green word *Break*, to send a signal to Junior that tells it to BREAK (stop running) a program.

Pressing Fn and then a number sends a signal to Junior, different for each number, to carry out special commands. (Note the green legends on each number key—F1 through F10. These labels correspond to the legends on the dedicated function keys on the larger IBM PC keyboards.) The exact command issued to your computer when you press these function-key combinations depends on the software that you are using at the time. In BASIC, these function-key combinations put special command words on the screen.

The other green legends on the keys—Pause, Echo, Sc Lk, Pg Up, Pg Dn, and End—refer to special functions that are called on when Junior is running certain specific programs, such as its disk operating system (DOS) and BASIC.

Note, too, the blue color-coding bar on the Alt key. Hold down the Alt button at the same time you press one of the keys that have an extra character printed in blue, and the symbol in the blue legend will appear on Junior's display. (Pressing Alt and other keys does some pretty spectacular things when you're using BASIC: each key can make a whole word appear on the screen. You can find a list of the combinations in chapter 6 on BASIC.)

If you have a good imagination, you might see an invisible blue asterisk (*) on the period/greater than (>) key. Although IBM doesn't tell you about the feature or even hint that it's there, when you hold down Alt while you press the period Junior will put an asterisk on the screen. (This "undocumented" feature will be a big help when you start working with DOS.)

The Freeboard's Inner Secrets

Junior's Freeboard is a big step ahead of even the IBM PC standard keyboard. Unlike the PC keyboard, the Freeboard is almost completely unaffected by moisture—while the PC may stop functioning if you get its keyboard wet with sweat, rain, or spilled coffee, the Freeboard will

continue happily to work. If something does happen to your Freeboard, odds are that it can be repaired. Many repairmen term the IBM PC keyboard a nonserviceable item—if it breaks, you've got to buy a new one. Nevertheless, the Freeboard has a light, reassuring feel that makes it easy to type on.

Although it bears the unmistakable IBM logo, Junior's Freeboard is actually custom manufactured for IBM, to its demanding specifications, by Advanced Input Devices, located in Coeur d'Alene, Idaho. The Freeboard was the first major product to use AID's ErgoKey technology, a new keyboard mechanism that combines very few parts (keeping its cost down) with long life and most important, near-perfect tactile feedback.

The secret is the underlying ErgoKey mechanism of the Freeboard which puts a resilient rubber boot under each key that acts as a spring to keep all the keys in their normal "up" position. The shape of the boot was carefully designed to collapse in a certain way under the pressure of your finger. First it puts up a big fight. It actually requires more force to press down than a standard typewriter key does. Then suddenly it becomes easier to press—it almost pulls your finger down. Near the bottom of its travel, the key requires more force again. The initial high pressure prevents you from accidentally pressing a key, and the two changes in the required pressure give you the reassurance that you've indeed pressed a key (pressure is lower) and that you've finished making the keystroke (pressure gets higher again).

The rubber boots of all the keys are molded together in one piece called a matrix. Inside each boot is a carbon button that connects two electrical circuits on a large underlying printed circuit board when the key reaches the bottom of its travel. That's all there is to the keyboard.

The simple keyboard mechanism permits the Freeboard to be more compact than traditional keyboard designs. Its low height is important because the German DIN keyboard standard specifies that the tops of keys be only about 1¼ inches above the surface the keyboard rests upon. German research indicated that higher keyboards make typists bend their wrists more, which produces more fatigue.

The light weight of the Freeboard, attributed primarily to the clever construction, and its electrical design, which uses direct electrical contacts, make it possible for the Freeboard to be wireless. Most other computer keyboards do not rely on switches making contact in order to detect keystrokes. Rather, they detect a change in a key's position by sensing a change in an electrical property like capacitance or inductance. Because no physical contact is made, these keyboards don't wear out—at least not very fast. Most are designed to last for 100 million keystrokes.

In noncontact designs, the electrical circuits that detect moving keys must always be operating so that keystrokes are never missed. Because the circuits are always active, they continuously draw electricity. This current drain is no problem as long as the keyboard can suck electricity from the computer that it is connected to, but the need for a constant supply of electricity makes noncontact keyboards much less desirable for wireless designs. The detection circuits inside a noncontact keyboard would quickly drain the life from any but the biggest batteries.

Junior's Freeboard was designed to draw battery current only when you press a key, so its batteries last a long, long time. Yet the materials used in its ErgoKey technology give the Freeboard an expected lifetime that's nearly as long as that of the best noncontact keyboards. Little wonder then, that several other large computer companies are starting to use the same design in their keyboards, many of which will be wireless.

The Economics of Wireless

There's another excellent reason for making keyboards wireless, one that makes sense for the computer *maker*. Wireless keyboards can cost less. The simple cable that connects a keyboard to its main unit can be a costly addition to any computer system. Junior's Freeboard cable, for example, costs you a hefty $30. Certainly wires are cheaper by the million, but even big manufacturers pay $5 to $10 for a good keyboard cable—and prices are rising. Hence the trend to go wireless.

An infrared-linked keyboard like Junior's needs no wires. Although the additional electronic circuitry needed for the infrared link does add to the cost of the keyboard, the amount is not significant. Much of this circuitry must be built into cabled keyboards, anyway, to translate keystrokes into computer signals. Currently, the costs of cables and infrared circuitry are comparable, and, as production quantities increase and technology advances, circuitry costs are certain to go down.

Keyboard wires add a hidden expense to computer manufacturing: the cost of getting Federal Communications Commission (FCC) approval of the equipment, necessary because it has the potential to generate radio and television interference. Keyboard cables, like any wires that have electricity flowing through them, act like a radio antenna, broadcasting into the air the signals meant only for the computer. If the escaping signals are strong enough, they can interfere with television pictures and radio transmissions.

Before it can be sold, a computer must be tested and certified by the FCC to ensure that the amount of interference it produces is acceptable. The approval process can be grueling and often causes delays in getting new computer products onto the market. Although infrared keyboards also must be approved, they usually meet FCC standards much more easily than keyboards with cables do. They smoothly sidestep the issue of wire-generated interference because the FCC does not govern light transmissions.

Other Inputs

Although the Freeboard's design is clever—more so than most people imagine—it's not the perfect keyboard for everyone. It may seem too small, too lightweight, or too flimsy to you. You might not like the two-step function keys. You might require an extra, calculatorlike number pad to make spreadsheeting easier. Although they seem like little concerns, they are important. If you're not comfortable typing on the Freeboard, everything you do with Junior will turn into an irritation. Remember, nearly every command you make and question you ask must pass from your fingers to Junior through the Freeboard.

If you like Junior but not the Freeboard, you can choose among several more familiarly designed replacement keyboards for Junior from outside manufacturers. Or you can choose from an array of alternate ways to communicate with your computer: like joysticks, touchpads, mice, and light pens. In chapter 13 you'll see how wide your choices really are.

But no matter how elaborate any of these alternate input devices may be, today they still convey only a limited amount of information to the computer; that is, directions and locations but not complete ideas. Perhaps someday you'll be able to give your ideas to a computer by voice command or in simple written words, but today the keyboard is the preeminent input device. And the Freeboard is a big step beyond ordinary, old-fashioned keyboards.

CHAPTER FOUR

Mass Storage: The Cassette and Disk Drive

Perhaps the greatest advantage of a personal computer like Junior is its own disadvantage as well: a computer is only what its software makes it. To make Junior worthwhile, you've somehow got to feed it software that tells it what you want it to do.

There are four ways to enter software, in the form of programs, into Junior. You can type in programs manually, line by line, that you find in books and magazines. You can slide a cartridge on which a program resides into one of Junior's slots. You can literally play cassette-based programs through a tape recorder into Junior's memory (although these cassettes are not widely available). Finally, you can load programs from floppy diskettes. Let's look at each of these methods.

Entering programs by hand seems simple, but as you become more familiar with the computer, the programs you work with may be hundreds of lines long. Each line will consist of numbers, letters, and symbols in unpredictable patterns. Typing programs like this requires painstaking care, because one misplaced character will prevent the program from running. And if the power supply is accidentally cut off as you are working, the lines already entered will be lost.

Cartridges are no better when it comes to making your work indelible. Although the cartridges themselves are immutable, they cannot preserve the work you do with them.

Floppy diskettes, or *disks*, and cassette tapes are different. Together, they are a special form of memory called *mass storage*. Mass storage refers

to the place where Junior puts all data and programs it doesn't need immediately but still wants close at hand. It is permanent, but changeable memory. Better yet, it's exchangeable—not only can Junior store and retrieve programs that you've written from floppy disks or cassettes, it can read programs you've bought and use the same devices to load them into the furthest reaches of memory. Mass storage is a reservoir for programs that would overflow Junior's RAM memory, a rest home for programs (and collections of data called *files*) that, if they haven't outlived their usefulness, aren't immediately required.

Although a disk and a cassette tape are both storage devices, they work quite differently. As you would expect, each one has its own advantages and disadvantages.

Cassettes are the low-cost alternative. The tapes themselves are only a couple of dollars and the drive unit, usually no more than a standard portable cassette recorder, is available at discount department stores, stereo emporiums, and drugstores, as well as at computer stores.

A cassette is as close to a universal medium as there is in the world today; most standard cassettes will play in any brand or model of recorder. They're rugged and generally suffer all sorts of abuse—even roasting in a sun-drenched car—with impunity.

Floppy disks, on the other hand, are so delicate that you can damage one by writing with a ballpoint pen on its label. And, unlike cassettes, they are not interchangeable among different brands of computers.

Disks are "read" and "written onto" in drives. Not only is a disk drive an expensive part of the computer, but it is controlled by a complex internal program called a *disk operating system*—DOS for short. An operating system will probably cost as much as a cassette recorder, and a single disk drive may cost you twice as much as, or more than, some popular home computers.

But disks are blessed with both speed and versatility. You could make a sandwich while a long program loads from a cassette. A disk drive will do the same job in a few seconds. You can get almost any program available for Junior on floppy disk. Very few are sold on cassette. Professionals prefer disks. Given the choice, you probably will, too.

Using a Cassette Recorder

As universal as cassette recorders are, not every one will work with Junior in conjunction with IBM's standard but optional cassette interface cable. Although the cassette connector on the back of Junior's system unit allows for plugging the computer either into the microphone or high-

level (*aux*) input of a cassette recorder, the IBM adapter cable only works with aux inputs. Not all cassette recorders have aux-style inputs; in fact, only an elite few do these days. Most of today's portable cassette machines allow you to plug only microphones in, and the signals from Junior's adapter cable will cause so much distortion at these microphone inputs that the cassette system will not work properly.

If you need a cassette recorder to plug your Junior into, a relatively inexpensive machine ($60 range) is the Radio Shack model number CCR-81. It's designed for Radio Shack's less expensive computers but will work fine with Junior.

The IBM PC*jr* cassette adapter cable plugs into the *C* connector on Junior's back panel. The three connectors at the other end of the cable plug into the following jacks on the cassette recorder: the black plug goes in the *earphone* jack, the red plug goes into the *aux* jack, and the smaller gray plug goes into the *remote* jack.

Almost any cassette tape will accept Junior's signals. You may prefer to use short cassettes, however, so that you're not tempted to store so much data that you have to wait for hours to find what you want. "Special" computer cassettes, usually only special in their labeling, are available in five- and ten-minute lengths. You can easily use plain old ordinary audio tape cassettes, but for best results choose an inexpensive ferric oxide, "type I" formulation. More exotic tapes will be wasted on most pedestrian cassette recorders and some might not even work. Portable cassette recorders like those you can connect to Junior won't appreciate the fine distinctions of "audiophile" quality cassettes.

If you don't have a disk drive, the BASIC commands for using the cassette recorder are simple. First, prepare the cassette recorder by putting a tape in and pressing the Record and Play buttons at the same time. Then, to put a copy of a program that's in Junior's memory onto tape, just type SAVE followed, inside quote marks, by the name (up to eight letters) that you want to give to the program. (The quote marks tell Junior that the name is to be taken literally, not as data.) For instance, the following BASIC command would start the process of recording the program MISHMASH onto a cassette:

```
SAVE "MISHMASH"
```

It's important to have the cassette and recorder ready *before* you give this command because Junior does not check what it is doing. It won't tell you if the cassette recorder is not turned on or if there is no tape in it. Furthermore, Junior dumps whatever is in its memory onto the tape even

if the tape is already full of data. One way to avoid recording over an existing program is to play the tape back and stop it after the last of the squealy sounds of recorded data ends. Junior does roll the tape for ten seconds before it starts recording to skip over any leader tape in the cassette. You might want to play it safe and advance the tape in the cassette beyond the clear leader before giving control to Junior.

To put a program back into memory from a tape, first load the tape into the cassette recorder and rewind to a point just before the program begins, even to the beginning of the tape. Then press the Play button and tell Junior to LOAD the name of the file (putting it in quotes); for instance:

```
LOAD "MISHMASH"
```

If there are several programs on the cassette, Junior will work its way through them one at a time, listing each one on the monitor screen. And if you get tired of waiting, you can press the Fn key then B(reak) to immediately halt the LOADing process.

To LOAD the next program from a cassette tape no matter what its name is, press the Ctrl and Esc keys on the Freeboard at the same time. This keyboard combination is particularly handy when LOADing programs from your friends' cassettes. (It was probably designed into Junior to make prefabricated programs sold on cassette tapes easy to LOAD.)

When you try to rewind your tapes, you'll note that as soon as you plug the remote connector in, Junior takes over the cassette recorder and won't let it move the tape without a LOAD or SAVE command. The Rewind button won't work. To regain control of the cassette recorder, give the BASIC command MOTOR, which tells Junior to turn on the recorder's motor. Another MOTOR command will turn the motor off.

BASIC gives you much more extensive control over the cassette recorder than these few commands demonstrate, and you should familiarize yourself with the rest of its features.

The Magnetic Personality of Mass Storage

A mass-storage device permanently preserves computer programs and data that are in a computer's temporary (RAM) memory. Junior is prone to forgetfulness because the part of its brain that remembers—the random access memory—depends on integrated circuits that require a constant supply of electricity to function. Remove the electricity, and Junior's thoughts and memories evaporate faster than raindrops on hot pavement. Consequently, all the programs, formulas, and information that you type

into Junior's RAM remain there only at the good graces of its electrical supply. If your lights wink at you for a fraction of a second during a minor electrical interruption, every byte of information that you've perhaps spent hours entering will be lost.

Junior's mass-storage device, whether cassette or disk drive, relies on an entirely different storage principle than its electronic short-term memory is based on. Data is stored on tapes and disks using magnetism. Unlike electricity, which is essentially tiny bolts of lightning that prefer to be always on the move trapped in wires and circuits, magnetism likes to stick in one place. A magnetic field can be permanently anchored to many metal-based particles; electricity usually drains off shockingly fast.

Cassettes and disk drives work like most other magnetic-storage media, including both exotic and everyday devices. The operating principle is the same for all magnetic devices: the electrical impulses of the music in your hi-fi or data in your computer system are converted into pulses of magnetism in an electromagnet—often called a *magnetic head* or *read/write head*—that serves as the part of the magnetic recorder that does the actual recording.

The magnetic field produced by this head can induce permanent magnetic fields in a suitable storage medium such as the magnetic-responding compounds that coat the surfaces of cassette tapes or computer disks. When the head's magnetic influence is removed, the field set up in the recording surface remains there permanently. The whole recording process is just a sophisticated form of magnetizing a piece of iron by rubbing it with another magnet.

No matter how exotic magnetic storage devices may be electronically, most of them rely on some form of mechanical machine. That's easy to understand—if the magnetic medium did not move in relation to the magnetic head, the head would try to record all of its data on the same single patch of material, and each new bit of information would be recorded or "written" over the previous one.

The easiest way to overcome the problem of instant erasure is the method used by the tape cassette. In a cassette, the magnetic medium, a long strip of polyester tape typically coated with iron oxide, is constantly moved past the read/write head so that a different point on its surface is continuously brought into contact with the head.

Although theoretically simple, it is complex to use a cassette recorder with Junior because of a signal difference. Cassette machines are designed to record analogue signals, which vary in strength to match variations in the loudness of sound frequencies. The magnetic signal strength is directly analogous to the sound intensity, hence the name "analogue."

But the signals that Junior uses to receive, process, and remember data are *digital*, and digital and analogue signals don't mix. In order for a normal analogue cassette recorder to handle Junior's digital output rather than music or voice signals, the digital data in Junior's memory must be converted into tonal variations that can be recorded like any other sound. When you want to put information or a program back into Junior's memory, the cassette recorder plays back those tones as if they were a symphony and Junior converts them into data.

Magnetic cassette tape recording has disadvantages, too. Cassettes are called "sequential" storage devices because whether music or data are recorded onto a tape, each selection must be recorded one after the other, in sequence. Information is stored on tapes in a continuous data "stream." Without elaborate "automatic music sensing systems," the cassette player must read all the selections (cuts, programs, data, or files) in the exact order in which they were recorded, from the beginning of the tape to the end. And you can't hurry the process up: computers are sensitive to the exact speed that the tape travels, so you can't set the tape to whiz by in fast-forward and expect its electronic ear to tell one weird squawk from another. You'll understand just how inefficient a mass-storage device cassette tape is the first time you wait for Junior to hunt through a whole length of tape to locate a specific piece of data.

Different Directions

The cassette is not the only solution to the instant erasure problem. Once an engineer makes the obvious decision to move a storage medium to record different data bits in different places, he has to decide which to move—the magnetic medium and/or the magnetic heads—and in what direction. The possibilities are both complex and perplexing.

The cassette tape has a very straightforward design. The tape simply moves from left to right past a stationary read/write head. This method is inexpensive and reliable; linear-motion tape recording has been widely used for almost two decades.

Data is always stored on tape one-dimensionally—in a straight line, one bit after the other for the full length of the tape.

You may have learned that the shortest distance between two points is a straight line. But cassettes complicate matters by showing that the shortest distance between two bits of data on a tape is usually a long time. To move between two widely separated points, you must pass through all intervening points, more a local than an express trip! At the slow speeds at which cassettes operate, finding the data you need can take a long time.

In theory, there's nothing wrong with sequential-storage schemes—some can be very fast. One form of solid-state computer memory, all-electronic "shift registers," move data sequentially at nearly the speed of light. Most practical mechanical mass-storage systems, like tape cassettes, operate somewhat slower than that, however. Light travels at 186,000 miles per second and cassette tape races by at 1⅞ inches per second, more than a minor difference. In computer terms, cassette tape speed amounts to only about 1,500 bits per second.

This means that when you need to retrieve a program from the end of a tape and you must start at the beginning (tens of thousands of bytes away), you'll have time to read half of this book.

Data recorded onto computer disks is organized differently. They are stored on the large flat area of the disk's surface in *two* dimensions, like lanes in a circular racetrack. The disk itself moves in one dimension, making the circular track, and the read/write head moves perpendicularly to the spin of the disk, to create the "lanes" of the track. Although the shortest distance between two points (or two bytes) remains a straight line, to get from one byte to another, the read/write head can take shortcuts across the lanes of the racetrack. Because the head can jump from byte to byte at widely separated locations on the disk surface and because data can be read and retrieved in any order or at random in the two-dimensional disk system, disk memory units are often called "random access," just like RAM memory chips.

A disk drive combines the best qualities of a tape recorder with the best qualities of a phonograph. Like a tape recorder, the floppy disks that Junior uses can be recorded, played back magnetically with utmost fidelity, and even erased to be used again. Phonograph records and floppy disks both permit random access. Just as you can drop the needle down anywhere you want on a record, Junior can almost instantly locate data that was recorded anywhere on a floppy disk. Junior can drop the "needle" of its disk drive—actually a movable magnetic head like the magnetic head of a tape recorder—anywhere on the disk for any amount of time, then go on to another selection.

The random-access ability of computer disks and disk drives makes the combination much faster than tape cassettes for the mass storage of data. In fact, disks are so superior and so much more convenient than cassettes that industry experts have predicted that 90 percent of all Juniors would be sold "enhanced" by a disk drive.

Finding the Way

One of the great verities of life is that every solution inevitably leads to another problem. Once wonderful random-access devices like disk drives were developed, the problem that arose was finding a place to store data after it had been recorded. With both the head and the media itself moving constantly—and in different directions—some means of organizing all of the elements became necessary. Junior had to learn to arrange data so that it could find it when it needed it again.

Computer disk drives use complex schemes to set up place markers to track disk-bound data. Special signals are recorded on the disk to mark it into *tracks*, concentric circles on the disk surface similar to the grooves on a phonograph record; and *sectors*, small segments of each track. Each individual sector is assigned a different number, which serves as an address for the disk-drive system to find the block of data contained on it.

Most disk drives need at least one physical place marker to help find the magnetic ones recorded onto the disk. Usually the disk is punched with a small hole called an *index* hole, and a light is shone through a matching hole in the disk's sleeve. Every time the disk makes a revolution, the light flashes through to the other side where it can be detected. The flash tells the disk drive where to begin the first sector of each track. Because the rest of the sectors are then marked magnetically, the disk drive can make them any length that it wants—the exact number of sectors or their length is not mechanically fixed. Because the size and number of the sectors can be programmed like computer software, disks with a single hole are called *soft-sectored.* Junior uses soft-sectored disks.

Another scheme uses a number of holes that flash through the single hole in the sleeve, usually 10 or 16 of them. Each hole marks the beginning of a separate sector. Because the location of each sector is then physically determined by the hardware, such disks are called *hard-sectored.* Don't use hard-sectored disks with Junior. The multiplicity of holes can confuse it.

In a soft-sectored disk system, the exact arrangement of sectors and tracks—called the disk's *format*—is determined by a program called the disk operating system, or DOS (see chapter 5). The disk operating systems of different brands and models of computers specify various arrangements of tracks and sectors. The DOS that Junior uses (which is identical to that used by the rest of the IBM personal computer family), puts 512 bytes of data into each sector and arranges 40 tracks around each side of the disk. Depending on how *you* tell DOS to configure the sectors, either eight or nine of them can be put into each track.

DOS also controls how information is written and retrieved, usually one sector at a time. With such an arrangement the disk drives can randomly access individual *sectors* but cannot randomly access each single byte of data. Because the sectors are quite short, sequential data retrieval from within each sector takes just a tiny fraction of a second.

Of course, every sector on a disk can be recorded, erased, and rerecorded separately. Computers even work to make it seem that each byte in every sector can be independently recorded or changed—the computer can read a sector into RAM memory, randomly change individual bytes while they are held inside RAM, erase the original sector, and then rerecord the changed contents of the RAM memory in place of the old sector. The entire process happens in milliseconds, a lot faster than the description takes.

The rigidly organized data structure of disks might at first seem incompatible with the way people work with disk drives, arranging and rearranging programs and files, often rewriting them to make them longer or shorter. Shorter doesn't seem to be much of a problem, but how can you squeeze a longer program into a full sector when all the sectors around it are full, too? You'd think trying to squeeze more information into the same memory area would be like trying to put a quart of milk into a pint bottle.

The disk operating system is cleverer than you might think. Rather than working with an entire program or file, which is ungainly at best, the operating system divides large masses of data into smaller "blocks," each the perfect length to fit into one disk sector. The different blocks of one file or program can be recorded into sectors scattered all over the surface of the disk almost randomly, wherever there is an empty or usable sector to write in.

Once you scatter your data to the wind, it's handy to be able to put it back together again in some semblance of order. The disk operating system makes the reconstruction process easy by numbering all the parts and fitting them back together as if they made up a kit. The numbering scheme is rather simple. The operating system records the location of the first sector of any file in the special place called the *directory,* which also records the file name, date, and time it was created, and its size. Another reserved section of the disk called the *file allocation table,* or *FAT,* keeps track of which sectors are used by different files. The FAT also tells the operating system which sectors are in use and which can be used for storing data. For your added protection, the IBM operating system records two copies of the FAT on each disk just in case something happens to one of them.

Using Disk Drives

If your normal mode of operation is learn-by-doing, the disk drive seems like an elementary thing to conquer. Simply slip a disk into the drive, press down the release arm, and you're on your way to saving data on disk. Mastering the slide-in process takes about one try. And as long as you put the right end of the disk in first, right side up, you won't have a problem.

Indeed, a floppy disk, although square and flat, has a definite front and back, top and bottom. Disks with labels are particularly easy to handle because the same principle holds for them as for game cartridges—just make sure that the label is on top and the last thing to disappear when you insert the disk.

If the disk you want to use does not have a label, the first part of it to go into the machine should be the part with a lozenge-shaped slot in the cover, revealing the actual magnetic disk inside. The top is the smooth side; on the bottom, the edges are folded over and welded.

If you're using a prefabricated application program, your next step is to follow the instructions that come with the program on the disk. If you want to do your own disk manipulations, you should take a few minutes to learn the primary disk-operating-system commands.

Dealing with DOS

DOS stands for *disk operating system.* It functions as an intermediary between you and Junior's disk drive. Many inexperienced programmers approach DOS with awe because it's another system to learn, but there's nothing magical about it. DOS is simply a program that runs like any other on the Junior.

DOS also organizes the data you store on disk so that it can be quickly found and used. It works invisibly in conjunction with other programs, so you may never realize that it is there. But if you have a disk drive, you must have a copy of DOS, version 2.1.

What Do the Numbers Mean?

IBM sells a variety of operating systems, among them PC-DOS 1.1, 2.0, 2.1, and 3.0. (You may have heard IBM's personal computer operating system referred to as "MS-DOS," for its manufacturer, Microsoft. Although both MS-DOS and PC-DOS were designed by Microsoft, there are subtle differences between them.) The numbers following the name DOS indicate the revision number of the program. A higher number is used for a more recent revision. Increases of fractional parts indicate that the program has had only minor modifications. A jump of a whole number indicates a major change. In other words, DOS 2.1 is only a little different from DOS 2.0, but 2.0 is a major revision over DOS 1.1.

Junior is designed to work only with DOS 2.1. Although DOS 2.0 can operate Junior—or so it seems—IBM does not recommend that it be used. There are subtle differences in 2.1, most of which accommodate Junior. For instance, Junior's half-height (or skinny) floppy-disk drive is more finicky than the bigger drives found on other IBM computers. DOS 2.1 is designed to allow Junior's disk drive just a few fractions of a second longer to settle down before data is fetched from a disk.

The Quick Start

If all you want to do with DOS is get your disk drive working, find the disk marked "DOS" (as opposed to "Supplemental Programs") inside the back cover of the DOS 2.1 manual and shove it into Junior's drive slot. Be sure to remove the shipping cardboard from inside the disk drive first, by rotating the release arm clockwise until it is horizontal and then slipping the cardboard out. Hold the DOS disk by the label, facing up, and slide the disk in. Rotate the release arm counterclockwise until it is vertical and will travel no further. Now you're ready to turn Junior on. (If you already had Junior on, press and hold down the Ctrl and Alt keys, then press Del to "reboot" the system.)

The disk drive will make all sorts of strange noises that will soon become familiar. The clicks and grinds are caused by the read/write head snapping from track to track to find the right data.

DOS announces that it's in your machine by showing a "dummy" date, which you should correct by entering the right date. Dates are handy for keeping track of when you created files or updated them. Change the date by typing in today's date in the numerical format, month/day/year. You must separate the individual parts of the date with slash marks or else DOS won't understand and will tell you that you've entered an invalid date. DOS will give you a similar reply if you try to use names instead of numbers for months or if you make a mistake and enter 12/7/41. DOS *knows* it can't possibly be 1941 and will tell you so. Press the Enter key when you're done typing in the date. If you're in a hurry, just press Enter and the computer will pick up IBM's dummy date, Tuesday, January 1, 1980. (Later you'll learn a special DOS command that will let you change DOS's recollection of the date anytime your computer is running.)

You're not done yet! DOS next wants to know the time, which will be recorded along with the date on your files. DOS keeps time with a 24-hour clock; that is, PM is designated by the number of hours over 12

and midnight is 0. For example, 7 PM is 1900 hours. You can enter just the hour; the hour and the minute; or the hour, minute, and second. DOS will understand them all if you separate the hour from the minute and the minute from the second with colons (:), as you might expect. After you've typed the time, press Enter. (Or just press Enter and DOS will start out assuming that it is exactly midnight. As with the date, DOS has a special program that lets you reset the time almost anytime you have DOS running in Junior.)

If all has gone well, you'll see the strange legend "A>" on your monitor screen. That means you've finally reached DOS. The A> is called the DOS *prompt.* The letter *A* is the default drive letter. It indicates that the first drive in the system (or the only drive, in Junior's case) is ready to take your orders.

The first thing you should do when you encounter the DOS prompt for the first time is type DISKCOPY and press the Enter key. By doing that you have given DOS a command to make a copy of a complete disk. You should make a copy of the DOS disk so you can put the original away and not risk ruining it. Once you've had more experience with DOS, you'll learn other ways of copying disks and programs.

After you type DISKCOPY, DOS will instruct you to put the source disk in drive A:. The source disk is the original disk that you want to copy from. In this case, it is the disk that is already in the disk drive. DOS also says it wants you to "strike any key when ready." Don't believe it! DOS really wants you to strike any *character* key; the Ctrl, Shift, Del, and similar keys that don't put characters on the screen won't work.

DOS then tells you to put a "target" disk into drive A:. Remove your DOS disk and put a blank disk (or one that you won't mind erasing) into the disk drive, again with the label up. Press any character key, and DOS will begin copying the disk. You'll have to exchange disks several times during the copying process—that's the penalty for having only one disk drive. Type "N" when you're asked if you want to make another copy. When the A> prompt appears on the screen again, the copy has been made and all is well.

Once you have a copy of the DOS disk, label the copy, put it in the disk drive, and put the original DOS disk away in a safe place. Your system disk is now the copy and you should use it whenever you need a system disk or whenever DOS asks you to put COMMAND.COM in the disk drive.

Now you're ready to start using the disk drive. Slide a program disk into the slot and follow the instructions that come with it.

The Secret Life of DOS

Although DOS is a complex program with intricate and highly detailed inner workings, its function is straightforward. In effect, DOS is a translator. You give it a command by typing on your Freeboard something that looks vaguely like a word—running a program can create the same command and send it to DOS. Once you have the DOS program running, as soon as you press the Enter key (or when a running program sends DOS the electronic equivalent of pressing the Enter key), whatever you've typed is sent directly to DOS.

DOS examines the first word of the line you typed before pressing the Enter key and compares it to all the commands that it understands. If the word matches exactly with one of DOS's commands, then DOS tries to carry out that command. DISKCOPY is one of these DOS commands.

DOS is an unforgiving perfectionist. If you make a mistake and misspell a DOS command, DOS will reject it summarily and display an error message. DOS commands, therefore, must be typed exactly right (except that either capital or lowercase letters will be recognized).

DOS's translation function is not as easy as it might seem. DOS also must organize any data inside the computer, send it to the disk drive with instructions about where to put all the information, and write down where the information has been located so that each bit of data can be retrieved when you need it. In this way, DOS determines how data are stored on disks.

DOS arranges your data into *files*. A file is the electronic equivalent of a folder that can hold all sorts of information. To DOS, a file is just a series of data bits; to find that data, DOS searches for its file under a name that you've assigned. But DOS is particular about the names that it recognizes. Names must be from one to eight letters, numbers, or symbols long.

DOS also recognizes a last name, called an *extension*. Extensions must be separated from the file name proper by a period, and can be no longer than three characters. A file name, with or without an extension, can have no spaces. A proper file name would therefore be FILENAME.EXT or KUNG.FU.

Although you can generally give a file any name that you want, certain extensions are often used for special purposes (see Table 1). If you use these extensions improperly, DOS may stop working entirely.

DOS remembers file names by recording or "writing" them on disks in reserved places in an area called the *directory*. The directory lists all files that are on a disk. IBM's DOS is more exacting than most, and even

Table 1. *Common DOS File Name Extensions*

ASM	Assembly language source code for IBM MacroAssembler
BAK	Backup file for DOS's Edlin, *Wordstar,* other programs
BAS	BASIC programs
BAT	Batch files
BIN	Binary file, used to store programs exactly as they would appear in Junior's memory
CAL	Data file created by *SuperCalc* program
CHK	File created by CHKDSK in recreating damaged files
CMD	*dBase II* command file or CP/M-86 program file
COB	Source code for COBOL program
COL	*Multiplan* spreadsheet data
COM	DOS program, executed by entering its name
DAT	General program data
DBF	*dBase II* data
DEV	Device file, used by DOS 2.1's CONFIG.SYS file
DIF	Data interchange format (*VisiCalc*) spreadsheet data
DOC	Document (text) file
EWF	Document file for *EasyWriter* word processor
EXE	Program (often compiled from BASIC) executed by entering its name
FRM	*dBase II* report-form file
HEX	File of hexadecimal numbers (used by DOS's DEBUG)
HLP	Help text used by many programs
LIB	Library files used by DOS's LINK
LST	Program listing generated by IBM's MacroAssembler
MAP	List file created by DOS's LINK
MEM	*dBase II* memory file
MSG	*Multiplan* message file
NDX	*dBase II* index file
OBJ	Object-code files (created by various languages)
OVL	Program segment used as input to DOS's LINK
OVR	Overlay files (parts of program not currently in memory) used by many programs
PAS	Pascal language source program
PRG	*dBase II* program file
SYS	System file used by DOS 2.1's CONFIG.SYS
TMP	Temporary file
TXT	Text file
VC	*VisiCalc* normal file data

records the time and day you named each file and how long it has been in the directory.

The directory tells DOS what files are on a disk, but it doesn't tell DOS where the files are. DOS keeps another "notepad" on the disk called a *file allocation table* (FAT), on which it records where the files actually are kept.

DOS itself is a program that resides in disk memory and takes up three entire files. One is easy to find, called COMMAND.COM. The other two are hidden—you can't ordinarily find them but they are there. All three DOS files are necessary to get DOS started; they must be on the disk that's in the drive when you turn on, or "boot," your Junior, or in the disk drive when you stop running a program and give DOS a command.

When it boots up, Junior memorizes COMMAND.COM until you either turn your computer off or run another program that needs so much memory that it must steal the space that COMMAND.COM is stored in. (That's why when you're done running some programs Junior will ask to take another look at COMMAND.COM— the program you were running made it inactive.)

There are two kinds of DOS commands: internal and external. Internal commands are those that DOS can look up by searching through the contents of COMMAND.COM. Internal commands are easy to use. After Junior has memorized COMMAND.COM, you can tell DOS to do an internal command whenever you see the prompt, usually A>. DOS then searches Junior's memory for the command, finds it, and carries out the command. Because the meanings of the commands are stored in Junior's memory, COMMAND.COM does not have to be on the disk in Junior's disk drive for DOS to find the command. You can change disks or even use disks without DOS on them and still use internal commands.

External commands are different. Most either are too little used or require too much memory for it to be efficient to keep them in Junior's memory all the time—they would just be wasting space. Instead, the commands are stored on disk, and DOS searches the disk in the default drive (the one whose letter shows in the DOS prompt) for the meaning of the command. If the command is not on the disk in the drive, or if the drive is not ready or able to read the disk, DOS puts an error message on Junior's display indicating in a vague sort of way what is wrong. DISKCOPY is an external command. It worked because of the DISKCOPY program in your original DOS disk which was in your disk drive.

Each external command is stored in a separate file bearing a name consisting of the command and the extension COM. The file containing the program of the command must be present in your disk drive for the

command to work properly. The DOS command COPY can be used to put the individual program files that you need on any disk you want (providing there is enough room on the disk), so you can avoid the two-disk shuffle involved in copying an entire disk. You'll learn how in the section of this chapter about the COPY command.

DIR

Perhaps the most useful of all the DOS commands is DIR, which stands for directory. DIR tells DOS to type out the names of files in the directory of a disk. It will give not only the names of the files on the disk but also how much room (in bytes) each file takes up on the disk, how much room is left, and when each file was created—and even the name of the disk itself, if you've given it a name. DIR is an internal command, so once you've loaded DOS into Junior's memory, you can shuffle disks without worrying whether this command file is recorded on a particular disk. Just type DIR.

Sometimes it seems that you have to be awfully quick when you use DIR; for instance, when the list of files on a disk is longer than your display's screen. To temporarily stop the scrolling list, all you need to do is press and hold down the Ctrl key and press and release the *S* key. DOS will pause and wait for you to press Ctrl and *S* again, then finish listing the directory. It's easy to remember what Ctrl and *S* do because the *S* key on Junior's Freeboard is labeled *Sc Lk*, scroll lock, and it stops the directory from scrolling up the screen.

You can also stop the directory display from scrolling by pressing and releasing the Fn key (on the upper right-hand corner of the Freeboard), then pressing and releasing the letter *Q*. After you use that combination to stop the Freeboard from scrolling, pressing almost any key will restart the rolling display. Note that underneath *Q* is the green label *Pause*.

Sometimes you might change your mind and not want to watch the whole directory spin past. DOS lets you cancel the DIR command by pressing Ctrl and the letter *C* at the same time—think of *C* as standing for cancel. Alternately, you can press and release the Fn key, then press the letter *B,* labeled *Break* in green. (Another more complex combination that accomplishes the same thing is to hold down Ctrl, press Fn, then press *S*.)

The directory can be very handy, especially when you remember the meanings of the reserved file-name extensions. Some experienced programmers are impatient to jump right into a new program, so they use DIR to scan through the directory of the disk to find the magic word to

type to start the program. They know a .COM or .EXE listing in the directory represents a program that can be run directly, simply by typing the file name—without the extension—and pressing the Enter key.

You can get very specific with a DIR command by specifying a file name that you would like to read directory information for. Using DIR with a file name is probably the best way to find out how much of the memory available on a disk is used by a particular file. To see the directory entry for a file, type DIR, skip a space, then type the full file name—including extension.

The typed-in space is very important—it prevents DOS from thinking the file name is part of the command that you typed in. If you don't add a space after DIR, DOS will combine the file name with the command name and end up looking for a command called DIRFILENAME, for example. DOS will then advise you that you've given it a bad command or file name.

You can use other characters to separate the parts of commands, such as a comma, semicolon, equal sign, or a press of the Tab key. However, the space is usually the most convenient because it's the biggest key on the Freeboard and makes reading the commands you type much easier. The DOS manual calls these characters *delimiters*.

You can use some of DOS's special abilities to put DIR to work helping you remember the name of a file without your having to scan through the whole directory. DOS understands "ambiguous" file names; that is, you can ask DOS (and therefore DIR) to work its magic on several files with similar names. To do this, substitute special characters called *wild cards* or *global characters* for the normal letters you would type as a file name. This causes DOS to match any letter or number it understands for the wild card. The question mark (?) tells DOS that it can match any letter to that position. That is, B?LL.TXT will match BALL.TXT, BILL.TXT, and BULL.TXT (as well as BXLL.TXT and other unlikely names if they are on the disk). Use as many ?'s as you like; the more you choose, the more files DIR will display for you to choose from.

DIR knows another wild card, the asterisk (*), which matches whole blocks of text instead of only single characters. For instance, *.COM will make DIR display all the files with the COM extension on your screen. If you ask DIR to display G*.*, you'll see all files that begin with G, from GOOSE.BRY to GREASE.E. Typing DIR *.* is functionally equivalent to just typing DIR.

You must be careful with your asterisks, however, because DOS ignores the characters that follow an asterisk in your ambiguous file names and extension references—*RD.COM means the same to DOS as

*.COM, and GOFOR.*IT will act exactly as if you typed GOFOR.*. There's a good explanation for this strange behavior. DOS ignores anything over eight characters in a file name and anything over three characters in an extension. An asterisk causes DOS to ambiguously match a full eight characters (it's equivalent to typing "????????"). When the asterisk comes first, DOS sees its eight question marks and ignores all characters that follow it.

TYPE

The command TYPE is similar to DIR but instead of displaying the directory on the screen, TYPE shows the *contents* of any file in the directory. To see what's in the file DRIVEL.TXT, you would type:

```
TYPE DRIVEL.TXT
```

Because TYPE will work only on one file at a time, you cannot use ambiguous or global file names when you specify the file you'd like to see.

Using TYPE indiscriminately may cause some strange symbols to appear on the screen. Most files contain raw data rather than text. When DOS tries to type them out, it matches the data bytes with Junior's extended character set with unusual results. If a file contains ordinary text, however, TYPE will display its contents letter by letter, word by word, very quickly. To stop the fast-scrolling display, use the same keys you would to stop the directory.

PRINT

You can make printed copies of file contents by using the PRINT command. Just type PRINT followed by the name of the file you want to print. You can use ambiguous or global file names to print several files one after another. While files are being printed, you can share Junior's time and do other things, but it won't devote its full attention to you, and might miss a few keystrokes now and again.

The first time you ask DOS to PRINT after you turn Junior on, DOS will ask you the name of the "list device" you want to print to. If you have only one printer attached to Junior, just press Enter. If you have both a Compact Printer and Graphics Printer (or any combination of serial and parallel printers), specify the parallel (Graphics) printer as LPT1 and the

serial (Compact) printer as COM1 if you do not have a modem installed inside your Junior, and as COM2 if you have an internal modem.

To cancel a complete PRINT command, type PRINT /T. To cancel just one of a series of files waiting to be printed, type PRINT [file name to be canceled]/C. If you want to cancel more than one but not all of the files to be printed, just list the file names after the /C.

You can also add another file to the series waiting to be printed by typing PRINT [file name to be added]/P.

Typing in PRINT alone will make DOS list all the files waiting to be printed.

Because PRINT is an external command, a file named PRINT.COM must be on the disk in Junior's disk drive when you type the command.

FORMAT

Blank disks obviously have no programs on them. No noise. No nothing. You might think that's good—all that empty space means you have all sorts of room for data and programs. But to Junior a blank disk is as good as no disk at all. It can't save any programs on or write any data to a blank disk.

By itself, even with the aid of DOS, Junior cannot find its way around a blank disk. It doesn't know where to put anything. Therefore, a disk must be recorded with special signals that divide it into storage areas, which Junior can then make use of. These magnetic guides make up what is called the *format* of the disk. The process of recording these place markers is called *formatting* a disk. And DOS has a special command called, as you might have guessed, FORMAT. Blank disks do not have format-control signals recorded on them and are often called "unformatted" disks. If the disk drive does not find the signals it needs, it cannot write or record anything on the disk.

In the process of formatting a disk, DOS also checks to make sure that there are no problems with the disk that might interfere with the proper storage of data. A bad spot on a disk could turn a file into a hopeless jumble or prevent a program stored on it from running. DOS does a particularly good job of formatting disks. If it finds a bad spot on a disk, it doesn't merely tell you the disk is defective (as some older operating systems do), it also notes where the bad areas are and makes the rest of the disk available for holding data. The bad spots are called *bad sectors.* Unlike many computer systems, with Junior and DOS one bad sector won't make a whole disk worthless.

The disk drives of different computers use different electronic for-

mats for their disks even if the disks are physically the same. Most of the time Junior cannot use disks formatted on non-IBM-compatible computers; it just won't understand strange formats. If the disk fits into Junior's disk drive, you can use the DOS FORMAT command to make the disk work in Junior, but every time you format a disk, *all information previously stored on that disk is erased.* In other words, you cannot transfer programs or information among different brands of computers by reformatting because the programs you'd want to transfer will be destroyed in the FORMAT process. Unless you want to erase all the data you have stored on a disk, do *not* format it again once you've put programs or files on it.

Junior's disk-drive format is conveniently like those of all other IBM personal computers, as well as those of many compatible computers that use the MS-DOS operating system. But different versions of DOS make slightly different disk formats, which can occasionally cause disk-format problems.

Because Junior uses IBM DOS 2.1, the latest version, and a double-sided disk drive (which allows it to store up to 360K bytes of data on one double-sided disk), you might run into compatibility problems when trying to use Junior's disks on older, simpler machines. Not all IBM and IBM-compatible computers use double-sided disks. Although Junior has no problem finding the information recorded on single-sided disks, machines that have single-sided disk drives cannot find the data on double-sided disks. Junior can sort through disks made using any IBM operating system, but a computer running DOS 1.1 will not be able to use a disk made by Junior unless you are careful to use a special option of the DOS 2.1 FORMAT command to make it compatible.

Obviously, the first thing you'll want to do with any new disk is record the format-control signals onto it. You can also format previously used disks, even those that have been used with other machines, to recycle them for use on Junior—as long as they are of high enough quality and soft-sectored rather than hard-sectored.

The actual formatting process is easy. All you have to do is type FORMAT. DOS helps you with the rest, giving you step-by-step on-screen instructions. In the simplest case, there will be only one step—putting a fresh disk in the disk drive. Because FORMAT is an external command, a file called FORMAT.COM must be in the disk drive when you tell DOS that you want to format a disk. Obviously, you'll want to change disks before the format process begins, otherwise you'd destroy the FORMAT.COM file as well as everything else on the disk. DOS tells you to take the program disk out of the drive and insert the target disk, the one to be formatted. Then just press any key and the formatting process begins.

If you give DOS no other instructions, it will format the disk up to its maximum capacity on both sides. FORMAT also allows you to use options that reduce the capacity of the newly formatted disk but make it more compatible with other (earlier) versions of DOS. For instance, if you want to exchange disks with someone who has a computer that has only single-sided disk drives, you can tell DOS to format your disks using only one side so that they will work in your friend's computer, too. All you have to do is type FORMAT, skip a space, and type /1. The slash tells DOS the next character is a special command. The 1 of course, indicates that only one side of the disk should be used.

DOS allows you the option of recording a copy of itself onto each disk you format. By doing so, you make the newly formatted disk into a *boot disk,* one that will load the operating system into Junior when you turn it on or reset it. To tell DOS to add the operating system to a disk when you format it, add the command /S to your format instruction.

In some cases, you can combine several of the optional commands simply by adding them after the FORMAT command. Although you must skip a space after you type FORMAT, do not skip a space between options. To combine the /1 and /S options, all you need do is type FORMAT /1/S or FORMAT /S/1—the order in which you type the commands does not matter.

If you fear that labels might fall off your disks, you can give the disks a name that Junior can read electronically. IBM calls such an electronic name a *volume label.* To add a volume label to a disk when you format it, just add the command /V to the format instruction. When DOS finishes formatting the disk, it will ask you for the name you want to give the disk. Just type in that name, up to eleven characters. You can even use a space in the disk name. Consider carefully what you want to call the disk—once you name it, you cannot (easily) change or remove the volume label.

DOS 2.1 has other special provisions for compatibility with older versions of DOS. You can format a data disk to be compatible with any version of DOS by adding the command /8 after the FORMAT command. This command forces DOS to use less of the disk surface for recording data, so a single-sided disk will hold only 160K and a double-sided disk only 320K. By adding the single-sided command /1 either before or after the /8, you can make disks compatible with virtually any version of DOS and any IBM personal computer, with capacity reduced to a paltry 160K.

You can instruct DOS to save space on a disk so that a friend can take the disk you format and put his own version of DOS on it to make your newly formatted disk a boot disk for his own computer system. The command to reserve space for any disk operating system is /B. Because

space is reserved on the disk for the operating system, capacity is reduced from what is possible with the /8 compatibility option—313K double sided and 151K single sided.

If you have a disk that has space reserved for the operating system (many non-IBM commercial programs are sold that way because selling a copy of IBM's DOS would be a copyright law violation), you can add the special DOS files using the external command SYS. The file SYS.COM, as well as DOS, must be on the disk in the default drive to use this command. Just type SYS and do the two-disk shuffle when the screen asks you to. See Table 2 for a recap of the format commands.

Table 2. *Junior Format-Command Summary*

FORMAT—by itself, for maximum capacity (360K), double-sided data disks.

FORMAT /1—for single-sided disks, with full capacity (180K).

FORMAT /8—for double-sided disks compatible with any version of DOS, but with reduced capacity (320K). Volume labels not permitted.

FORMAT /8/1—for single-sided disks compatible with any version of DOS, but with greatly reduced capacity (160K). Volume labels not permitted.

FORMAT /B—formats double-sided disks with capacity reduced to allow compatibility with all versions of DOS and to allow any version of DOS to be recorded on disk. Effective disk capacity is 313K.

FORMAT /B/1—formats single-sided disks with capacity reduced to allow compatibility with all versions of DOS and to allow any version of DOS to be recorded on disk. Effective disk capacity is 151K.

/S—to put operating system (DOS 2.1) on disk to make it a *boot disk*. Reduces disk capacity by about 40K. Cannot be used in conjunction with /8 or /B options.

/V—to put volume label (name) on disk. Takes up no space, but cannot be used with /8 or /B options.

COPY

A thousand misfortunes can happen to any disk, from being crushed under the coaster of an office chair to a careless scribble across the label with a ballpoint pen. Even the slightest damage can make a disk unusable.

Because disks are easily damaged, it's a good idea to keep a backup copy of all the data and programs stored on them. The best place to put a spare or "backup" file is on a separate disk that can be kept in a safe place away from the original.

DOS has a special internal function called COPY that allows you to duplicate any file on a disk—either to put more than one copy of the same file on the same disk (but with different names to avoid confusion) or to put duplicates on different disks.

The simplest copy procedure is to make a duplicate of a program or file on the same disk. Just type:

```
COPY [name of the original file] [name of copy]
```

Using real names, the command to copy GARBAGE.DOC to BACKUP.DOC would simply be:

```
COPY GARBAGE.DOC BACKUP.DOC
```

The command is easy to remember if you translate DOS's shorthand to its English equivalent, "Make a COPY of the file called GARBAGE.DOC and call the copy BACKUP.DOC."

Backing up on the same disk offers some protection, but odds are that if you accidentally destroy one file on a disk, other files on that disk will also be affected.

You are protected better when you copy files from one disk to another. DOS makes the procedure especially easy with a feature called *virtual disk* that makes Junior think it has two disk drives instead of only one. DOS puts messages on Junior's screen telling you that it has changed the name of the disk drive from "A:" to "B:" (or vice versa), and you insert the proper disk for that disk drive. DOS keeps track of which is which, and if you make a mistake, it will warn you without wiping out any of your files.

To use the virtual drive, specify drive names in your COPY command. If you don't specify a disk-drive name for the file you want to copy to or from, DOS will assume that the file in question is on the default disk drive, that is, the one whose letter shows in the prompt of the line you're typing the COPY command. If the screen shows the prompt line A>, then A: is the default disk drive. If the prompt is B>, then B: is the default disk drive.

One command that will copy GARBAGE.DOC on disk B: while keeping the original intact in drive A: is:

```
COPY A:GARBAGE.DOC B:GARBAGE.DOC
```

If A: is the default disk drive, you don't have to put the A: in front of the file that's on the disk in that drive. In addition, if you fail to specify a different name when copying to another disk, DOS will still copy the file, but will assign the new copy that same name as the original. The following command, therefore, is identical to the previous example.

```
COPY GARBAGE.DOC B:
```

Just remember the English equivalent of this command would be, "Make a COPY of the file GARBAGE.DOC on the default disk drive onto the disk in drive B: with the same name as the original."

Because DOS automatically assumes that you want to use the default disk drive when you don't specify a drive letter, copying to the default disk takes an even simpler command:

```
COPY B:GARBAGE.DOC
```

From that simple instruction, DOS will know that you want the file called B:GARBAGE.DOC copied on the default disk drive—providing it is not drive B:. If B: is the default disk drive, you will get an error message, because the command essentially asks DOS to give the copy the same name as a file that is already on the disk.

You can copy more than one file at a time by using ambiguous file names or wild cards in the COPY command. For instance, to copy all of the files on the default drive, A:, to the virtual disk drive, B:, you can type:

```
COPY *.* B:
```

As DOS progresses along, it will name each file as it is copied. If you run out of room on the disk that you are copying to, DOS will stop and tell you so by noting "insufficient disk space." All the files it copied and listed on the screen until it ran out of space will be okay, and no damage will be done. But DOS will abandon the command; even if you insert a new disk, DOS won't pick up where it left off. You'll have to either copy all the files over or give DOS more specific names of the rest of the files to be copied.

DOS's COPY function has one feature that can be good or bad, depending on whether it catches you by surprise. If you tell DOS to copy a file onto a disk on which a different file with the same name already

exists, DOS will obediently blast the old file to oblivion and replace it with the new one, never warning you that the old file will be forever lost. The feature is good because you can update files with new versions with little fuss and bother and a minimum of commands. But it might backfire if you're forgetful and unimaginative and occasionally use the same file name over and over again. The moral is to use imagination in your file names, don't reuse them except purposefully, and always keep a backup copy, just in case you forget the rest of the advice.

The COPY command can do much more than merely copy files. To send them to your printer or modem, just use a file name and an appropriate destination name. For instance, if your Junior system has only one printer, DOS will call it PRN. You can print out a copy of a file by telling DOS:

```
COPY BANANA.TXT PRN
```

As we've seen, if your Junior system includes more than one printer or if you want to be more pointed in your printer commands, you can address each printer with more specific names. For instance, if you have an IBM Graphics Printer attached to your Junior through an IBM printer adapter, DOS will call the printer LPT1. If you installed a modem in Junior, a serial printer (like the IBM Compact Printer) is called COM2. If you don't have an internal modem, DOS calls a serial printer COM1. Unlike with the PRINT command, when you COPY to the printer, Junior will be completely occupied with the print job, so you must wait to do anything else.

The combination Freeboard/monitor is a device DOS calls CON. By copying from CON to a file, you can put what you type on the Freeboard into a file. (To properly close the file and end the copy process, after the last Enter, type in the new file name, hold the Ctrl key, and press *Z*, or press Fn then 6. Then press Enter and DOS will respond with "1 File(s) copied.") You can also use COPY like TYPE by copying from a file to CON.

This example will put the numbers 0 through 9 in a disk file called STORAGE.BIN. Type:

```
COPY CON STORAGE.BIN (press Enter)
0 1 2 3 4 5 6 7 8 9 (press Enter)
```

Then press Ctrl-Z. Your computer will respond:

```
1 File(s) copied
```

To verify that the file has been created, enter this line:

```
TYPE STORAGE.BIN
```

And Junior will read your newly created file back to you.

Sometimes two is not better than one. Perhaps while concentrating on solving all the world's problems through computing, you built two files that by rights should be combined together into one big file. Using DOS's COPY function, you can make one file out of two or more shorter ones (but still preserve the shorter files to do with as you please).

Programmers often use the term *concatenate* to mean the process of combining files. The word *combine* should do you just fine. To combine files, just use COPY as you would to duplicate a file, but string together the old files that you wish to combine by putting plus signs between them. A command to combine three files into a new file called "MOE.DOC" would be typed like this:

```
COPY EENY.DOC+MEANY.DOC+MINEY.DOC MOE.DOC
```

If you don't specify a file name for DOS to concatenate to, it will add all the files to the one listed first, changing it while leaving the rest of the list untouched.

If you're a little worried about the reliability of these computer functions and you don't trust Junior to make a perfect copy every time, you can make DOS check every bit of the new file as it is being made. Just add the command /V (for verify) after your copy instruction, like this:

```
COPY BUSY.BEE B:/V
```

Another way to be sure that the copy process was accurately completed is to use another DOS command called COMP, short for COMPare. The command makes DOS compare two files to see if they are exactly identical. To use it, type COMP followed by the names of the two files that you want to compare, like this:

```
COMP A:ORIGINAL.FIL B:DUPLICAT.FIL
```

You can use ambiguous file names with COMP, and DOS will do its best to match them. For instance, if you type:

```
COMP *.ORG *.CPY
```

DOS will compare all files with the same name before the period and the extensions .ORG and .CPY. You can use this command to compare files on the same or different disks or anywhere DOS can get at them. Because COMP is an external command, however, you must be sure to have a copy of the file COMP.COM on the default disk.

As we've seen earlier, if you want to make an exact copy of a complete disk, DOS has an external command called DISKCOPY designed to do exactly that. Because it is an external command, a copy of the file DISKCOPY.COM must be present on the default disk for the command to work. It will copy a complete disk onto another, and even format the disk it is copying to as it goes along (if formatting is necessary).

If your Junior is equipped with the standard one disk drive, just type DISKCOPY, and DOS will tell you every step to take along the way. If the disk you want to copy is crammed full of files, you'll play the two-disk shuffle for a while—DOS will read all the data that will fit into Junior's available memory off the original (or source) disk, then instruct you to shuffle disks, dumping the contents of the memory onto the new disk (or target). Once that's done, DOS asks you to shuffle again and reads another memoryful of data. The process repeats until everything has been copied.

For your peace of mind DOS has a DISKCOMP function that corresponds to the COMP command. To compare disks on a single disk drive Junior, type DISKCOMP and follow the on-screen instructions.

ERASE: The Road to Oblivion

After you've made a thousand copies of a file, you may discover that you've overdone things and decide to do some pruning, a bit of weeding, or maybe even plow everything on the disk under and start fresh. What you need is a handy way to eliminate the files that you no longer need on a disk. DOS has an internal command designed for that purpose, ERASE. It's probably one of the easiest commands to use, second only to DIR. Type in ERASE, insert a space, and type the name of the file that you want to eradicate. For instance, to eliminate the text of a promotional piece that you no longer need to save, you might type:

```
ERASE SNOW.JOB
```

ERASE accepts ambiguous file names, so you can obliterate a whole range of JOBs merely by typing in:

```
ERASE *.JOB
```

You can even erase all the files on a disk by typing:

```
ERASE *.*
```

DOS knows that few of us are infallible and that sometimes you regret rashly erasing a whole disk. DOS could prevent such mistakes by backing up each file before it erases the whole disk, but that would defeat the purpose of erasing the disk. But DOS will give you a second chance to reconsider by asking if you really want to blow away what may be a lot of hard work. It will laconically ask, "ARE YOU SURE? (Y/N?)". If you respond with *Y* and a press of the Enter key, all the files on the disk are turned into free space. If you reply with any other character, DOS will ignore the previous ERASE command.

DOS also understands an abbreviation for ERASE, named DEL, which is a carry-over from an older operating system that called its erase command DELETE.

You can also use the virtual drive feature of DOS to erase a file from a disk not currently in the disk drive. If A: is the default disk drive and you type:

```
DEL B:FIRST.ACT
```

DOS will ask you to put the disk you want to erase from into the disk drive. When you've done that and are ready to send FIRST.ACT to never-never land, just strike any key on the Freeboard. Then switch disks back again.

Altering File Names with RENAME

It happens to us all. You had a bad night at the computer. Your every keystroke inevitably led to an error message. Your word processor stopped processing and your database got caught by the shortstop between second and third in a double play. You were so frustrated you saved your work in a file that you named with some Anglo-Saxon oath so vile it etched itself into your monitor screen. Now that it's the morning after, you've recollected your thoughts and composure, and that angry legend is still emblazoned in your disk directory. Quick! You want to change it before anyone sees it. What to do?

You could copy the file under a different name and erase the original. But you're in a hurry. Poise your hands on the Freeboard and get ready for the one-step alternative—RENAME. Just type:

```
RENAME [old file name] [new file name]
```

Press Enter, and DOS will whisk away your transgression against society. If you're really pressed for time, you can abbreviate RENAME as merely REN and DOS will understand completely.

If you want, you can rename many already-written programs with your own labels so when you want to run a program, you call it what you want—change the file name but not the extension. It must remain .COM or .EXE to tell DOS that the file is a program. (Occasionally this strategy won't work, particularly when a program requires several different files that try to access each other. Although you know the new name you've given a program, that secret is not shared with the other files in the program. Change the name of COMMAND.COM and DOS will never be able to find it.)

RENAME also recognizes ambiguous file names, so you can change a whole range of files with one simple command. To change all of your files that have the .DOC extension to an extension of .TXT, you can type the following simple command:

```
REN *.DOC *.TXT
```

DOS will not let you use RENAME to give a file the same name as one that already exists on the disk. If you try, DOS will warn you, "duplicate file name or file not found error." If you run into that message, just try over again with a different name.

New MODEs

Junior is designed to operate in a certain way as soon as you unpack it. It displays characters across the monitor screen 40 letters to a line. It can show you all the colors in the rainbow. It will assume that you'll use a serial printer unless you added a printer adapter—then it will use only the printer attached to the printer adapter. It sends signals to the serial printer at a predetermined speed. Fortunately, DOS has the ability to make Junior do what you want. All it takes is a special external DOS command called MODE.

The MODE command actually handles three entirely different functions. It tells Junior how you want characters displayed on your monitor. It tells Junior which port will handle the printer. And it tells Junior at what speed to operate the serial port (as well as instructing it in the digital language to use for the serial-port communications).

DOS gives you your choice of four ways of displaying characters on the monitor screen: 40 or 80 columns wide in color or monochrome. When you first turn Junior on, it is set up to display 40-column text in color because that arrangement makes a readable display whether you use a cheap television set or an expensive RGB monitor. Using MODE, you can customize Junior's signals to match whatever kind of display you have. For instance, if you have bought a composite monochrome computer display or an RGB monitor, you'll probably want to change its display to 80 characters, as these monitors support the wider measure.

The MODE command uses several abbreviations to indicate your display choice. CO means color; BW means black-and-white; 40 requests a 40-column display; and 80 requests an 80-column display. Using MODE, you can set the color and width separately or together. If you combine them, the color command must go first and you *do not* add a delimiter between the color and width command. Here are some examples:

To make a display 80 columns wide: `MODE 80`

To change the display to color: `MODE CO`

For a monochrome display 40 columns wide: `MODE BW40`

For a color display 80 columns wide: `MODE CO80`

Not all of these modes make readable displays with all monitors. Try various combinations and see what works best for you.

MODE also changes the communications parameters of Junior's serial ports. If you use standard IBM serial accessories—the internal modem or Compact Printer—you'll probably never need to change the parameters that Junior assumes are correct. The only time you might want to change these parameters is if you decide to do your own programming for an external modem or if you want to use a non-IBM serial printer with Junior. In most cases, it's easier (and more permanent) to set the printer to match Junior's assumptions than to change Junior's mode every time you want to use the printer. But if you do need to change the communications parameters of your serial port, see the MODE command in the reference section of your DOS manual.

If you splurge and buy both an IBM Compact Printer and an IBM Graphics Printer (or a similar printer from another manufacturer), you can use the MODE command to redirect signals from the parallel printer port to the serial port. For instance, even if you have two printers, most

programs will send their output to the parallel printer. To send printer data to a serial printer connected to Junior's serial port *if you do not have an internal modem*, use this command:

```
MODE LPT1:=COM1
```

If you have *both* a serial printer and an internal modem, use this command:

```
MODE LPT1:=COM2
```

Hidden Power in Batch Commands

After you've used Junior for a while, you may find yourself going through the same steps each time you turn the power switch on. You might use the MODE command to set the width of your screen, or BASIC to change the color of the screen, and you'll probably want to jump automatically into a specific program. You can teach Junior to automatically follow a standard procedure each time you turn it on or reboot.

When DOS 2.1 starts running, but before it gives you control by showing you the A> prompt, it checks the disk in the A: drive of your computer (probably Junior's only drive) for two very special files, CONFIG.SYS and AUTOEXEC.BAT, always in that order.

CONFIG.SYS automatically adds outside devices to your computer system. The data in the CONFIG.SYS file tells Junior how to assign these devices in the system and how to use them. The workings of this process are best left to advanced programmers.

AUTOEXEC.BAT, on the other hand, is easy enough for anyone to use. DOS reads that file as if you had typed its contents on the Freeboard. That means that if you write a command and save it in the AUTOEXEC.BAT file, DOS reads that first, even before giving you control, and it carries out the commands in the file every time you turn the computer on. For instance, if you filled your AUTOEXEC.BAT file with the commands:

```
MODE C080
BASIC INVNTRY
```

DOS would automatically set Junior's display to color, 80-column mode, then start the BASIC language, load the program in the file "INVNTRY.BAS" (probably a program that handles a business inventory), and run the program—and you wouldn't have to type a thing!

Here's a handy AUTOEXEC.BAT file and BASIC program that in

combination will automatically set an on-screen color combination every time you turn Junior on:

AUTOEXEC.BAT File

```
BASIC SETUP
```

BASIC Program

```
10 COLOR 15,1,1
20 CLS
30 SYSTEM
SAVE "SETUP"
```

With these instructions, Junior will read the AUTOEXEC.BAT file, start BASIC, then load and run the SETUP program. Line 10 tells BASIC to set Junior's display colors to bright white characters (15) with a blue background (1) and blue border (1); substitute other numbers for your favorite colors (see chapter 6 for a color chart and a program that will automatically set up these files for you). CLS in line 20 is a Clear Screen command, and if you don't include it, odds are your display's colors will revert to the normal default of dim white against black when you leave BASIC. Line 30 tells BASIC to stop running and return control to DOS. If you want, you could add a MODE command before the BASIC command in the AUTOEXEC.BAT file or even include the name of the first program you want to run.

The AUTOEXEC.BAT file overrides DOS's compulsion to ask you the date and time. If you want DOS to wait for you to enter the proper date and time before running further commands, you must use the commands DATE and TIME (see later) in the AUTOEXEC.BAT file.

With the following command, you can easily make an AUTOEXEC.BAT file by telling DOS to copy what you type on Freeboard to the AUTOEXEC.BAT file:

```
COPY CON: AUTOEXEC.BAT
```

Press Enter and the cursor will position itself directly under the command. Type in the commands you want in your AUTOEXEC.BAT file, one complete command to a line. End each line and advance to the next by pressing the Enter key. If you make a mistake on a line, you can use either the Backspace key or the ← cursor-control key (the → cursor-control key will cause the character directly above the cursor on the previous line to be repeated!). If you get so bogged down in mistakes that you want to give up, either press Fn then *B*, or hold Ctrl down while pressing *C* (for cancel). To tell DOS you're finished creating your new file, either press Fn

then the number 6 or hold Ctrl down while pressing *Z* (either one will put a ^Z on your display) and then Enter. If all goes well, DOS will tell you that it has copied one file.

The AUTOEXEC.BAT file is just one of a special class called *batch files.* All batch files are collections of commands that DOS executes in sequence. Batch files are named like other files except that they have the special extension .BAT. Typing the name of a batch file (without the extension) and then pressing Enter causes DOS to start executing the sequence of commands inside the file. For instance, if you type AUTOEXEC then press Enter, DOS will start executing your AUTOEXEC.BAT file as if you had just turned Junior on.

DOS has special subcommands that you can put in batch files to give them so much versatility that they can be used almost like a programming language.

The ECHO subcommand tells DOS either to show you what it is doing as it progresses through the sequence of commands in the batch file (ECHO ON) or to keep its inner workings secret as it goes along (ECHO OFF).

The REM (remark) subcommand causes DOS essentially to ignore the rest of that command line so that you can add remarks (which print out on the monitor screen) to your batch files to help you understand why you did the things you did when your memory gets foggy. The remarks, up to 123 characters per command line, are displayed on the monitor if ECHO is turned on. Use only the REM abbreviation as using the full word "REMARK" will cause a "bad command or file name" error message even though the batch file will continue executing.

You can also add remarks that never print out by preceding the first word in the line of remarks with a colon (and not skipping a space before the first word of the remark). The following line will never print on the display screen and will have no effect on batch-file execution (unless its first word is used as a label for a GOTO command, discussed later):

```
:THIS IS A REMARK THAT WILL NEVER PRINT
```

PAUSE in a batch file causes DOS to pause, put the message "Strike any key when ready" on the display, and wait for you to strike any character key. PAUSE also acts like remark when you add a prompt line (up to 121 characters) after the word PAUSE on the same command line. If DOS is in the ECHO ON mode, the prompt will be displayed while the execution of the commands in the batch file is paused. If you press Fn then *B* (for Break) or hold Ctrl and press *C* during a pause in a batch file,

execution of the commands in the file will stop and you will return to the A> prompt.

Batch files can be interactive. For instance, you can make a generic batch file that functions like a form letter; you leave blanks in the commands and then tell DOS how to fill in those blanks when the file runs. You can use the same batch file with different information to fill the blanks each time the batch file executes. Here's how to do it. When you make a batch file, indicate the blanks to be filled in with the symbols %1 through %9, which stand for *input variables.* After you type in the name of the batch file but before you press Enter to tell DOS to start executing the file, type in the information you want inserted in the blanks, skipping a space between the information for each blank. The blank symbolized by %1 will be filled with the first word after the batch file name, the %2 blank by the second word, and so on. (If you label a blank with %0, it will be filled with the name of the batch file!)

Here's an example. Apple II computers have a command in their operating systems called CATALOG that is similar to DOS's DIR. You can make your Junior behave like an Apple when you type in CATALOG by creating a batch file called CATALOG.BAT, containing the rather trivial instruction DIR %1. After you create that batch file, it will read the directory entry of a file whenever you type CATALOG [file name]. If you type CATALOG *.COM, the batch file will read the directory entries for all files on the disk in the default drive that have the extension .COM. And if you merely type CATALOG, the batch file will read the entire directory—exactly what you'd expect from an Apple.

In the unlikely case that you need more than nine variable inputs to a batch file, you can use the SHIFT command, which is discussed in your DOS reference manual, to multiply the number of variables available to you.

The GOTO command makes batch file processing jump out of the normal line-by-line order of command execution either ahead or behind. The GOTO command must be followed by a "label," which corresponds to the name of the line preceding the line you want batch file processing to jump to. The name of the line must be preceded by a colon so that DOS doesn't think the name is just another command. For instance, when DOS encounters the line GOTO FRANK, it will immediately look for a line labeled :FRANK in the batch file, and upon finding it, start executing the command on the *next* line of the batch file. If DOS cannot find :FRANK, it will display the error message "label not found" and *stop executing* the batch file.

Batch files can even make decisions based on existing conditions, just

as programming languages do. The IF command makes the batch file evaluate relationships among data. Currently IF can usefully evaluate two conditions—the existence (or nonexistence) of certain files and whether two strings of characters match. Should the condition being evaluated be true, IF executes a DOS command following and on the same line as the IF command. If the condition is false, IF skips the command and goes on to the next line of the batch file.

To carry out a DOS command if a file exists, use this line in your batch file:

```
IF EXIST [file name] [DOS command]
```

Use this command line to carry out a DOS command in a batch file if a file does not exist:

```
IF NOT EXIST [file name] [DOS command]
```

To carry out a DOS command only when two strings match, use this command in your batch file:

```
IF [character string 1]==[string 2] [DOS command]
```

To carry out a DOS command only when two strings do not match, use this command in your batch file:

```
IF NOT [string 1]==[string 2] [DOS command]
```

As an example of using IF with an interactive batch file, you can use IF to add a rudimentary password system to your batch commands like this:

```
:LOOP
IF NOT %1==PASSWORD GOTO LOOP
```

If you don't type in the right password after the name of the batch file when you start to execute it, the file will get caught in a loop as it is being processed. By adding ECHO OFF, you can prevent anyone from seeing why the batch file won't work properly.

Other DOS Commands

CLS—Clear Screen—is a DOS command much like the BASIC command of the same name. It comes in handy when your display screen gets cluttered or if you happen to set the display colors to an unreadable combination. CLS erases everything from the display screen and resets the colors to DOS's default combination of dim white against a black background and border. When CLS is done, all you will see on the screen is the A> prompt. An internal command, you can use it whenever you see the A> prompt (or when you know that it should be there but is invisible because of a poor color choice).

CHKDSK stands for Check Disk. It's an external command that sends DOS hunting around in the memory of Junior and the default disk drive. CHKDSK will tell you how much memory is installed in your system and on the disk you are using, and how much is available. Further, it hunts for sectors on the disk on which the data may have become damaged. If you specify the option /F after your CHKDSK command, DOS will gather together all the damaged sectors it finds and put them in special files. You might be able to use them to reconstruct lost information if you can figure out where the data came from. CHKDSK is an external command, so the file CHKDSK.COM must be on the disk in the drive for the program to run.

DATE is an external command that tells DOS to display the current date (or whatever date you've given it). You can correct the date if necessary. Then press Enter.

TIME is an external command that works like DATE but shows and updates the current time.

DOS has many other commands, but most of them are of primary interest to people with hard-disk drives in their computer systems. Several commands allow the manipulation of additional directories on disks so that each directory functions almost like a subdisk and lets you transfer files to and from different directories just as if they were disks. Other DOS commands are designed to make backing up large files from hard-disk memory to floppy-disk memory easy. You will probably never need to use them with Junior.

Stroke Savers

One of these days you're going to make a mistake when you type a lengthy DOS command on the Freeboard. Or, if you're immune to making

mistakes, you'll find yourself issuing a dozen commands that are all the same except for one bothersome letter or number (like COPY LETTER1.DOC B:, COPY LETTER2.DOC B:, and so on) and waste half a morning keystroking in roughly the same command over and over again. Programmers often face these very same problems, but the difference is that they can do something about them, and IBM's engineers have. Tired of typing, they created some special DOS keystrokes that may save you a bit of time when you start doing heavy DOS work, toting bales, lifting barges, and moving files around.

Most people don't know it, but DOS remembers every command you give it—until you type in another command. In itself, that short-term memory might not seem to be very significant, but that one-command memory can be useful because DOS also lets you call the memorized command back to the screen and use it over again or *modify* it into a different command, which you can then issue back to DOS.

The last memorized command that you give DOS is called the *template* because it can function as a drawing guide or mold for the next command.

You can call the template back to the screen by pressing Fn then 3 on the Freeboard. Try it. Type some nonsense like "Hello there, Mr. Dos" at the A> prompt and press Enter. After DOS tells you "bad command or file name," press Fn then 3. Like magic, your nonsense should reappear on the screen, all ready to irritate Mr. Dos one more time.

Actually, Fn-3 works a little differently than just repeating the last line you typed. It repeats only the characters from the current cursor position (the number of spaces from the left margin of the screen) to the end of the previous line. In other words, you can type in the first few letters, press Fn-3, and DOS will finish the line for you.

You can demonstrate how Fn-3 works by using your previous nonsense ("Hello there, Mr. Dos") as the template. On the next line of the screen, type XXXXX, then press Fn and finally 3. You should see this:

```
XXXXX there, Mr. Dos
```

With those few keystrokes you've done something that most people never figure out how to do—you've edited a DOS command.

Fn-3 by itself isn't that useful, but combined with the other DOS editing keys, you can get down to some serious keystroke saving.

Fn-1 repeats the template command one letter at a time; every time you press Fn then 1, one letter of your previous command will be retyped. Although in the context of Junior the two keys for one trade-off seems somewhat inefficient, at least those two keys always stay in the same place

so you don't have to hunt and peck for them. (On other IBM computers, however, the Fn-1 combination is achieved with the press of a single key, and the command becomes somewhat more desirable.)

IBM's engineers probably thought that having to type two keys to save time in getting one on the screen was really pretty stupid, too, so they invented an alternative to the Fn-1 combination—the right arrow (→) key on the cursor pad. Pressing it once gives the same effect *when you are working directly in DOS* as the two-key combination of Fn-1.

Fn-2 is even handier than Fn-1. It copies the template up to a certain character, which you specify by typing it immediately after you press the 2. DOS then repeats the template command up to but not including the *first* occurrence of the character you specify.

Using Fn-2 and Fn-3 in combination is a handy keystroke saver when you are copying several similar files, like LETTER1.DOC through LETTER8.DOC. After DOS completes the execution of the first command, just type Fn, then 2, then the number 1 of the first command. DOS will automatically copy the entire command up to but not including the number 1. Now type the character you want, say a 2, to change the template into a command to copy the next file. The 2 will then be typed *over* the 1 in DOS's memory. After you've typed the revised number, press Fn then 3 to finish the line. You'll have edited and copied the whole command line in just five keystrokes and with a minimum of hunting and pecking.

You are likely to run into one problem, however. You've probably noticed that overtyping the template requires that your new command be exactly the same length as the old one, unless you want to add extra characters at the end of the old command. Often, however, the number of keystrokes in your new command is not the same as in the old template. Fortunately, DOS will let you add or subtract keys from the template using the Freeboard's Ins and Del keys.

Pressing Ins acts like a wedge. It squeezes all the keys you type after pressing it into the template, pushing all the later characters further down the line. To stop inserting characters and revert back to the normal overtyping mode, press Ins again.

Ins is very handy when you get carried away with what you're doing and type the file name of something you want to copy but forget the word "copy" itself. After DOS chastizes you with an error message, just press the Ins key, type the forgotten command and a space, press Ins again, and finish the command line by pressing Fn followed by 3. You'll probably find that Ins can save a lot of typing.

Del is trickier to use because it erases characters from the template

before you get to see them. In effect, it pulls characters from the middle of the template and lets the two pieces that are left (before and after the deletion) snap back together again. Del is most useful when you type Fn-2 to get to the letter you want to eliminate, press Del, then press Fn-3.

Fn-4 works just the opposite of Fn-2. It skips over all characters until it reaches the one you specify by typing it immediately after you type the 4. If you just type Fn-4 and a character, however, you won't see anything on the screen—remember, Fn-4 just *skips* characters from within the template, it doesn't show you anything. To see the effect of using Fn-4, you'll have to use one of the other editing functions, like Fn-3, after typing it.

Fn-5 plays the old switch with Junior's template memory. It takes all the commands on the line you've just typed and puts them into the DOS template memory *without sending them to DOS to act upon.* Although your line will be committed to DOS's template memory, you won't get a "bad command or file name" message when the flight of your fingers leads to gibberish. Remember not to press Enter before Fn-5. If you do, you'll just be sending a blank line to the template memory!

The Esc key cancels the line that you are on without changing the contents of DOS's template memory. In other words, if your editing goes awry, press Esc and try again. Obviously, the DOS engineers thought they needed a second chance when editing commands, and you're likely to as well. Knowing all the DOS editing functions and putting them to use can save you a good deal of time and frustration.

CHAPTER SIX

BASIC's Basics

BASIC is PC*jr*'s first language. Junior is born with the ability to understand and use it. And at that, it has an advantage over you. Although you've probably spent your entire life learning one language, English, you now must go back and learn another one, a computer language, so that you can communicate with Junior.

Computer languages are special programs that convert words and symbols from a form that people understand into one that a computer can understand. The translation is necessary because Junior only understands digital data bits, tiny pulses of electricity in its circuitry.

The special patterns of digital pulses that tell Junior what to do are called *machine language*. Each pattern is activated by a code number, which is a representation of machine language most familiar to programmers. Machine language is complex and very difficult to learn. It is called a *low-level language*.

BASIC is a *high-level language* that translates computer programs written in familiar English words, symbols, and numbers into the low-level machine language that computers understand. Developed in the early 1960s by John Kemeny and Thomas Kurtz at Dartmouth College, BASIC, which stands for Beginner's All-purpose Symbolic Instruction Code, has become the most common of all microcomputer languages. It's a "beginner's" language not because it is particularly easy to learn but because it was the only language that would work on the early personal computers used by computer hobbyists. Beginners had little choice but to

learn BASIC. It did not require powerful computing power or much memory—some versions of BASIC will operate fine in computers with as little as 4 kilobytes of working memory. Because computer integrated circuits had not yet been refined, memory was expensive when these first machines were made, and hobby computers with small memories were the only affordable machines around. These hobbyists grew up with BASIC and went on to become the reigning monarchs in today's personal computer industry, so they've brought BASIC with them.

BASIC's ability to fit into small places has made it Junior's first language. The program that makes BASIC work is so small that it can easily fit the confines of read only memory. When you first turn Junior on, this compact version of BASIC, called *cassette BASIC* by IBM because its primary mass-storage medium is cassette tape, is the default means of commanding Junior. (A default is a setting chosen by the manufacturer as the one most users will select.)

As both the use of BASIC and the power of the average personal computer have grown, the language has been expanded to make it more powerful. But the increased power requires more storage than is available in Junior's ROM chips. So that you can take advantage of the increased power of the extended versions of BASIC, IBM sells *cartridge BASIC* as a cartridge-based add-on software package for Junior. IBM's cartridge BASIC is, in fact, one of the most extensive—that means powerful—implementations of the BASIC language available today.

Cartridge BASIC actually has a few *more* functions than the BASIC supplied for the bigger IBM Personal Computer. Because of differences in the ROM chips between the IBM PC and Junior, the PC's BASIC (alternately called "Advanced BASIC" or "BASICA"), which comes recorded on the DOS 2.1 or 3.0 distribution disks, will not work in Junior. You must buy IBM's cartridge BASIC to enhance Junior's built-in version of the language.

IBM's BASIC is also available in another version called *compiled BASIC.* One of the greatest strengths of the cassette, cartridge, and advanced versions of BASIC, and greatest weaknesses, is that all three are *interpreted* languages. This means that the computer must translate each BASIC program line into computer language while the program is running. The program is therefore ready to run as soon as you write it; in fact, you can run pieces of a program to test them before you finish the whole program. The alternative is the compiled language, which works much differently. In this version, after the program is written, another program called a compiler translates the program into machine-level commands in one big chunk. The resultant program code, called the *compiled program,* is then run. Because it has been translated directly into the computer's

language, it runs much, much faster (possibly ten to twenty times faster) than the interpreted version. But the compiled program is not easily changed. IBM BASIC gives the best of both worlds—programs can be written and honed to perfection in easily changed and tested interpreted cartridge or advanced BASIC, and once the program works perfectly the same program can be compiled into a quick-running speed demon.

This chapter will introduce you to some elementary concepts of BASIC programming. If you try the examples on your Junior as you read through the book, you'll become familiar with the basic BASIC commands and learn the rudiments of organizing and putting together a program.

Besides providing one of the best versions of BASIC available, IBM also gives some of the best lessons in learning the language in the book that comes with Junior, *Hands-On BASIC,* and in the reference manual that accompanies Junior's cartridge BASIC. Reading them will give you an excellent grasp of the language.

However, if you want to get Junior running sooner than that, this quick introduction to BASIC will help you learn several important functions. You'll see that BASIC programming is relatively easy and can actually be fun. It is not all the BASIC you'll ever need to know, but it is enough to get your computer computing.

Beginning with BASIC

Computers are different from other machines that you've faced. They are so versatile and hold so much potential, but they must wait for you to give them the proper instructions. Often, this is in the form of a BASIC program that you purchase or write yourself. Let's look at how you get started in BASIC.

You've got the Freeboard on your lap and you want the computer to do something. The obvious thing to ask it to do is compute. Try typing that in (don't worry, nothing you can do with the keyboard can hurt Junior). Type the letters C-O-M-P-U-T-E. Something's happened! The word "COMPUTE" has appeared on the screen. Good! You've actually got Junior to recognize each key that you pressed down. (If the letters don't appear on the screen and Junior beeps at you, it can't see the keyboard very well. Try pointing it more toward the computer.)

There's only one problem with the letters on the screen. The computer hasn't done anything at all, much less compute. But there's a good reason for Junior's reluctance to spring into action—you haven't told it that you're done typing.

To signal Junior that you're finished typing in your command, press

the Enter key. Junior won't react to your typing until you press Enter, but that's actually helpful because if you make a mistake you can correct it before sending it to the computer. For instance, if you make a mistake and type "COMPUTW," you can go back over the last character by pressing the Backspace key. Then press Enter.

What happened? You saw a very strange message, didn't you? "Syntax error." And there it is, "OK," just below the error message. That's fine, but why didn't the machine compute like you told it to?

The OK is a good sign, really. It's just Junior's way of telling you that it's OK to run a program. But you have not yet given it anything to compute—the machine has no program to run.

Actually, Junior *did* do some rather trivial computing. After you typed in "COMPUTE," the BASIC language program in Junior went looking inside its memory bank to find out what that word meant. When BASIC couldn't find COMPUTE on its list of commands, it told you that you must have made a mistake, an error. And so you got an error message.

Is the situation hopeless? No. You could slip a game cartridge in and start playing immediately. But that's not why you bought a Junior, is it? What you want to do is learn to master this machine. And things aren't going too well right now. The machine is winning, and you're not even playing a game.

Can you do anything with a computer without writing a program first?

Yes—if someone else has written the program for you. Video-game cartridges work without your doing any programming at all because programs are stored inside the cartridges and fed into the computer.

Ease of operation is important in helping new users make a graceful transition into the computer world. Above all, Junior was designed to be easy to get along with and use. Its software and memory arrangement are designed to make it easy for the programmer to take advantage of all the business-, educational-, and entertainment-oriented features of the machine. Every Junior has a program that teaches you how to use the computer and cassette BASIC built right in. In seconds you can be learning about and interacting with your new computer. You need buy nothing more to start programming.

Junior Shows Its True Colors

From the look of the first screen you might think Junior is a drab computer. Look at that—white letters on a black background. How unimaginative. How unimpressive. How about doing something to change it? How about learning your first BASIC command?

Table 3. *Junior Color-to-Number Translation*

Color	*Number*	*Color*	*Number*
Black	0	Dark gray	8
Blue	1	Light blue	9
Green	2	Light green	10
Cyan	3	Light cyan	11
Red	4	Light red	12
Magenta (purple)	5	Pink	13
Brown	6	Yellow	14
Light gray (white)	7	White	15

The command that changes the screen colors is easy to execute. All you do is type in COLOR. It doesn't matter whether you use lowercase letters or capitals—Junior will understand no matter how you type the word. Try typing the word COLOR on the Freeboard, and once it's properly displayed on the screen, press Enter.

If you try just that, you'll soon be wondering what the mysterious message "missing operand" means. That's just BASIC's way of saying you left something out. You forgot to say what color you wanted to change to. But Junior won't understand if you type in RED, BLUE, GREEN, or LAVENDER. It knows colors only by code numbers, which you'll find in Table 3.

Try typing COLOR 2. Don't forget the space between the word and the number. If you don't include the space, Junior will try to find a command called COLOR2 in its memory and then display the "syntax error" message.

However, if you've correctly inserted a space, there is a perfect match and Junior follows the instructions, contained in its memory, that BASIC has linked to that command word. In Junior's BASIC language, spaces after command words are very, very important.

After you've typed COLOR 2, press Enter so that Junior knows that you're done with the command.

Voilà! The next OK on the screen is in green, as will be everything that you type thereafter—providing you're using a color television or color monitor with your computer. Try some other colors.

Note that when you try COLOR 0 you'll see nothing on the screen because black characters on a black screen are invisible. Obviously, when the character color matches the screen color, the characters will be invisible.

You can change the color of the whole screen (which Junior calls the

background color) using the same BASIC COLOR command. Junior assumes the second color number you give in the COLOR command is the one that you want the screen to be. However, you must separate the first number (character color) from the second number (screen color) with a comma or else Junior will think that you want one big number instead of two smaller ones, and it won't know what to do.

Choose only dark colors for the background. If you ask for a light color—a number between 8 and 15—Junior will subtract 8 from it and display the resulting, darker color.

To change the background, try typing this:

```
COLOR 15,2
```

You should get a bright white "OK" in a green rectangle. The whole screen doesn't turn green because Junior is lazy and changes only the color of the active area of the screen.

You can make the whole screen active by telling Junior to erase everything. To do this, type in CLS, which you remember as standing for "CLear the Screen." As Junior clears the screen, it changes the screen color.

Now you have a complete green screen except for the black border that runs around it. But you can even change the color of this border by adding another number to your color command just like you added the background color. You have a full choice of 16 colors for the border. Try typing:

```
COLOR 14,6,2
```

then clear the screen to get yellow characters on a brown background with a green border.

If you don't have a color monitor, you probably weren't very impressed by the changes the COLOR command made on your display screen. A command that monochrome monitor users might find more useful, however, is one that changes the characters displayed on the screen from wide to thin, from 40 across the screen to 80 across. Although the command will work even if you have a color monitor, unless your color display is an expensive RGB monitor, the narrow characters will probably be unreadable.

The command to change the width of characters on the screen is WIDTH, and it must be followed by either the operand 40 or 80, the number specifying how many characters you want in each row across your screen.

Try typing:

```
WIDTH 80
```

Note that the WIDTH command automatically resets the screen colors to white (actually light gray) on black.

To change back to Junior's wider, 40-column characters, type:

```
WIDTH 40
```

Operands other than 40 or 80 will cause BASIC to send you the error message "illegal function call." Just try again with one of the numbers that BASIC finds acceptable.

If you do have a color monitor, you can change the colors to please you after you change the screen width.

Computing with Your Computer

The one thing that you'd expect a computer to be designed to do is compute. By now you're probably thinking that it's about time that Junior did some real computing and showed off some of its mathematical prowess. You can command it to do your calculating very simply, by typing in the problem you want solved.

You must set up your questions in a way that BASIC will understand. All arithmetic problems must be arranged horizontally using some special symbols: + for addition, − for subtraction, * for multiplication, / for division, and ^ to indicate that the following number is an exponent. BASIC even has two special kinds of division—*integer division,* indicated by a backslash (\) instead of a standard slash, which ignores fractional parts and remainders; and *modulo division,* indicated by the word MOD, which gives merely the remainder to the division problem (for instance, 7 MOD 3 equals 1).

BASIC will read your problem and solve it. You don't need an equal sign because Junior will know you're waiting for an answer as soon as you type in the problem and press the Enter key.

Go ahead. Type 2+2, then press Enter.

In an instant Junior has calculated the answer and is proud to announce it's ready to figure out another problem.

Is something wrong? You mean you can't see your answer? What did you expect? You should remember that Junior is very lazy. It won't do anything unless you tell it to, and even then you must be very specific and

careful because a computer is very literal-minded. You asked Junior to answer a question, and it did. What you forgot to do was instruct Junior to tell you what the answer was!

PRINT is the command that causes the answer to your calculation to appear on the computer screen. If you type in PRINT 2+2 and press Enter, you'll get your answer. Try it.

Junior's BASIC even recognizes a quicker way of writing PRINT, a question mark. Just type ? followed by your math problem (you don't even have to skip a space) and you'll get an answer. Although you must add a space after the word PRINT to avoid a syntax error, BASIC ignores spaces in math problems. Type these variations:

```
PRINT 4 * 4
? 4*4
```

BASIC can solve easy or complex problems. Try giving it something to really think about with this sample problem:

```
PRINT ((4523/42)*(3-78)/4)^0
```

BASIC performs the operations called for in this problem in a very particular order. The part buried deepest in parentheses is worked out first, then the next deepest layer, and so on until the whole problem is solved. If two expressions are equally buried, BASIC solves them from left to right. BASIC even follows a specific fixed order in carrying out different operations, called *rule of precedence.* Within each level of parentheses, BASIC first calculates exponents, detects negative numbers (and affixes a negative value to them), and then does all the necessary division and multiplication. After all other multiplying and dividing, BASIC works out integer divisions, then modulo divisions. When that's done, BASIC does the addition and subtraction and moves on to the right or the next level of parentheses.

Junior should have told you the answer to this problem was 1 because any number with an exponent of 0 is 1.

For a less trivial example of Junior's math capabilities, try the same problem but to the first power by changing ^0 to ^1.

If you don't want to type the whole equation over again, you can use some of Junior's secret powers to avoid repeating your keystrokes. Use the cursor controls (arrow keys) in the diamond-shaped array at the lower right-hand corner of the Freeboard. Move the cursor directly under the character you want to change, the 0. Type the new number directly over

the old one. When you press Enter, BASIC will look at this whole line as if you just typed it and give you a new answer!

Now try raising the number in the sample problem to the third power, ^3, and press Enter.

Does your answer look a bit strange, with an E and a + (plus sign) in the middle? What you see is BASIC's way of writing numbers in scientific notation. If BASIC finds an answer that uses too many digits, it converts them to scientific notation. In the answer, the number to the left of the letter *E* should be multiplied by ten to the power of the number to its right. In more human terms, the answer to the sample problem would be -8.232575×10^9.

Occasionally you'll get an answer with a *D* instead of an *E*. The D follows the same rule but it also indicates that the answer is given in *double precision* (hence the *D*), with double the normal number of decimal places of accuracy.

Single-precision numbers have seven digits, only six of which BASIC guarantees as accurate. In double precision, BASIC prints 16 digits, all of which are accurate.

You can force BASIC to think of the numbers that you give it to calculate as double precision by adding a pound sign (#) to the end of the problem. Try adding # after the ^3 in the sample problem, and you'll see the extra digits of double precision. (If you want BASIC to think in single precision, follow the end number with an exclamation point.)

You can also enter numbers in scientific notation to match the form in which BASIC prints them. The plus sign (+) to indicate positive powers is optional. If you add either a *D* or an *E* after a number with no exponent following, BASIC will treat the number as if the *D* were a # and the *E* were an !, eliciting double- and single-precision responses, respectively.

Complex Calculating

So far, the problems that Junior has worked out have been easy, and it has shown its calculating ability to be equal to any sheet of paper and pencil. But Junior can do much more, as its built-in BASIC is very rich in advanced mathematical functions that it can carry out with simple commands.

If you plan to do a lot of computing with Junior, you may find the following BASIC advanced numeric functions helpful.

Remember that you must type these functions in carefully. To avoid syntax errors, be sure to put the number or formula to be operated on in parentheses after the name of the function. However, BASIC doesn't

require a space between the function name and the number or formula that you want to work the function on.

ABS: makes BASIC ignore the plus or minus sign associated with a number, giving its absolute value. For instance, if you type PRINT ABS(−3), BASIC will answer 3.

ATN: finds the arctangent of the number following in parentheses. The answer is an angle (in radians) corresponding to the tangent of the number following the command. When you type in PRINT ATN(30), BASIC will give the answer 1.537475.

CDBL: converts the number following in parentheses into a double-precision number. (Remember it as Convert DouBLe.)

CINT: converts the number following in parentheses into an integer by rounding it off. Typing CINT(12.34) would produce the result 12. (Remember it as Convert INTeger.)

COS: determines the cosine of the number following in parentheses. BASIC assumes the number is given as an angle measured in radians. For instance, when you type PRINT COS(4), BASIC answers −.6536437.

CSNG: converts the number following in parentheses into a single-precision number. (Remember it as Convert SiNGle.)

EXP: makes BASIC calculate the value of "e" (the base of natural logarithms) raised to the power of the number following.

FIX: converts the number following in parentheses into an integer by lopping off all the digits to the right of the decimal point. In other words, typing FIX(4.965) would produce the answer 4.

INT: returns the largest integer less than or equal to the number.

LOG: gives the value of the natural logarithm of the number following in parentheses. The number cannot be 0 or less. PRINT LOG(12) would result in 2.484907; PRINT LOG(−12) would produce an "illegal function call" error message.

RND: gives a "pseudo" random number between 0 and 1. The

number that follows in parentheses serves as a "seed" from which the random numbers are calculated. Every time the RND operation is called, a new "random" number is calculated in sequence. The same seed number always generates the same sequence. If the seed is 0, the last random number given is repeated.

SGN: makes BASIC return a value based on whether the number following is positive, negative, or 0. Positive returns a 1, 0 a 0, and negative a −1.

SIN: determines the sine of the number following in parentheses. BASIC assumes the number is given as an angle measured in radians. For instance, when you type PRINT SIN(4), BASIC answers −.7568026.

SQR: gives the square root of the number following in parentheses.

TAN: determines the tangent of the number following in parentheses. BASIC assumes the number is given as an angle measured in radians. For instance, when you type PRINT TAN(4), BASIC answers 1.157821.

Programming in BASIC

Computer programming is not too tough to understand. A program is just a list of instructions—a set of problems for the computer to solve—arranged in step-by-step order. The computer follows each command and works the problem, then advances to the next one. The various computer commands are no more difficult to understand than any of the simple ones that you've already used. In fact, using just the few computer instructions you used before, you can write your first program. All you need to learn is the proper way of stringing the individual instructions together so that the computer goes through them in the right order.

You can actually make use of BASIC programs with absolutely no knowledge of BASIC at all. After you search through books and magazines full of program listings and find a program that you like, all you have to do is type it into your computer character by character exactly as it is shown. Simple. Just remember that there are subtle differences between different "dialects" of the BASIC language. Your Junior will only run programs that do not contain graphics or *machine-specific* commands that have been written for computers other than Junior.

Nevertheless, an understanding of the underlying structure and workings of BASIC will unlock the mystery of prefabricated programs and help you customize them for your own purposes.

The rules for programming Junior in BASIC are simple. Commands are written on program lines, and each separate line must begin with a *line number* that tells the computer in what order to work the problems. It will execute the command on line 1 before trying line 2. You indicate the end of each line of the program by pressing Enter. A line can consist of one or more commands—if more than one command is on a line, however, they must be separated by a colon. The line numbers need not be consecutive.

The BASIC language used by Junior has two modes: an *immediate*, or *direct,* mode in which it reacts to your commands as soon as you press Enter at the end of the command line (you've been using this mode) and a *program* mode, which allows you to type almost any number of program lines—until you run out of those "BYTES FREE" that IBM alerts you to when you turn Junior on—and store them in memory to be executed one after another when you command the computer to RUN the program. The only outward difference between the commands in the two modes is that program-mode commands must be preceded by a line number, and immediate commands must not be.

Creating and Destroying Your First Program

Taking the first step from direct mode to program mode isn't hard. Just type the following lines and hit Enter after each one:

```
1 Y = 2 + 2
2 PRINT Y
```

Once you've typed in the program, tell the computer to run it by typing RUN. Instantly you should have your answer.

Big deal. You could have typed "PRINT 2 + 2" and got the same results, but there is a big difference. The computer has memorized what Y stands for, and it remembers that value until the next line. In fact, it will remember the value you assigned to that Y until you decide on a new value for it! Whenever you call for Y later in your program, it will use the value you've assigned. Try this:

```
3 PRINT Y
```

Now if you RUN your program, you'll get your answer twice. You didn't have to type lines one and two again because Junior remembers them—until you either type new lines with those numbers or tell it to erase its memory.

In mathematics (and computer programming), a letter that can be assigned any numeric value is called a *variable* because its value can vary. Although Y is equal to four in your program, you can change its value by changing the numbers in the first line of the program. The opposite of a variable is a *constant,* a number whose value does not change.

In the version of BASIC that Junior uses, you can name variables almost anything that you want, with a few restrictions. The first character of the name of a variable must be a letter of the alphabet. The other characters in the variable name can be either letters, numbers, or periods. You cannot name a variable with a reserved word that BASIC understands as a command. Although you can make variables any number of characters long up to the maximum line length of 255 digits, BASIC only recognizes the first 40 characters as significant. If two variables have the same first 40 characters, BASIC thinks that they are identical.

BASIC recognizes several classes of variables. To force BASIC to store the value of a variable as a single-precision number, make the last character of the variable's name an exclamation point (!). A pound sign (#) forces BASIC to store the variable as double precision. A percent sign (%) at the end makes BASIC store the variable as an integer. A dollar sign ($) at the end indicates to BASIC that the variable is not a number at all but a collection of characters called a *string.* You'll soon see how to use strings and string variables.

Variables are extremely important to computer programming. An example will show you part of the reason why. Try typing this:

```
1 X = 2
2 Y = 3
3 Z = ((X * Y)/(Y + X)) * ((X / Y) - (X * Y))
4 PRINT Z
```

RUN the program after you have checked that all the parentheses are properly paired. If you work the problem longhand, you should come up with the same answer as the computer. If your answers don't match, you'd be better off checking your work rather than taking Junior in for repair! Note, however, that you've only had to type the value of both X and Y once, while the computer has used the numbers in the formula four times each. When both X and Y are only one digit long, that doesn't make much difference. To see how assigning a value to a variable can save you some time, try typing this:

```
1 X = 3434
2 Y = 5432
```

and RUN the program. Voilà! A different answer! By only changing the values of the variables you've got a new answer without worrying about retyping the problem itself. Every place X appears, the computer substitutes the value you chose in your new line 1. Ditto for the new Y in line 2.

As great as variables are for saving keystrokes, it's not the best use for them. Changing the values of variables automatically and under program control is one of the most powerful abilities of Junior and its BASIC language.

You should also have noticed that whenever you retype a program line, using the same line number as an old one, Junior automatically replaces the entire old line with the new one, making changes in programs easy. It does not matter when or where you type replacement program lines, nor does the order in which you type the original lines matter. Before Junior runs a program, it sorts through all the line numbers and arranges all the program lines in proper numerical order, starting with the lowest one.

If you think about it, you should find a potential problem. Old program lines can get accidentally mixed with new program lines. If there was no way of weeding unwanted old lines out, pretty soon there'd be a goulash of commands that even the computer couldn't sort out. If this havoc went on too long, the entire memory of the computer could be swamped with old programs. One way to erase an unneeded program line is just to enter the line number of the unwanted line and press Enter. The blank line replaces the old one, and Junior, not wanting to be bothered by nonentities, just lets line numbers followed by big blanks slip from its memory.

Alternately, you can delete one or more lines with BASIC's DELETE command. (Don't confuse BASIC's DELETE for the command ERASE in DOS which is abbreviated DEL.) Typing DELETE, then a line number, will erase that line. You can delete a sequence of line numbers by specifying the beginning and end line separated by a hyphen. For instance:

```
DELETE 150-200
```

would delete all the BASIC program lines between 150 and 200, including both 150 and 200.

Obviously, deleting individual lines and sequences of lines can get time-consuming, especially when you have to keep checking which lines are left to delete. BASIC gives you a quick way to eliminate old programs in their entirety to make room for a new program. That command is NEW. Type in:

```
NEW
```

and press the Enter key. Nothing happened? It only looks that way. Type RUN again. You're back to nothing more than the OK.

Your program is gone, but it didn't die in vain. You've learned that you should always type NEW before you start entering any program.

Learning Loops

Although we used line numbers that increased one by one to emphasize the step-by-step way programs run, most experienced programmers number their program lines in increments of 10, starting at 10 or 100. By skipping 10 numbers between program lines, a programmer can add an extra line—or several—between the other line numbers if he or she forgets to include a step in the program. It happens all the time.

Once you've squeezed in extra line numbers, you can space everything out by telling BASIC to renumber the lines with the command RENUM. IBM's BASIC changes line numbers wherever they occur in the program.

You can even make BASIC automatically type line numbers for you as you enter your program. Type AUTO and BASIC types in the first line number as 10. Every time you press the Enter key, BASIC puts the next line number (10 higher) on the screen for you. The only way to stop the AUTO process is to type Fn then *B* for Break after the first line number you don't want to use. (If you type Fn then *B* on the last line of your program, BASIC forgets that line!)

It's time to write a program that will give you some genuine (and maybe unexpected) on-screen results. Try this. Remember—type in everything *exactly* as it appears here:

```
10 FOR A = 1 TO 20
20 PRINT 999
30 NEXT A
```

en you RUN the program you should get a column of 999s, almost all vay to the bottom of the screen.

ou've just created a FOR-NEXT loop. It makes BASIC repeat all the tween the FOR and the NEXT. The A in line 10 is a variable (which ually as well be represented by any legal numeric variable name) IC holds in its memory. When the loop begins execution, BASIC iable equal to the number after the equal sign. Then BASIC

carries out all the commands on the lines between FOR and NEXT in their proper order. Each time it reaches NEXT, BASIC jumps directly back to FOR, adds 1 to the variable it is holding in memory, and tests its value. If the variable is less than or equal to the number after the TO, BASIC goes through the loop again. When the number in memory is greater than the number after the TO, BASIC skips to the line after NEXT.

If you use only one FOR-NEXT loop, NEXT requires no operand. However, you can put one FOR-NEXT loop inside another (using different variable names) to create "nested" loops, but you must indicate which NEXT belongs to which FOR by following each NEXT with the name of the variable used in the FOR line.

You can even make BASIC print out the value of the variable that it holds in its memory as the loop executes. Change line 20 so that your program looks like this:

```
10 FOR A = 1 TO 20
20 PRINT A
30 NEXT A
```

Now you should see a column of numbers, 1 through 20. You can also make BASIC alter the way it adds to the variable held in its memory. Try changing line 10 so that your program looks like this:

```
10 FOR A = 1 TO 20 STEP 2
20 PRINT A
30 NEXT A
```

Your program should now count up by twos. By changing the numbers, you can make BASIC count as high as you want in whatever intervals you want. You can even make it count down by using negative steps!

Punctuation marks mean a great deal to BASIC. For example, change line 20 of your FOR-NEXT loop by adding a semicolon so that it looks like this:

```
20 PRINT A;
```

Run the program. Ending the line with a semicolon told the computer not to advance a line or skip a space before PRINTing the next A.

You can also end lines with a comma. Try:

```
20 PRINT A,
```

The comma makes the computer "tab" between characters; it starts the second character at certain predetermined column positions.

Now let's add something to our program to emphasize why programmers number their lines in multiples of ten. Without using the NEW command (because we're not yet done with the old program), type in:

```
15 PRINT "THIS IS LINE NUMBER ";
20 PRINT A
```

The semicolon must be outside the last quote mark! Now run the program.

The computer inserted line 15 into the middle of the loop and used this line before each number in the loop was typed. But what about those quotation marks?

Anything within quotes following PRINT will be sent to the screen exactly as typed. It's called a *string,* and it has some very important properties you'll soon be investigating.

The quotation marks tell Junior that the material typed between them is meant to stay together as a string, and that the individual numbers, letters, and symbols in the string should not be treated as a lot of different individual variables. A PRINT command causes your computer to type out letter by letter everything in the string. (BASIC allows a little keystroke saving with strings—when a string is the *last* thing to appear on a program line, the last quote mark, which indicates the end of the string, can be omitted without causing BASIC confusion.)

Another of the powerful features of Junior's BASIC language is its ability to automatically change the values of variables within a program as it is running. To demonstrate this facility, type NEW, then enter this program:

```
10 J = 1
20 FOR I=1 TO 50
30 PRINT J
40 J = J + 1
50 NEXT
```

Line number 40 seems not to make sense, at least to us human beings, but if you run the program, the results will not look unusual. In BASIC it's perfectly acceptable and means that the value of J should be changed to one greater than its old value.

This feature of BASIC is quite versatile. Try this simple variation of the last program:

```
10 A = 100
20 FOR B = 1 TO 50
30 PRINT B","A,
40 A = A - 2
50 NEXT
```

Now the computer prints pairs of numbers. When one of the pair increases by 1, the other decreases by 2. Note that the comma is a string because it is enclosed within quotes, with the result that it is printed exactly as it appears in the program.

Closely related to the FOR-NEXT loop in IBM BASIC is the WHILE-WEND loop, which causes a program loop only as long as a given condition is true. For example, try:

```
10 WHILE A < 100
20 PRINT A
30 A = A + 5
40 WEND
```

The Amazing Jumping Program

Now that you have a good grasp of FOR-NEXT loops, you should try another way to loop around in BASIC. Junior can make loops with another command, GOTO. GOTO is sometimes called a *jump* because instead of letting the program progress step by step, it makes the program jump ahead or back to a given line number. In effect, a GOTO tells the program to unhesitatingly "go to" a given line number. The line number of the command that the program is to execute next must always follow a GOTO.

Type NEW, then type in and run this short program:

```
10 PRINT "HELP! I AM IN AN ENDLESS LOOP!"
20 GOTO 10
```

Junior will continuously print the string in line 10 until you press Fn then B to Break the program. Line 20 makes the computer go back and execute line 10 again, after which it progresses to line 20, which makes it execute line 10 again, and so on.

After you Break a program, you have a choice of two ways to restart it. Typing RUN starts the program over again from the beginning. Typing CONT (or Fn then 5) causes the program to continue running where it left off. The difference between the commands becomes very important with

long programs. With CONT you don't have to wade through all the preliminary steps of a program over again.

As a variation on the endless loop principle that lets you see what's happening, type NEW and try this simple program:

```
10 A = 1
20 PRINT "THE LOOP HAS EXECUTED "; A; " TIMES"
30 A = A + 1
40 GOTO 20
```

Note how the program inserts the numeric value of the variable A between the strings on line 20 each time the loop executes. The value of A increases with every execution because line 30 is inside the loop. If you reverse the commands on lines 30 and 40, the incrementing command is outside the loop. Try retyping those lines as shown:

```
30 GOTO 20
40 A = A + 1
```

The value of variable A will never change because the program never gets to line 40. It is locked in the line 20 and line 30 loop.

Closely related to the GOTO jump is the GOSUB command, a two-way jump. A GOSUB sends the program looking for a new line number out of the step-by-step sequence. Then the program jumps to the new line and starts executing a subsidiary part of the main program called a *subprogram,* or *subroutine,* there. The subroutine must end with a RETURN statement, which bounces the program back to *the statement after* the GOSUB command. (If it went back to the GOSUB command, the program would be caught in an endless loop, right?) Try this:

```
10 PRINT "THIS IS THE MAIN PROGRAM"
20 GOSUB 50
30 GOTO 10
40 PRINT "THIS LINE WILL NEVER PRINT"
50 PRINT "THIS IS THE SUBROUTINE"
60 RETURN
```

Line 40 never does print because it's outside the GOTO loop, and the program never gets that far. But the GOSUB command forces the program down to line 50, then back to line 30.

Subroutines are extremely useful because they let you use chunks of

a program inside another program as often as you want, yet you have to type the subroutine only once.

Making a LIST

After a while you can pile up so many commands in a program that it's difficult to keep track of them all. Fortunately, you can look at your program by typing in another command: LIST. This command lets you peer inside the computer's memory and see how it has rearranged the program. Type LIST, press Enter, and the whole program will roll forth from the computer's memory, with all of the program lines arranged in order.

If the program listing is longer than the screen, lines will scroll off the top of the screen as new lines are added at the bottom. If this scrolling goes too rapidly for your eyes to follow, you can slow it by holding down the Ctrl key and pressing *S* (Sc Lk or scroll lock) as the program is LISTed. Pressing any other key will start it going again. You can stop the LIST by pressing Fn then *B*.

You can also look at a single program line by typing LIST followed by the line number, or at a group of lines by typing LIST followed by the starting number, a hyphen, then the ending line number. For instance, LIST 50-100 would list all the lines from 50 to 100, inclusive. To LIST a given line number and all the lines thereafter, simply type the line number followed by a hyphen—for example, LIST 50- . To type a program line and all the lines that come before it, type LIST, a hyphen, then the line number—like LIST -50.

You can use the cursor controls and the Del and Ins keys to edit any line that is LISTed on the screen. Don't forget to press Enter for each line you change!

Getting Data In

The key to getting data into your computer while a program is running is the INPUT statement. When a BASIC program encounters INPUT, it stops and waits for something to be typed into the keyboard and for the Enter key to be struck. Further, an INPUT statement can prompt the operator by asking a question, or it can hint at what characters should be typed in.

First, let's try a program that changes the screen colors. Type this into Junior:

```
10 INPUT "COLOR NUMBER FOR CHARACTERS"; A
20 INPUT "COLOR NUMBER FOR BACKGROUND"; B
30 INPUT "COLOR NUMBER FOR BORDER"; C
40 COLOR A,B,C
50 CLS
```

Run the program. It will ask for the first color you want. Type in your answer and press the Enter key, and the program progresses to the next question. When you're done, it carries out your command.

You can type in any number you want in response to the question, and the computer will try to comply with your request. However, if you want to be obstinate about it and have the computer only accept numbers less than 15 to comply with the IBM scheme for naming colors, you can add a "conditional test" to each reply called an IF-THEN statement.

IF-THEN works by testing for the truth of the equation immediately following the IF. If the equation is true (A = A is true, 2 + 2 = 4 is true, and so on), then the program executes the command immediately following the THEN half. When that command causes the program to go to a subroutine or other separate part of the program, the program is said to *branch*. If the equation after the IF is false, the computer ignores the THEN and goes on to the next *line* of the program.

Add the following lines to the previous program (without typing NEW):

```
15 IF A > 15 THEN GOTO 10
25 IF B > 7 THEN GOTO 20
35 IF C > 15 THEN GOTO 30
```

The > sign means "is greater than" to BASIC; here, it tests the variable A—then B and C in sequence—to see whether the variables are larger or smaller than 15. Try running the program. Try to specify a color number 16 or higher. Just try! The IBM will stubbornly repeat the question until you come up with a reasonable value. Note that IBM BASIC is smart enough to know that when a number follows a THEN, that number must represent a program line to GOTO. If you want, you can omit the GOTO when typing programs. For clarity, however, the examples here will show all the GOTOs.

Closely related to IF-THEN statements are ON-GOTO and ON-GOSUB. Rather than just being followed by a single line number or subroutine to GOTO, ON-GOTO/GOSUB statements are followed by several. If the value of the ON part of the statement equals one, the program jumps to the first line number in the list. If the value equals two, the program jumps to the second line number in the list. And so on.

INPUT statements also permit letters (or complete strings) to be typed in instead of numbers. However, if the program expects a number and you type a letter, it will give you a "?Redo from start" error, which means, as you can probably figure out, that BASIC wants to give you another chance to type in the right information.

What makes the program finicky is the variable that follows the INPUT statement. If the variable is a letter or character combination (like A, XY, or BERTHA), the program will expect, and demand, a number. But when the variable is followed by a dollar sign (like A$, XY$, or DOLLAR$), the computer treats the INPUT as a string of letters or other characters without any numeric value at all. Strings that you type in response to INPUT commands should not be enclosed in quotation marks. Junior knows that your response will be a string because of the $ after the variable name, so it doesn't require quotes.

A variable with a trailing dollar sign is called a *string variable* because the computer remembers its value as a string of characters.

If you try to make a string equal to a numeric variable or something not within quote marks equal to a string variable, BASIC balks and sends you a "type mismatch" error. To rouse that error message, try these immediate commands:

```
A$ = 8605
B$ = Liederkrantz
C = "HAPPY DAYS ARE HERE AGAIN"
```

As with any string, the computer keeps the value of the string variable always in the same order, and only in certain circumstances will it manipulate the string. A string variable can be made equal to any string enclosed in quotes the same way any variable can be made equal to a number.

To see how strings can be manipulated, start by adding the next lines to your program:

```
60 PRINT "DO YOU WANT TO TRY ANOTHER COMBINATION?"
70 INPUT D$
80 IF D$ = "YES" THEN GOTO 10
```

If you type the word "YES" in response to the query from the INPUT statement, you'll run through the program again. Any other reply will dump you back to the OK.

You should note that when prompt strings are too long, or you want to use more than one line as a prompt for an INPUT, you can use one or

more PRINT statements to put words on the screen before the INPUT is asked for, instead of including the prompt string as part of the INPUT statement.

Most programmers don't want to be bothered by typing all three letters of the word YES every time a positive response is called for, so they would shorten the last line to:

```
80 IF D$ = "Y" THEN GOTO 10
```

Try it. Note that if you type the whole word YES in reply to the computer's query, it will think you said NO because Y is not exactly the same string as YES. There are two ways around the problem. So the computer operator knows what kind of answer the program expects, line 60 could be rewritten to say:

```
60 PRINT "DO YOU WANT TO TRY ANOTHER COMBINATION
   ('Y' OR 'N')?"
```

Better still, you can make the last IF-THEN line only look at the leftmost character in the response string. Replace line 80 by typing in the following:

```
80 IF LEFT$(D$,1) = "Y" THEN GOTO 10
```

The LEFT$ command is a string-manipulation function. It tells the program that part of the string should be pulled out and looked at separately. The LEFT part of the command tells the computer to start by taking the leftmost character. The $ indicates that it is a string function. The first variable in the parentheses is the string variable to be manipulated, and the second is the number of characters, starting with the leftmost, to be "pulled" out of the string. In this command, only the first letter on the left is examined. If it is a Y, the response will be considered a YES no matter what follows it. Y, YES, YUP, YEAH, and YELLOW ROSE OF TEXAS will *all* be considered YES answers by the computer.

You should note that your test is case sensitive. Lowercase and capital letters are not recognized as the same thing when they are part of a string. You might want to add another line to accept a lowercase y to send the program back to line 10.

Two other string-manipulating commands work similarly. RIGHT$(A$,n) takes the rightmost *n* characters of string A$. MID$(A$,m,n) will pull *n* characters from the middle of string A$, starting *m* characters from the left.

Type NEW to clear the old program from Junior's memory and try these lines:

```
10 A$ = "HIGH THERE, EVERYONE"
20 PRINT LEFT$(A$,2)
30 B$ = "EARTHLANDER"
40 PRINT MID$(B$,6,3)
50 C$ = "GOOD-BYE"
60 PRINT RIGHT$(C$,3)
```

The command LEN determines the number of characters contained in a string. The results you get can sometimes be surprising, however. Computers consider a blank space to be a character, too! Try this:

```
10 INPUT "WHAT IS YOUR NAME?"; A$
20 B = LEN(A$)
30 PRINT "YOUR NAME IS "; B; " LETTERS LONG."
```

Putting Data in Files

A file is Junior's way of handling data. Information must be tucked into a file to be sent to a peripheral or stored on tape or disk.

The easiest way to understand how files work is to think of them as file folders that can be stuffed into mailing envelopes. Before anything can be put into a file, the folder must be opened. The BASIC command that prepares a file for data entry is therefore called, logically, OPEN. A typical OPEN command looks like this:

```
OPEN "AUTOEXEC.BAT" FOR OUTPUT AS 1
```

Every file folder must have a name to identify it. IBM BASIC lets you give the file a name up to eight characters long when you use cassette tapes for mass storage, or eight characters with a three-character extension when you're using floppy disks. The file name, as a string, must be in quote marks, immediately following the space in the OPEN instruction. In this example, the file name is AUTOEXEC.BAT.

When you OPEN a file, you must also specify which way you want your data to go, either from BASIC as output or into BASIC as input. You can specify the data direction by using the word INPUT or OUTPUT after the word FOR after the file name. In this example the file will receive information from BASIC.

BASIC also requires a shorthand name for referring to the file in

other commands. You must use a number from one to the maximum number of files that you're allowed to have OPEN at one time. (This maximum varies according to the form of BASIC being used.) The example file is number one.

Another group of commands send and retrieve information from files—GET # (the pound sign is optional), INPUT #, and PRINT #. These commands work like their namesakes that lack the terminal # except they send data to, or receive data from, the specified file instead of the monitor screen. Each command must be followed by the number that identifies the file and then by a comma.

If your Junior is not equipped with the IBM disk operating system, it allows only four files to be OPEN at one time—the cassette drive, the monitor screen, the keyboard, and the printer port (a file can also be a device). If you use cartridge BASIC and DOS, however, ordinary start-up allows you to have only three files OPEN at a time, although through special instructions to BASIC when you start the language running from DOS you can get it to allow more files to be OPEN.

So that you don't run into problems, another command CLOSEs files that are open. A CLOSE command need only be followed by the number of the file to be CLOSEd. If no number follows a CLOSE, all OPEN files are CLOSEd. An END command also causes all OPEN files to be CLOSEd (and terminates the execution of the program).

Using data files properly is one of the highest arts of BASIC programming. Yet mastery of file functions is necessary only when your programs must manipulate massive blocks of data, which cannot all be kept in Junior's random access memory at one time. For now, probably the most important use you'll have for file commands will be to send text to the printer, which you'll learn more about in chapter 8 on printers.

Using the simple color changing you've already experimented with, you can build a special version of the disk file called AUTOEXEC.BAT (see chapter 5), which will set the color scheme on your monitor to what you want every time you turn Junior on or reset it by pressing the Ctrl, Alt, and Del keys (providing your Junior has a disk drive and DOS). The following program actually creates two files: AUTOEXEC.BAT, which tells Junior to automatically enter BASIC and run a program called SETUP; and SETUP.BAS, which automatically sets the screen colors to the ones you select while the program runs. You must have a disk that is not write-protected in your disk drive when you run the program.

```
10 CLS
20 INPUT "COLOR NUMBER FOR CHARACTERS"; A
```

```
30 IF A > 15 THEN GOTO 20
40 INPUT "COLOR NUMBER FOR BACKGROUND"; B
50 IF B > 7 THEN GOTO 40
60 INPUT "COLOR NUMBER FOR BORDER"; C
70 IF C > 15 THEN GOTO 60
80 COLOR A,B,C
90 CLS
100 PRINT "DO YOU WANT TO TRY ANOTHER
    COMBINATION (Y/N)?"
110 INPUT D$
120 IF LEFT$(D$,1) = "Y" THEN GOTO 20
130 IF LEFT$(D$,1) = "y" THEN GOTO 20
140 OPEN "AUTOEXEC.BAT" FOR OUTPUT AS 1
150 PRINT #1, "BASIC SETUP
160 OPEN "SETUP.BAS" FOR OUTPUT AS 2
170 PRINT #2,"10 COLOR ";A;",";B;",";C
180 PRINT #2,"20 CLS"
190 PRINT #2, "30 SYSTEM"
200 CLOSE
210 PRINT "Program completed. Files made.
```

If you do not have a disk drive, delete lines 140 and 150 from the program, and be sure you have your cassette recorder plugged into Junior, with both the record and play buttons pressed. If you store this program as the first file on a cassette tape and rewind the tape before you turn Junior on, you'll be able to set the colors on your monitor by simply pressing Ctrl and Esc when you see the BASIC sign-on message.

Keystrokes, Numbers, and Symbols

IBM has programmed the Freeboard to save you time when you're programming in BASIC. Using the Fn, Ctrl, or Alt keys, you can enter whole words or functions in just one or two keystrokes. But IBM doesn't like to reveal all of its secrets. Although the Fn key commands are shown at the bottom of your monitor screen, you have to get cartridge BASIC to find out about the Alt key commands, and you have to guess and experiment to find the Ctrl commands.

After you've been programming in BASIC for a while, the legends at the bottom of the display that describe the Fn key commands may get bothersome. Fortunately, you can turn them off with this simple command:

```
KEY OFF
```

To display them again, just type:

```
KEY ON
```

Although the table describing the whole words you can type by pressing the Alt key and a letter key at the same time can be found only in the PC*jr* cartridge BASIC reference manual, these functions will also work in cassette BASIC. For instance, to type the word COLOR, you need only hold down Alt and press the letter *C*. Table 4 lists all the words that can be typed using the Alt key.

Table 4. *The Key to Alt-Key Combinations*

Alt with:	*Becomes:*	*Alt with:*	*Becomes:*
A	AUTO	M	MOTOR
B	BSAVE	N	NEXT
C	COLOR	O	OPEN
D	DELETE	P	PRINT
E	ELSE	R	RUN
F	FOR	S	SCREEN
G	GOTO	T	THEN
H	HEX$	U	USING
I	INPUT	V	VAL
K	KEY	W	WIDTH
L	LOCATE	X	XOR

Pressing the Ctrl key with various letter keys also will elicit whole functions with single keystrokes; however, Ctrl-key functions cause BASIC to take immediate action instead of typing out the name of the function on the screen. Table 5 lists some Ctrl-key combinations that work in Junior's BASIC.

Actually Junior stores and manipulates all characters and symbols as numbers in its memory. Each letter of the alphabet is assigned a numeric code. To see the code number of any letter, just type ASC followed by the character within parentheses. If you put more than one character inside the parentheses, you'll only get the value of the first character. Try these immediate commands:

```
PRINT ASC(F)
PRINT ASC(FRANK)
PRINT ASC(7)
```

Table 5. *Control-Key Combinations*

Ctrl with:	*Becomes:*
B	PREVIOUS WORD
C	BREAK
F	NEXT WORD
G	BEEP
H	DELETE BACKSPACE
I	TAB
J	NEXT LINE
K	HOME
L	CLEAR SCREEN
N	END OF LINE
M	ENTER
→	PREVIOUS WORD
←	NEXT WORD
6	CURSOR UP
-	CURSOR DOWN
ESC	DELETE LINE

The process works the other way, too. With the CHR$ command, you can type in a code number to get back the character assigned to that code. The command is valid only for values from 0 to 255, but not all the numbers in between have characters assigned to them. Try these:

```
PRINT CHR$(1)
PRINT CHR$(42)
PRINT CHR$(70)
PRINT CHR$(122)
```

Detail Work—Remarks and Menus

Once you're able to write a program, the next step is to practice making programs that are easy to use and easy for other neophyte programmers to understand and learn from.

The command that helps others understand the programs you write is the one BASIC command that seems to do almost nothing at all. REM, which stands for remark, advises Junior to ignore everything else that follows it on a program line. If another statement precedes REM on a line, you'll have to place a colon before REM on the line. You can abbreviate the already abbreviated REM—or even : REM—with a simple single quotation mark.

The REM command is of immense value in documenting your software. It allows you to add comments to program lines to tell others (and yourself, as a reminder) what a program line is supposed to do. Documenting your programs properly helps others learn from your experiences and share your programming knowledge.

REM statements do take up valuable room in Junior's memory, however. When you want to crunch a program down to the smallest possible size, the first thing that you normally eliminate will be the REMs.

You can insert REMs anywhere in a program or even make an entire program from them. Try this:

```
10 REM THIS LINE WILL NOT PRINT
20 PRINT "THIS WILL": REM "BUT THIS WON'T"
30 'THIS LINE DOES NOTHING
```

When you copy a program from a book or magazine, you don't have to waste time typing in the REM statements or material following a single quotation mark. The comments don't do anything in the program, and the program should run flawlessly without them.

A Finished Program

The lengthy color-selection program you experimented with earlier might have been useful except for one problem—someone who has never seen the color codes for Junior will have no idea what color equals what number. To make the program usable by almost anyone, even those who have no idea what colors Junior can make, you need a *menu* that lists the available selections.

Using PRINT statements, you can paint a menu across the top of the screen. The following listing is a fine-tuned version of the color-selector program with a primitive menu. The initial GOSUB makes the program easy to add to the one you've already tried (and gives you a chance to try out a jump). Because lines 120 and 130 jump back to the GOSUB, if you elect to go through the color-selection process again, the menu is repainted on the screen. The new question in lines 150 through 170 (and the provision for responses to it) allow you the option of running the program to change the colors on your screen *without* updating your AUTOEXEC.BAT and SETUP.BAS files. The END command on line 320 stops the program from racing through the menu once it has completed its chore.

```
10 GOSUB 330
20 INPUT "COLOR NUMBER FOR CHARACTERS"; A
30 IF A > 15 THEN GOTO 20
40 INPUT "COLOR NUMBER FOR BACKGROUND"; B
50 IF B > 7 THEN GOTO 40
60 INPUT "COLOR NUMBER FOR BORDER"; C
70 IF C > 15 THEN GOTO 60
80 COLOR A,B,C
90 CLS
100 PRINT "DO YOU WANT TO TRY ANOTHER
    COMBINATION (Y/N)?"
110 INPUT D$
120 IF LEFT$(D$,1) = "Y" THEN GOTO 10
130 IF LEFT$(D$,1) = "y" THEN GOTO 10
140 CLS
150 PRINT "DO YOU WANT THESE COLORS TO BE SET
160 PRINT "AUTOMATICALLY WHEN YOU TURN YOUR
170 PRINT "COMPUTER ON";
180 INPUT Q$
190 IF LEFT$(Q$,1) = "Y" THEN GOTO 240
200 IF LEFT$(Q$,1) = "y" THEN GOTO 240
210 CLS
220 PRINT "COLORS HAVE NOT BEEN RECORDED
230 GOTO 320
240 OPEN "AUTOEXEC.BAT" FOR OUTPUT AS 1
250 PRINT #1, "BASIC SETUP
260 OPEN "SETUP.BAS" FOR OUTPUT AS 2
270 PRINT #2, "10 COLOR ";A;",";B;",";C
280 PRINT #2, "20 CLS"
290 PRINT #2, "30 SYSTEM"
300 CLOSE
310 PRINT "Program completed. Files created.
320 END
330 COLOR 15,4,1
340 CLS
350 PRINT "* * * * * * * * * * * * * * * * * * * *
360 PRINT
370 PRINT "* Automatic Start-Up Color Selector *
380 PRINT
390 PRINT "* * * * * * * * * * * * * * * * * * * *
400 PRINT
410 PRINT
420 PRINT " 0 = BLACK", " 8 = DARK GRAY
430 PRINT " 1 = BLUE", " 9 = LIGHT BLUE
```

```
440 PRINT " 2 = GREEN", "10 = LIGHT GREEN
450 PRINT " 3 = CYAN", "11 = LIGHT CYAN
460 PRINT " 4 = RED", "12 = LIGHT RED
470 PRINT " 5 = PURPLE", "13 = PINK
480 PRINT " 6 = BROWN", "14 = YELLOW
490 PRINT " 7 = LT GRAY", "15 = WHITE
500 PRINT
510 PRINT "* * * * * * * * * * * * * * * * * * * * * * * *
520 PRINT
530 PRINT
540 RETURN
```

The program looks long and complex, perhaps forbidding, but if you examine each line, you'll see that they're all familiar friends. They show how easily you can put everything that you've learned into one big program and get it all to work.

CHAPTER SEVEN

Pictures and Sound— Monitors

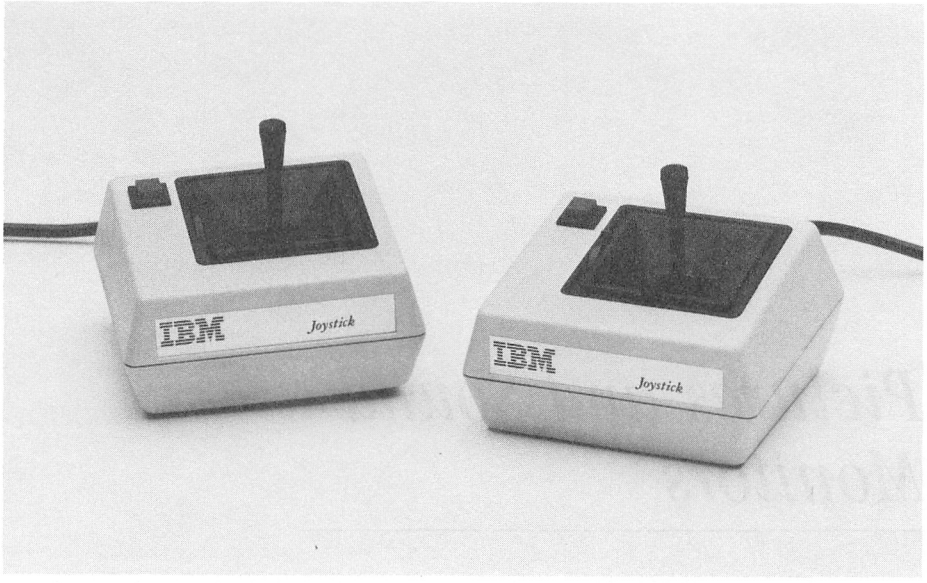
Just as the Freeboard lets you communicate with Junior, the monitor system is the computer's way of "talking" back to you. While a program runs, the display shows you what's going on inside Junior; when the program is done, you see the results on the screen.

The monitor system provides feedback when you are using the computer. As you press the keys of the Freeboard, the video display visually repeats (or "echoes") your every keystroke while the sound system clicks to confirm every button you press. Together, they assure that you are in control and that Junior's central microprocessor has received and understood all of your instructions.

Graphic images on the display can be the end product of a program itself, showing relationships clearly or serving as a window into a world of colorful aliens that provoke your imagination.

Junior's sound section is an important part of its monitor system, too. The three-voice music system is designed to plug directly into a stereo. Using the PLAY statement in BASIC, you can make Junior a musician *extraordinaire,* able to play Bach's three-part inventions at an unhuman rate or create sound never before heard this side of the moon.

Junior's monitor system is very adaptable. It is as sophisticated or as simple as the equipment you choose. You can use anything from a high-resolution color or monochrome video display and super stereo to an ordinary television set, either color or black-and-white. You can start your computer system by time-sharing your television set with the rest of



IBM joysticks are among the best available. PC*jr* users can plug two joysticks directly into their systems without needing any additional adapters.

the family, then graduate to a "dedicated" monitor—a video screen that is used for nothing but computing—when you get really serious about your on-screen work.

If you're new to computing, the difference between a television and a dedicated monitor may not be apparent at first. In truth, television sets and monitors look very much the same because both have big cathode-ray tubes (CRTs) as their screens on which they form their pictures. But that will probably be the only common ground between the two. Television sets and monitors differ not only in their inside circuitry but also in the signals that create the images on their screens. Televisions use broadcast "RF" signals; monitors use *video* signals.

Video signals come in several varieties. The one that interests us is a *composite* video signal, which is essentially a television signal before it is broadcast. A composite monitor is one that uses a composite video signal. Both monochrome and color composite monitors are available.

The best and most expensive color computer displays are called "RGB" monitors. RGB refers to the three primary colors of light—red, green, and blue—each of which is sent to the monitor as a separate signal through a separate wire. RGB signals are capable of producing sharper pictures than any other kind of video signals, and they are the choice for exacting video displays. The higher-quality signals also require higher-

quality CRTs to display all their detail, which means that a good RGB monitor can cost hundreds of dollars.

The Monitor Connection

Television

Once you decide on the video display you want for your Junior, connecting it is quite easy. Television sets use a special cable/converter called the IBM PC*jr* "Connector for TV." One end of the cable plugs into the jack on Junior's back marked *T* for "television," appropriately enough. The black box that swells in the cable as if it were a cobra that just swallowed a small cigar box is called a *modulator*. It converts Junior's computerlike sound and picture output into a standard television signal. You'll need a screwdriver to connect the short end of the wire leading out of the modulator. The two "spade lugs" on the end of the flat cable are meant to connect to your TV set's VHF (very high frequency) antenna input.

Although neither your television nor Junior *needs* to be turned off to connect them together, to be safe you should by habit turn off your computer system whenever you make connections to it.

After you plug everything in and are ready to operate Junior with your television, match the channel selector on the TV with the channel selected on the modulator, and throw the other switch on the monitor to the "computer" position. Although technically it doesn't matter whether you turn on Junior or the television first, it's more practical to have your television already warmed up and tuned in when you switch Junior on. That way you can use the initial IBM color bar display to set the TV's color and hue (or tint) controls and also see the diagnostic message in the screen's lower right-hand corner.

Unlike many modulators used with lesser computers, the modulator in Junior's TV adapter connects both Junior's video and sound output to your television set. You can use the speaker in your TV to listen to the music Junior makes.

If the sound is too fuzzy or distorted, try adjusting the fine-tuning control on your television. Although the technical quality of your TV set's speaker will probably be lower than you might expect from a connection to a good stereo, you may find it acceptable.

Composite Monitors

You can usually tell a composite monitor, whether it's color or monochrome, by the connector used for its signal. Most composite monitors designed to be used with personal computers use RCA-style phono jacks (or pin-plug jacks) for their connectors. The jack on Junior's back panel that is marked *V* (for "Video") is an RCA-style phono jack. Most stereo components also use the same style connector.

Connecting a composite monitor becomes a simple matter of running a cord between Junior's *V* jack and the video input jack on the monitor. Although you can invest in special video cables, for a short run—less than 2 meters or 6 feet—an ordinary hi-fi patch cord should work just fine. In fact, if your composite monitor has a built-in amplifier and speaker that uses an RCA-style jack for its input, you can use a single stereo patch cable for both the audio and video connections. Use one channel for audio and the other for video. If you get them confused, you'll have neither picture nor sound on your monitor. To cure the condition, just reverse the wires. There won't be any permanent damage.

Often a composite monitor will try to confuse the issue by having two video jacks—one marked "in" or "input" and one marked "out" or "output." (Sometimes they even have a third jack of the same style marked "audio.") Accompanying the jacks usually is a switch, most often marked "hi/low" or "bridge/term." The multiple jacks are there so you can "loop" the video through multiple monitors. Although it's unlikely you'll need more than one display with your computer system, you can attach two or more composite video monitors to one Junior when the multiple jacks are present.

Should you want to hook up multiple monitors, the wire coming from Junior gets attached to the "in" jack of the first monitor. A cable connects the "out" jack of this monitor to the "in" jack of the next monitor down the line. You can continue down the line for several monitors. The switch adjusts the monitor for its position in the chain. The last monitor (or only monitor, if you use only one) should have its switch set to the "term" or "low" position. All other monitors should be switched to "bridge" or "high." If you don't set the switches properly, the picture in *all* the monitors may lose contrast or be overly bright or dim.

RGB Monitors

Several different styles of connectors are used for RGB computer displays. Two types are guaranteed to work with Junior without a hitch: monitors specifically designed to plug directly into Junior and monitors designed to plug into other IBM personal computers.

Junior's RGB output is available at the jack marked *D* on its back panel. The *D* stands for "direct drive." Junior's direct video jack uses a full twenty pins because IBM strove to make Junior as compatible as possible with different computer displays. Besides the RGB signals, Junior's internal connections to those twenty pins include two different types of synchronizing signals and an audio output.

Monitors designed specifically to work with Junior usually come with a cable equipped with the special connector that mates with Junior's direct-drive output. Simply plug it in and switch your system on, and you'll be ready to compute.

So that Junior would be compatible with a wide variety of high-quality monitors, IBM also sells a short (about 4-inch-long) adapter cable that makes Junior compatible with just about every RGB color display designed for the IBM PC and PC-XT. To connect Junior to PC-style monitors, just plug the adapter cable into Junior's direct video jack, then plug the monitor's cable into the adapter. This adapter cable does not provide an audio signal connection, so you'll have to run a separate cable to your sound system if you want your Junior to make beautiful music for you.

Audio

Although Junior has an internal "beeper," a small piezo-electric speaker, it only beeps and clicks. It will not play the music or broadcast the noises that Junior's sound-generator chip makes. To hear Junior's sweet voices, you must connect the computer to an external amplifier and speaker system, which can be your stereo system, a separate sound system, or the audio section of a video display. (Of course, not all video displays have sound capabilities.)

The output of Junior's sound-generating system is available at the jack marked *A* (for "audio") on its back panel. Junior's audio signal is at the same level and impedance as most audio components, which means you can plug it directly into your stereo system. To do that, all you need do is run a patch cable from Junior's audio jack to any available high-level input on your stereo system's receiver or preamplifier/integrated amplifier. The "aux" and "tuner" jacks are the best candidates for the Junior sound connection.

The only problem you're likely to run into is that Junior, being a monaural computer, has only one output jack, and most stereo systems have two inputs for each source—one for the right channel and one for the left. To get Junior to sound through both of your stereo's speakers, either switch your stereo to "mono" when using Junior or buy a "wye" cable with one female RCA-style jack and two male RCA-style plugs. The wye cable will split Junior's audio output equally between your two stereo channels.

Why Buy a Monitor?

The chief reason most people decide to use a color television as their computer display is that nearly everyone owns one. Buying a computer display doesn't seem to make sense: why invest in another picture tube when the one already there works just fine?

The answer is simple. Junior is too good for your television. When a television is used as Junior's video display, much of the computer's innate talent is lost. Most of the time, the on-screen image made by television sets is disappointing—especially for serious programming and business applications—because television sets cannot handle the exacting detail created by a computer as sophisticated as Junior.

Color computer displays make television sets look like poor in-laws. Composite color monitors make bright, saturated (and stable) colors without a trace of snow or strange picture deformations like jagged edges and warped lines. High-resolution RGB displays go even further and reproduce more detail with greater control than any television can ever aspire to. Monochrome monitors produce sharp images that are easy on the eyes, so you can compute all day, and all night, without headaches.

If you plan to use Junior for serious computing, a high-resolution display will multiply the value of your system and make your work more comfortable and more understandable.

The first step up from a television is the monochrome computer display. (Although Junior will *not* support the IBM PC Monochrome Display, it will happily put characters and pictures on any *composite* video monochrome monitor.) Most monochrome monitors make sharper images than television sets—two or three times as sharp. With television sets, you can usually use only Junior's widest characters which fit 40 across the screen—the narrow ones blur and bunch together so badly you cannot read them. This puts many programs off-limits to a Junior so equipped. Monochrome monitors handle Junior's narrow, 80-letter-per-row character set with ease and can open up a new world of software to you.

Color computing is a step further. Color can add excitement and vivid realism to the most mundane games and programs. A color display is not just today's ultimate image for Junior, it's the wave of the future. Junior has been designed to be a preeminent color computer. Its built-in graphics capabilities put even the more expensive IBM PC and PC-XT to shame and encourage the use of color in computing. Most of the software being written for Junior relies heavily on graphics and color.

To take full advantage of all Junior's on-screen display capabilities, you must have a high-resolution color monitor. The variety of tints that

can be "painted" on its monitor separate and clarify elements of a display. For example, in an analysis of the annual profits and losses of a small business, figures for sales, salaries, rent, taxes, and so on can be sorted by color. And it goes without saying that charts and graphs are much more pleasing—and effective—in color. The best of the new programs that mimic the executive's desktop with "windows" corresponding to the many different sheets of paper that he must shuffle use a different hue for each imaginary piece of paper.

Color is so much a part of our lives that we think and dream in color. Color advances communication so that more than just ideas can be conveyed—it can telegraph emotion and feeling. It's only natural, then, that Junior should be equipped to take full advantage of this potential.

A Signal Difference

The features that distinguish a monitor from a television set are the type of input signal and the circuitry that processes that signal. Different video displays use different signals to produce their pictures. Once again, the three most common signal names you'll encounter are RF (or radio frequency, the signals used by television sets), composite video, and RGB (for red, green, and blue).

An RF signal is broadcast through the airwaves by a television station. Most television sets have RF inputs, usually called "antenna terminals." Junior by itself cannot produce an RF signal to feed to your television set. That's why you need to buy a special adapter cable with a built-in modulator, which acts as a small television station and changes the signals created inside the system unit into RF (usually channel 3 or channel 4) to use Junior with your television.

We've seen that a composite video signal is a television signal before it is broadcast. The composite signal combines all the brightness, color, and synchronization information necessary for transforming a television picture so that it can be carried through a single wire or communication channel. Unfortunately, when all of these elements are combined into one signal the technical quality of the resulting television image suffers. This is why TVs make poor computer monitors.

In the United States and Japan all color composite video signals must be made to meet an official standard called NTSC—National Television Standards Committee. (Among television engineers, the acronym unofficially stands for "Never Twice the Same Color.")

A standard signal was mandated so that the color signals sent by U.S. television stations could be received by any television set in the country, a

worthy goal that is now taken for granted. Before the standard was established thirty years ago, this was far from the case.

The big problem with color television back then was stuffing all sorts of extra information about the video picture and the color into the same space in the airwaves occupied by the simpler black-and-white signal. That all those colors could be squeezed into America's tiny television channels at all was a technical miracle in the 1950s, when vacuum tubes were high technology and room-sized computers were less effective than today's tiny Junior.

In most color video systems, the picture originates as three component images, each representing one of the three primary colors of light—red, green, and blue. Each one is created in a separate pickup tube in a single-color video camera. The NTSC system starts with the three separate color signals, and then mixes them together to create three more "intermediate" signals: one, called *luminance,* which specifies the brightness of the picture, and two, called *chrominance,* which code the picture's color information, or chrominance signals.

The luminance signal is the key part of the NTSC system that makes color pictures compatible with black-and-white television. By itself the luminance signal contains all necessary brightness or black-and-white information (essentially it is a black-and-white picture) and it will put a pleasing picture on most regular black-and-white television sets.

But the luminance signal leaves little room for color. The two chrominance signals must be squeezed together into a single subcarrier that shares a tiny portion of the television channel with the luminance signal. There isn't enough room left over to stuff in much color information.

Because of this space shortage, the resolution of some colors is restricted to one-third the black-and-white resolution by the NTSC process; others are restricted to about one-ninth! Worse yet, all the signal space given over to color is taken away from the black-and-white image, so its quality is limited somewhat, too.

Even this mixture is not complete, however. To make this squeezed-together package into true composite video, synchronizing and control signals are mixed into the luminance/chrominance goulash.

Perhaps the only reason color television is watchable at all is that the human eye cannot perceive color changes in as much detail as it perceives brightness changes. Therefore most of us do not notice the lack of sharpness in our color television pictures.

The important lesson in this is that although the black-and-white part of the NTSC signal produces a reasonably high-resolution picture, the color parts of the signal do not. Without the added confusion of color

information, composite video signals are good enough for reasonably high-resolution pictures on monochrome monitors—in fact, 80-column-wide displays are no problem. But composite color displays are limited by the NTSC processing to resolution sufficient only for IBM's 40-column-wide character set.

Fortunately, the strict limitations that regulate broadcast signals do not apply to personal computer systems. A computer monitor does not have to be designed to combine the color and synchronizing signals together; separate signals can be used for each color in an image, and even for the synchronizing signals. Of course, those separate signals are the original RGB, the picture before it is degraded by the NTSC process. Because RGB signals do not suffer the inherent limitations of the NTSC system, they produce pictures that are as sharp as or sharper than black-and-white images. That's why Junior's highest-quality color images appear on RGB monitors.

Trouble in the Tube

You might think that it would be easy to make a cheap computer monitor by chopping out of an ordinary TV set all the complicated circuitry necessary for handling NTSC-style television signals. Alas, though it can be done, the resulting picture wouldn't look much better. Even if you eliminate the problems inherent in the NTSC process, you're still faced with the limitations of the television set's picture tube (CRT). These tubes quite literally are unable to reveal all the intricate detail locked up inside computer signals.

A television picture is actually a collection of the glows of thousands of multicolored dots of a substance called phosphor on the inside surface of the screen. The dots are clustered in groups of three called *triplets,* each of which represents one picture element, or *pixel.* Each triplet has one phosphor dot for each of the primary colors of light, red, green, and blue. These dots can be made to glow individually or together using the energy in electron beams that are shot from three electron "guns" located in the neck of the picture tube. Each gun is electrically aimed at every dot of each of the three colors in a sequence from the top left-hand corner of the tube to the bottom right-hand corner.

Whether they are to be used in television sets or in computer monitors, all color picture tubes require a thin, finely perforated metal sheet between the electron guns and the phosphor dots to keep the electron beams meant for dots of one color from striking dots of another color. Because selected dots are kept in the shadow of the sheet of metal, the sheet is called a *shadow mask.*

The spacing of the holes in the shadow mask determines how closely the phosphor dots and the triplets made from them can be spaced on the monitor's screen. Therefore, the spacing determines how fine the detail of the picture on the screen can be. The closer the holes—and hence the dots—are to one another, the finer the detail that can be shown in the picture.

The distance between holes is called *dot pitch.* If the dot pitch is small, the picture on the screen will be good. Of course, the extra precision required to manufacture tubes with finer dot pitches makes them generally much more expensive.

Dot pitch is important because it determines the finest resolution that is physically possible to display on a picture tube. Junior's advanced color capabilities make resolution very important. To show its 80-column character set, in which each character consists of a matrix 8 dots by 8 dots, Junior requires a resolution of 640 dots (sometimes given in terms of pixels or lines) horizontally. If a computer display tube is 9 inches across (about right for the typical 12-inch *diagonally measured* monitor), those 640 dots must be displayed across roughly 230 millimeters, about 230/640 or 0.36 millimeter per dot. So, for a 12-inch color display of the IBM 80-column character set, the largest dot pitch for adequate sharpness would be 0.36 millimeter. Larger displays can use correspondingly larger dot pitches to produce the same image quality.

Monochrome monitors do not need shadow masks—the entire inside surface of their screens is uniformly layered with a single color of phosphor so there is no important *physical* limitation on their resolution. Nor are they limited by the NTSC color system. Instead, the quality of their images is limited by their internal circuitry.

To produce fine details on the screen, picture elements must be spaced close together. This requires that the electron beam energizing the phosphors inside the tube be able to change states more quickly as it sweeps by. In electronics, faster changes of state translate into a need for greater frequency response. Therefore, monochrome monitors with higher frequency responses (measured in megahertz, or MHz) will have finer resolution and show more detail on the screen. Most monochrome monitors have frequency responses of 15 MHz or higher, which is sufficient for clearly displaying Junior's narrow 80-column character set.

Counting Colors

Computer manufacturers often brag about the number of different colors their monitors can display—usually about six, eight, or sixteen. IBM

claims sixteen for Junior and the rest of its clan. If you've ever tried counting the colors in a television picture, you probably found many times that number. If a computer can show more detail than a TV, why then can't it produce more colors? To answer this question, let's look at another fundamental difference between computers and televisions and the signals they use.

Televisions can display dots of varying brightness using "analogue" signals, which vary in strength. Computers, on the other hand, use digital signals that have only two "states" or strengths, either on or off: their dots, therefore, are either on or off—lit or unlit.

Each primary colored dot at each position on the screen will be on or off. The specific combination of primary colors that are turned on at each position at any given moment determines the final color shown there on the screen. The three control signals in Junior's RGB output can be mixed together in eight possible combinations, resulting in the eight colors that can be displayed. The colors and the control combinations that produce them are as follows:

Color	**Control Combination**
Red	Red gun alone on
Green	Green gun alone on
Blue	Blue gun alone on
Yellow	Red and Green on
Magenta (purplish red)	Red and Blue on
Cyan (greenish blue)	Blue and Green on
White	All three guns on
Black	All three guns off

But this adds up to eight colors, and we know that Junior can actually display sixteen colors. To accomplish that, IBM has added an extra control signal to Junior's RGB trio, which regulates the intensity or brightness of the dot triplet on the screen. Because the extra signal is digital, each color of the eight possible color combinations has a bright and dim tint, for a total of sixteen.

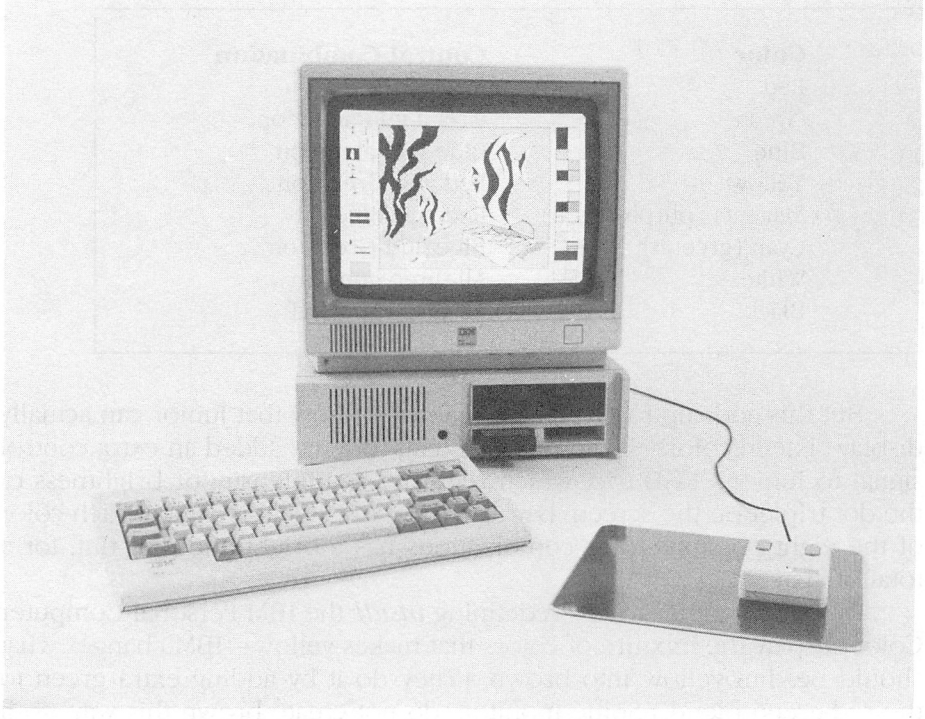
By using another trick—redefining *inside* the IBM Personal Computer Color Display the mixture of colors that makes yellow—IBM changes what should be dim yellow into brown. (They do it by adding extra green to the red-green blend.) Some monitors do not create brown this way, so if you're using Junior with another manufacturer's monitor, the brown might look different from IBM's brown.

Choosing a Monitor

Your choice of a monitor depends on a number of factors. Your budget may dictate that you use your television set. Even an inexpensive black-and-white set will be an adequate display for your own programming. But if you plan to do a lot of programming, use a word processor, or do bookkeeping or financial planning with an electronic spreadsheet, a high-resolution color or monochrome computer display is recommended.

In selecting the best computer display from the multitude that are available, trust your own eyes and good judgment. Once you are familiar with the strengths and weaknesses of the various monitors, you will probably be able to pick the best one without much difficulty. The first thing you must do, of course, is learn how to judge a computer display.

RGB color monitors adhere to the highest standards, and, not surprisingly, have the highest price—$500 and up. They must operate with so much precision that even their minor shortcomings are glaringly obvious. They can be the example from which you learn how to judge the



With the IBM *ColorPaint* program and a mouse, PC*jr* users can draw complex artwork with ease.

relative capabilities of displays. (All the features that you must consider in choosing a color display also apply to selecting a monochrome monitor, except for the color processing.)

In any color display, the first indicator of quality to look at is the set's *convergence*—how close to the mark the three electron guns come when aiming at specific colored phosphor dots. Together, all three of the electron beams inside any monitor must converge on exactly the right tiny area on the screen to illuminate a single set of phosphor dots (a pixel). Each beam must strike the dot of only one color. When a monitor is either improperly designed or badly manufactured, the electron beam trio will not converge properly at each pixel. Poor convergence results in a loss of sharpness and detail with rainbowlike shadows on each individual character when text is displayed. The problem is particularly noticeable at the edges and corners of the display screen because the electron beams must be bent most when pointing in those directions, and errors in aiming show up more obviously.

The best way to check for convergence problems is to fill the screen with *white* text in the 80-column mode and look to see that each character is crisp and clear, and not made from three wispy characters, one for each primary color. One way to nearly fill the screen with text is to run the following simple BASIC program:

```
10 WIDTH 80                'Sets 80-column mode
20 COLOR 15,0,0            'Sets text white on black
30 FOR A = 1 TO 2000       'Fills screen with X's
40 PRINT "X";
50 NEXT A
```

(The words to the right of the single quote marks are comments that need not be typed in to make the program work.) If you notice after running the program that the supposedly white letters really have distinct shadows or halos of red, blue, and/or green, you're probably seeing a convergence problem.

You can use that same program to check out the resolution of any color or monochrome display that you are thinking of buying. But if you are testing a composite color display, change line 10 to "WIDTH 40" to be fairer to its more limited capabilities. And if you want to be really critical, substitute the letter *M* for the *X* in the program; *M* is the most difficult character to display because it must be compressed to fit in the same amount of space as much narrower characters.

If you prefer knowing numbers to developing eyestrain by examining

all the monitors in contention, Junior requires a color or monochrome display to have a horizontal resolution of 640 dots (pixels or lines—they all mean the same thing) in order to display text 80 columns wide across the screen. The narrower 40-column text of the entry-model Junior requires 320 dots of horizontal resolution. Similarly, high-resolution graphics require 640 dots of horizontal resolution, and medium-resolution graphics require 320 dots horizontally.

Another problem that will be evident on a screen full of text is a condition known as *overscan.* Some displays try to make images larger than their tubes can handle and consequently the edges get cut off. The problem is called overscan because it's as if the electron beams try to scan past the edges of the tube. Look in the corners of the screen and be certain that no part of any character is cut off. Monitors with substantial overscan make computing a headache; imagine trying to guess at the characters of an unfamiliar text that look as if they've run off the screen.

The opposite of overscan is *underscan,* in which an image much smaller than the display tube is produced. Junior is actually designed to have a limited amount of underscan to ensure that all of its characters are displayed on all monitors. It paints a border around the active area of the screen, the part on which characters actually appear. The wider the border appears to be, the more underscan the display has. If you have a color monitor, you can make the border red so that you can see how wide it is by simply typing this one-line BASIC program (after the BASIC language is running, of course):

```
COLOR 15,0,4
```

If you don't see the border on a monochrome display, turn the brightness control up.

If you've run this program and you still see dark screen outside of the red border, the display has *severe* underscan. Although underscan doesn't hurt anything, it makes the effective size of the display's screen smaller than it really is, a factor you should take into consideration when comparing displays of different sizes.

If you plan to use Junior for graphics, you should make sure the video monitor you choose is free from geometric distortion. Straight lines should be straight, not gently flowing curves. The best way to check the drawing ability of a display is to construct a gridwork of squares on the screen and be sure that the squares are really square and not trapezoids, pincushions, or barrels. A cartridge BASIC program that will draw a gridwork for such checks is as follows:

```
10 SCREEN 2                    'Sets high-resolution
                                graphics
20 KEY OFF                     'Turns off on-screen
                                labels
30 CLS                         'Clears screen
40 DRAW "BM 0,0 NR 627"        'Sets cursor in upper
                                left
50 FOR X = 1 TO 8              'Repeats next line eight
                                times
60 DRAW "D 24 NR 627"          'Draws horizontal lines
70 NEXT X
80 FOR Y = 1 TO 11             'Repeats next line 11
                                times
90 DRAW "R 57 NU 200"          'Draws vertical lines
100 NEXT Y
```

This program also gives you a chance to check the convergence of the display in question. The lines of the grid should be white and not three separate images of each primary color lying close to one another. (But you must have IBM's cartridge BASIC installed in an enhanced Junior to run the program.)

Several characteristics of displays do not affect their on-screen quality but can be very important in making their images a pleasure to view in a variety of environments. Brightness, glare, and background color are qualities that affect how comfortable a display will be to use.

The brightness of the screen is determined by several factors, including the design of the tube and the power of the circuitry that operates it. Make sure that any monitor you consider is capable of making a clearly visible picture in the brightest surroundings in which you're likely to use it. Bright sun from a nearby window is enough to wash out many displays. Furthermore, be sure that when you turn the brightness up to the desired level the characters on-screen do not blur to illegibility.

When a room in which you'll be using your computer has a bright, high source of light, glare off the screen can be a problem. Most people find that some glare-reducing treatment is easier on their eyes. A number of computer displays have screens that have been etched or coated to reduce annoying glare. Although there are a number of different antiglare finishes that vary in durability (most resist the usual cleaning solvents), the important consideration is that the glare be reduced enough so that it does not interfere with your use of the computer.

In both color and monochrome displays, the background color of the screen when it is turned off can affect the way the display looks when it is

turned on. Darker (the so-called black matrix of color tubes) screens can generally show their images with higher contrast because the darker background makes the glowing color dots stand out better. Better contrast can mean better viewing, when all else is equal—which it rarely is!

When you buy a monochrome monitor, you are forced to decide on the color of screen you prefer. Although plain black-and-white is good for watching television, it's thought not to be the best choice for computing. Green is believed to be easier on the eyes and it's thought by many experts to be the best color to use when the surrounding light is dim. Amber (or yellow) is gaining favor as the preferred color in brighter work places.

The interference generated by a video monitor can be critical to the operation of Junior. Interference from its display can cause Junior to make mistakes in reading data from disk files and can cause it to beep constantly if it confuses the frequencies radiating from inside the display with the frequencies sent by the wireless Freeboard.

IBM recognizes these problems and recommends that you keep Junior at least 6 inches away from the IBM Personal Computer Color Display. You should follow the same advice when using other manufacturer's displays as they can have similar effects. And don't sit the display on the computer, as this may put too much stress on Junior, the plastic case is not designed to support the weight of the typical display. The case may become distorted and other disk-drive problems may result.

Alternately, you can look for a display that creates a minimum of interference. The FCC has defined two standards for the maximum allowable interference resulting from computing devices: a business standard, Class A, and a home standard, Class B. Of the two, the Class B standard is stricter, allowing less interference to leak out of the equipment in question. Your display should meet at least the Class B standard.

Finally, be sure that the cabinet styling of the display pleases you. The design has little effect on its performance, of course, but it can have an insidious effect on how you see your computer system. Every time you look at the screen you'll be seeing the cabinet, too. If it's an eyesore your computing just may suffer, without your even being aware of it.

Remember, you're the one who has to live and work with your computer display. The monitor you choose should be the one that looks and feels best to you, both on the screen and in your home or office.

A Brief Look at Some RGB Monitors

For high-resolution (80-column) color images, an RGB display is mandatory—but remember, only an enhanced Junior produces images that

can make use of these expensive displays. Following are brief critiques of some popular RGB monitors.

IBM PC*jr* Color Display

IBM's official monitor for Junior is a good choice but not the best. It overcomes many of the problems found in other monitors. It happily sits atop Junior, delivers bright, saturated colors of the official IBM spectrum. And it also has an amplifier and speaker to bring all of Junior's voices to life. However, although the PC*jr* Display is a reasonably good buy among inexpensive RGB monitors, Junior's own high-resolution graphics and color abilities far outstrip those of its official display. If you want the best to see what Junior really can do, you'll want an RGB monitor with a finer pitched tube. If you want versatility, for about the same price you can get an RGB/composite monitor that will work both with your computer and your video system.

The primary problem with the PC*jr* Color Display is its picture tube, which hardly climbs above television-set quality—its dot pitch is 0.63 millimeter. Although the display's dark, greenish face gives a picture with good contrast and rich, deeply saturated colors, it's barely sharp enough to show Junior's narrow, 80-column character set (the set's big dots make the on-screen letters appear fuzzy and blotchy, like newspaper photographs).

For programs using Junior's 40-column text mode or lower-resolution graphics modes (like games and many educational programs), the Color Display works well. The sound is good (for a televisionlike display), and the set generates so little interference that you can place it very near your Junior without making the computer squawk. Use the PC*jr* Color Display for serious work, however, and you may feel like you're trying to word process through a screen door. Further, its permanently attached cable, complete with a connector that plugs directly into Junior's *D* back panel slot, means that the PC*jr* Color Display can be used only with Junior.

IBM Personal Computer Color Display

IBM's "official" product, this is the standard by which all other color monitors must be judged. It performs so well with the enhanced Junior that it might have been designed just for that computer. It displays a full sixteen colors, all bright and saturated. Brown is really brown. Eighty-column text displays are no problem, nor are high-resolution 64O × 200-dot graphics displays.

The IBM Color Display uses a 13-inch tube with a dark, contrasty

black matrix. Its on-screen characters are big for an RGB monitor, easily readable from 4 to 6 feet away. But the tube barely produces the sharpness and detail Junior's high-resolution modes demand. If the brightness is turned too high, the characters are likely to blur together. The tube hasn't been treated to reduce glare, so it's not the best display for a bright room that's dappled with sunlight.

Although the IBM Color Display generates a good deal of interference, it is within the FCC Class B standard. When using it, however, IBM recommends that you place your Junior at least 6 inches away.

Amdek Color-II Plus

The Amdek Color-II Plus shows a bigger face than that of the IBM Color Display and a televisionlike case that's bigger than that of any other RGB monitors in its class. Alas, not all of that big screen is put to use—it has quite a lot of underscan, so its on-screen characters appear to be smaller than those on the IBM, notwithstanding the big face. The cathode-ray tube is sharp enough to display 80-column characters and high-resolution graphics.

The dark, black matrix tube shows bright colors with good contrast; but the contrast between "bright" and "dim" colors is not great. The screen has not been antiglare treated. The Color-II Plus is genuinely IBM-compatible, however, and does display real IBM brown.

The bulky case is very sturdy, making this the monitor of choice for harsh conditions, such as a household with young children. The big case does an exemplary job of keep interference in. Junior happily works right next to the monitor without a beep or "read error" to be found.

NEC JC-1216DFA

While other monitor makers strive to make their products resemble the official IBM standard as closely as possible, this NEC RGB display shows that you don't need to copy IBM to make a good-looking monitor. Although it looks like a portable television set, the 1216 outperforms any television and most other color displays. Its image is literally one of the sharpest, and its quality does not change appreciably as the brightness is turned up.

If you decide to go picking nits, you'll find a few in the NEC, however. The 12-inch tube of the 1216 produces smaller characters than does the IBM Color Display. What IBM calls brown, the NEC makes yellow. The face of the tube looks pale gray and it has not been treated to reduce glare. But interference is not a big problem with the 1216.

The Princeton Graphic Systems HX-12

The Princeton Graphic Systems (PGS) HX-12 monitor is a nearly exact clone of the IBM Color Display except for its smaller, light gray screen and sharper tube that has been treated to reduce glare. Its styling is a perfect match for the IBM and its circuitry is identical: what IBM calls brown, the PGS makes brown! The image on the PGS is noticeably sharper than that of the IBM, and it does not blur when the brightness is turned up.

The PGS picture tube measures 12 inches diagonally, but don't be fooled by its smaller dimension—the whole monitor is actually bigger than the IBM Display. The case is longer, with its full profile enveloping the tube neck for its entire length. IBM makes the case of its Color Display more manageable by shrinking the extreme back of the case into a small "neck extension."

Compared to the IBM, the PGS's interference shielding is very good—good enough that you can put Junior on top of it.

Taxan RGBvision 420

The Taxan 420 is probably the most compact 12-inch computer monitor around. Though styled similarly to the IBM Color Display, it looks about one-third smaller. Further, because it's set up with less underscan than most comparable displays, its characters are a tiny fraction *larger* than those on other 12-inch tubes.

The Taxan displays characters as sharply as does the IBM, but its greatest strength is its brightness. It can be used in bright, sunny rooms, and its medium gray tube face has been treated to reduce glare.

A weak point is its abbreviated palette. When called on to produce brown, it actually makes a red-orange that is virtually indistinguishable from genuine red. In effect, the 420 has a range of fifteen hues rather than sixteen when it is used with Junior. It also generates a substantial (though FCC Class B) amount of interference and can cause read errors and miscellaneous beeps if Junior is brought too near.

Amazing Adaptables: RGB/Composite Monitors

If you're looking for the best of both worlds, either one of the following monitors fits the bill. Both the Sears and Teknika have everything you need in a monitor for Junior—not just RGB inputs with a fine enough dot pitch to display 80-column text, but sound-monitoring abilities as well. Both let you use the monitor for more than just computing. Their

composite inputs will plug right into your videocassette or videodisc player for the best pictures you've seen from either. By adding a separate component tuner, the Teknika will become a top-notch television set. The Sears has a television tuner built in.

Sears 4084

If you can't make up your mind whether to splurge on your computer or on a television, forget the decision making and buy the Sears 4084. You might call it a monitor with built-in tuner. But then again it could be called a television with RGB inputs.

The Sears has most of the features that you might expect from a state-of-the-art television, like express tuning of 12 channels from the front panel. (It is not cable ready, however, so its range is only 83 stations rather than 105.) Performance as a television will seem disappointing when you switch from displaying your computer's output—the picture is "noisy" or "grainy" despite the best intentions of the high-quality tube. The culprit is the tuner section.

As an RGB monitor, the Sears has some nice features. Its 0.5 millimeter dot pitch and 13-inch screen make it sharp enough for 80-column text. A switch changes your Junior's white-on-black default display color selection to a more eye-pleasing green on black. Colors don't quite match the IBM specification, so you won't see a true brown, and you might find it difficult to discern the difference between bright and dim image intensity.

To let your Junior sound off, the Sears also has an audio input, which requires an RCA plug for input.

Teknika MJ-22

Even though its dot pitch (actually slot pitch) seems too big for 80-column quality at 0.5 millimeter, the MJ-22 outperforms Junior's official IBM monitor and comes very close (within 0.07 millimeter) to the on-screen sharpness of the IBM Personal Computer Color Display. Its color text displays are even sharp enough for everyday word processing without eyestrain.

Although the Teknika does make true IBM brown, overall its colors don't quite meet the IBM standard. Blue looks overly bright and cyan appears very gray (no matter how you set the brightness and contrast controls). The Teknika's image, however, is brilliant enough that you can read it outdoors (though a direct sunbeam on the screen will wash out most of the image). The screen is dark gray but not antiglare treated.

In normal view on the front panel, you'll find only a slender "volume" knob and a red, diagonally slotted pushbutton (the current fashion rage) on/off switch. Hidden behind a panel directly under the screen (which can be easily and inadvertently removed) is a complete array of thumbwheellike adjustments for horizontal position, vertical hold, contrast, brightness (which work in all modes), and color and tint (which work only in the composite mode). A single switch here controls the input between RGB and composite, letting you leave both an RGB source (like Junior) and a composite color source (like a VCR) attached simultaneously.

With rear panel switches you can also select two forms of RGB signals (type I works with Junior's IBM-style inverted synchronizing signal; type III mates perfectly with the Apple RGB standard), either NTSC composite or split luminance/chrominance video (which gives better on-screen quality with the Commodore 64), and even a screwdriver-adjustable control for the input video level in composite mode if your camera, recorder, or computer is a bit out of adjustment.

Audio and composite inputs use RCA (pin or phono) jacks. The RGB signal input requires an adapter for use with Junior.

In styling, the Teknika resembles a television set, hiding its actual depth with a clever design and a neck extension, which precludes putting Junior atop it. It's heavy enough so you wouldn't want to put it on Junior, either.

Composite Color Monitors

Composite color monitors will not produce acceptable results with an enhanced Junior in its 80-column text or high-resolution graphics modes, but can do yeoman service with an entry-model Junior.

Amdek Color I

The Amdek Color I is a family monitor with a big tube. It has a large, rugged case and a display tube about an inch bigger than most. The entire picture tube is put to use, giving the largest image area of any monitor listed here. Although the case is bulky, it's strong enough to withstand attack from small children and does a good job of keeping potential interference problems inside. A small plastic panel under the screen hides a full array of television-type controls.

Although the screen face is medium gray and has not been treated to reduce glare, colors of the Color I are bright and saturated. Sharpness is as you would expect for a composite monitor—fine for colorful 40-column

characters but unusable for narrow, high-resolution 80-column characters.

The Color I also handles Junior's sound output with a built-in amplifier and speaker. The audio input requires a miniature phone plug rather than the more common RCA-style connector, however.

Commodore 1701 and 1702

Commodore's color display models 1701 and 1702 are the same machine, and although it might not seem like a likely choice to plug into your Junior, it has one big advantage—it's cheap. It will easily produce acceptable 40-column images with Junior and has sound to boot. Both audio and video connections should be made to the jacks on its front panel.

Sure, the Commodore monitor has a few disadvantages: the intensity of its colors does fall a bit short of some of the more expensive composite monitors, and the styling of the monitor might clash with Junior, as well as with other decor. Nevertheless, if you're looking for a good buy, you shouldn't haughtily pass by this monitor.

NEC JC-1215MA

The cabinet of the NEC 1215 resembles that of the 1216 listed earlier, but on the inside it's entirely different. The 1215 will give a sharp, medium-resolution picture with intense colors. Behind an access door on its front panel the 1215 has a full range of televisionlike controls for manipulating colors. It has both input and output video jacks on the back, so with an RCA-style audio input jack and built-in speaker, it can give Junior a voice. Like its more expensive sibling, the 1216, the 12-inch, pale gray screen of the 1215 has no antiglare treatment.

Taxan RGBvision 210

Although the Taxan 210 has both RGB and composite video inputs, its CRT limits its on-screen image to below 80-column quality. When used in RGB mode, however, it can give a sharper 40-column picture than most composite displays.

The 210 has a full range of color-television-style controls (brightness, color, hue) on its back panel to supplement two on the front (contrast and combined on/off switch and volume control). Its audio input requires you use a mini-phone plug rather than an RCA-style cable.

Composite-Input Monochrome Monitors

Amdek 300A

Amdek monochrome monitors are probably the most popular replacements for IBM's own. Although the styling is very unlike IBM's, it's pleasing and quite compatible with Junior. The 300A is very sharp, more than is necessary for 80 columns of text, with bright amber characters. The 12-inch, high-contrast screen is nearly black, so it's a pleasure to use, particularly in brightly lit offices. Should you prefer a green screen, the model 300 (without the "A") is just that, with a lighter gray screen. Neither model has sound capabilities.

Be aware that Amdek makes a very similar model, the 310, which is IBM PC-compatible *but will not work with Junior.*

Panasonic TR-120MDPA

Panasonic now manufactures a series of monitors that are compatible with Junior. This one is a particularly good choice for 80-column monochrome work. It has both easy-on-the-eyes amber phosphors and a very good audio amplifier/speaker combination. The dark gray, antireflection-treated screen gives a high-contrast, readable image even in bright light. The color and styling of its plastic case is a good match for Junior. The internal sound system is more than capable of driving you out of the room. It's so light and sturdy that I've thrown one in the back of a station wagon and lugged it along on business trips. Unfortunately, its on-screen image is small for a 12-inch monitor and when the contrast is turned up, bright characters have a tendency to blur.

Sony Watchman

As unlikely as it may seem, you can use the tiny Watchman pocket-portable television with your Junior as a monitor. Granted that its 1½-inch screen is an invitation to eyestrain, that the image can't be varied from two not-very-contrasty shades of gray, and that the tube is only sharp enough for 40 columns of characters—but it does work.

The Watchman even has a monitor input so you do not need to use (and cannot use) Junior's television connector cable. Rather, get a patch cord with an RCA plug on one end and a miniature phone plug on the other (try your stereo store). Plug the RCA end into Junior's *V* jack and the miniature phone plug into the Watchman's monitor input

(on the upper left). There's no good way to get audio into the Watchman, however.

Zenith ZVM-123

Although it has a real 12-inch screen, this Zenith looks much smaller, and it has an unusually compact case that makes the tube actually look squarer than it is. Nevertheless, the green screen (pale gray when unlit) will easily display a full 80 columns of text.

Behind a small plastic door are controls that allow you to change both the horizontal and vertical size of the picture so you can control both its overscan and shape. Other controls let you adjust the contrast between bright and dim and even which of Junior's shades turn to true black in the background.

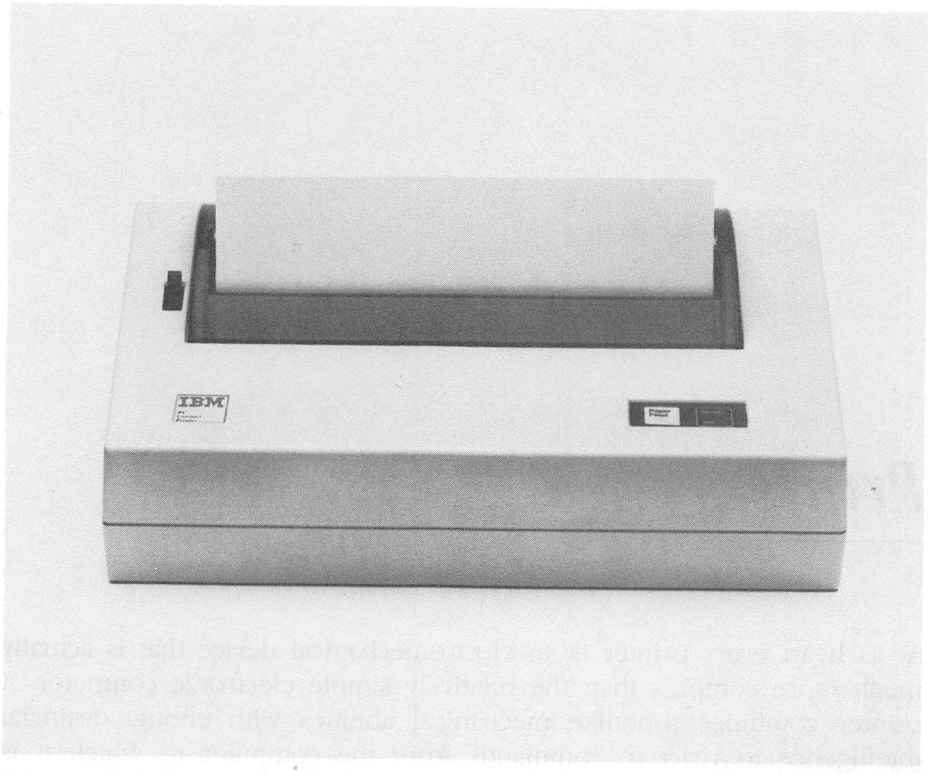
Although the color of its case marks the Zenith as having been designed for Apple computers, it works well with Junior. It has no sound capabilities.

Printers

At its heart every printer is an electromechanical device that is actually much more complex than the relatively simple electronic computer. A printer combines robotlike mechanical abilities with enough artificial intelligence to react to commands from the computer to which it is connected. Printers must be precision instruments, able to lay characters on paper with tolerances measured in a thousandth of an inch or less. Like clocks and watches, the quality—and the price—of printers varies considerably according to the skill used in their design and building.

If you're interested in printing as soon as possible but don't want to turn your mind into soft mush in the learning process, your best bet is to buy one of IBM's own printers. You have a choice of models, at a range of prices, that are guaranteed to plug into Junior and work with a minimum of confusion and fuss. You won't be reduced to a babbling idiot trying to decode the ravings other manufacturers laughingly call "installation instructions," which cover four dozen different computer connections in lighthearted prose that reads like the *Congressional Record.* The IBM Compact Printer will plug directly into Junior and come to life with your first command—whether you bother to read the instructions or not. Even though installing the IBM Graphics Printer and the IBM Color Printer is a bit more complex because you'll have to add an IBM printer adapter module to Junior, whether you have an entry model or an enhanced model you can do all the necessary work with a single screwdriver.

In truth, however, no printer—not even one bearing the proud IBM



IBM's inexpensive PC Compact Printer prints at a claimed speed of 50 characters per second and doesn't require any additional printed circuit boards.

logo—does everything well. Some printers produce beautiful color graphics, others are more utilitarian but very speedy. In making your selection you should test as many printers as possible, keeping in mind what you will be using it for. And by no means restrict yourself only to IBM products.

Printer Mechanics

Different printers are designed with various goals in mind, using technologies that optimize a given facet of the printing job.

The most common sort of printer that you'll encounter as a suitable replacement for IBM's own is the *impact* printer. They earn their name because one way or another they use the impact of a hammer to squeeze ink from a ribbon onto paper to make an image—the same principle that the classic typewriter is based on. Both the IBM Graphics Printer and IBM Color Printer are impact machines.

All impact printers have a number of desirable qualities. Their mechani-

cal design and function are straightforward, easy to understand, and reassuringly familiar. Like typewriters, they'll do the dirty work of transferring ink onto whatever kind of paper you can fit into them—most will accept any size or weight of paper, and even acetate for transparencies. Making several copies at once with an impact printer is simply a matter of loading it with carbon paper between individual sheets or with "carbonless" multicopy sets.

But because of the physical contact between hammer and paper, impact printers make noise, often lots of it. Their sharp staccato rattles and piercing whines can echo louder than normal conversation, and the noise can be a problem if you plan to do a lot of printing.

There are two kinds of impact printers. One constructs each individual character from an arrangement (or *matrix*) of tiny dots; the other forms characters all at once on a fully detailed mold, exactly the same way a typewriter prints. The former style of machine is called a *dot-matrix* printer; the latter is variously termed a *fully formed character* printer because of how the characters are made, *letter-quality* printer because at one time they were the only machines that typed clearly enough to be used for business correspondence, or *daisy-wheel* printer after the element holding the character molds, which resembles a flower.

The IBM Graphics Printer and the IBM Color Printer are both impact, dot-matrix machines. The print mechanism they use is similar to that of most common matrix printers. Each character is formed from many dots that are individually made by single tiny hammers, actually not much more than wires, which strike a fabric or plastic ribbon to transfer ink onto the paper.

In most dot-matrix printers a seemingly complex but efficient mechanism controls the print wire. The wire is normally held away from the ribbon and paper, and against the force of a spring, by a strong magnet. To fire the print wire against the ribbon and paper, an electromagnet is energized (under computer control, of course), which then neutralizes the force of the permanent magnet. Without the permanent magnet holding the print wire back, the spring forcefully pushes the print wire against the ribbon.

When the electromagnet de-energizes, the permanent magnet pulls the print wire back to its normal position, away from the paper and ready to fire again. One important consequence of this elaborate design is that when all power is removed and the printer is turned off, the print wire is safely retracted by the permanent magnet.

The print head of a dot-matrix printer is made from a number of these print-wire mechanisms. In a typical home or small-business dot-

matrix printer, seven or nine (or more) print wires are arranged in a vertical column to make the print head. To print a line of characters, this print head moves horizontally across the paper, and each wire fires as necessary to form the individual characters, one nine-dot column at a time. The individual print wires are fired as the print head flies across the width of the sheet; the print head never slows down or stops until it comes to the end of a line.

A major factor in determining the printing speed of a dot-matrix machine is the amount of time between successive strikes of each print wire. Physical laws of motion limit the acceleration that these wires can achieve and hence the speed at which they can move. Barring the repeal of the laws of nature, the time it takes to retract and reactivate each print wire puts a physical limit on how rapidly the print head can travel across the paper; it cannot sweep past the next dot position before each of the print wires inside it is ready to fire. If the print head travels too fast, dot positioning (and character shapes) become rather haphazard.

The quality of the characters printed by a matrix printer is primarily determined by the number of dots in the matrix of each character and how closely those dots are spaced. The denser the matrix—the more dots in a given area—the better the characters look. A 5 × 7 (horizontal by vertical) matrix is sufficient, but hardly satisfying, in rendering all the upper- and lowercase letters of the alphabet unambiguously.

However, such a minimal matrix is usually too small to let the descenders of characters such as *g, j, p, q,* and *y* go below the main line of type. This lack of true descending characters makes printed-out text look cramped and scrunched up. Better matrix printers, like the IBM Graphics and Color printers, use character matrixes with greater numbers of dots.

Although the letters and numbers made by matrix printers usually look rough, dot-matrix machines possess a unique advantage over the fully formed character printers. Each dot pattern of each character the dot-matrix machine prints is computer controlled and can be instantly changed by altering the computer's programming, without any mechanical adjustments to the printer at all. Although you can switch typefaces or sizes on a fully formed character printer simply by swapping print wheels, on a matrix printer such changes are even easier—just send a computerized instruction. You can change whatever you want whenever you want, provided the printer is equipped to carry out the instruction.

The programming instructions that a dot-matrix printer uses to create letters and numbers can actually make almost any pattern imaginable. Some of these machines can switch among alphabets and other strange built-in character sets that might include *block graphics,* picture-building

blocks each about the size of an alphanumeric character. Block-graphics characters are simple shapes like squares, rectangles, triangles, and horizontal, vertical, and diagonal lines from which larger, more complex pictures can be made.

Because each block-graphics character can be electronically coded and recognized by the printer like a letter of the alphabet, all the printer does to make a picture is lay down line after line of the strange symbols, sort of like filling in blocks on graph paper with different shapes. Of course, because the building-block characters are relatively big compared to the overall size of the finished drawings, pictures created from them look a little chunky or inelegant.

Most matrix printers can make more detailed pictures by allowing you to decide where to put each individual dot on a printed sheet. With the appropriate software, you can make such a printer draw graphs, charts, and pictures in great detail. Each individual dot can be printed at any location or *address* on the paper, so this dot-matrix printer feature is often called *dot-addressable graphics.* Other names for the same feature include *dot graphics* and *bit-image graphics.* Drawings made in this mode are reasonably sharp and of high quality because each individual dot is quite small, perhaps no bigger than 1/72 (or even 1/288) of an inch. The exact quality of the picture is expressed as *resolution,* the number of dots printed in each linear inch.

Junior's Compact Printer is also a dot-matrix printer and a good example of an alternative to the impact machines, called a *nonimpact* printer. The Compact Printer uses one of the two nonimpact technologies that are currently most popular for use with small computers, *thermal printing.* The other popular nonimpact printers are called *ink-jet* printers.

Like other thermal printers, the Compact Printer relies on the heat from a stylus to change the color of specially treated paper as the print head races across the paper. Thermal-printer technology is long-proven; it's been used for years in inexpensive calculator-type printers, and machines that use it can be very low cost.

However, the special thermal-sensitive paper can be hard to find and expensive, and most varieties of it feel slick and unappealing. All thermal papers tend to discolor with age and mistreatment, gradually getting either too light or too dark to read. Images on thermal paper deteriorate particularly quickly if the paper is left in sunlight.

Ink-jet printers use tiny nozzles to spray drops of ink onto the paper rather than hammering it on. Although the technology sounds exotic, the precision timing possible under microprocessor control makes the

machines both feasible and economical. A number of four-color ink-jet printers are now available at reasonable prices.

Nonimpact printers share some common characteristics: simpler mechanical construction than that of the impact machines and relatively quiet operation—both thermal and ink-jet machines are as quiet as printers get. But the very fact that they are nonimpact means that these printers cannot produce carbon copies. Keep that in mind if you have a business that uses duplicate invoices.

Most fully formed character printers are impact machines that use a slightly different character-making mechanism. Rather than having a number of print wires to shape the on-paper image, the print hammer in a fully formed character printer strikes an embossed impression of the character to be put on paper. The embossed impression serves as a mold to form the printed character. The hammer pushes the mold against the ribbon, which then strikes the paper. The ink is applied to the paper and forms the whole image of the printed character in one motion, exactly as a typewriter makes a whole character.

The fully formed character printer must move different character molds in front of the print hammer to form each different character to be printed. One of the fastest and simplest ways to do this mold shuffling is to make a printing element by arranging each mold at the tip of a spoke of a wheel. The wheel can spin each character rapidly into place as it is needed. Because of the shape of the printing element, these machines are called daisy-wheel printers. Sometimes the element resembles a thimble, and its machine is known as a *thimble* printer.

The most notable characteristic, and the greatest virtue, of fully formed character printers is their on-paper quality. Characters look like they were produced by a typewriter; they have serifs and round, smooth curves. They're the printers to select when the way your hardcopy looks is nearly as important as its content.

The biggest disadvantage of most inexpensive fully formed character printers is speed. Some home-quality printers of this type peck along at 12 to 20 characters per second (CPS). Although that may be quick in comparison with your own fingers, it doesn't compare with the 80- to 240-CPS speed of many equivalently priced matrix printers.

You might think that fully formed character printers should be faster than dot-matrix machines because they make only one strike per character as opposed to the dozens of dots in every matrix character, but the need for quality slows the fully formed character printers down. To avoid smearing its first impression, the whole daisy-wheel mechanism must come to a complete halt just before the hammer strikes; the daisy must

stop spinning and the carriage mechanism that moves it must stop and settle down before the character mold can be activated. After each hammer stroke, everything must speed up again to move to the next character. Quite literally, it's printing by fits and starts.

Understanding Other Printer Features

On most computer printers, the print head—either the print wire or daisy-wheel mechanism—races right and left across the paper to put each character in the right place, but it's not the only part that moves. To place characters in their proper vertical positions, the paper must be shuffled around as well. A number of different schemes have been developed to help both the printer and you deal with this chore.

The most familiar paper-moving mechanism is that of the typewriter, called *friction feed.* Squeezed between a large rubber roller, called the *platen,* and smaller rollers, the paper is moved into position and held in place by the friction between rubber and paper. Loading paper is usually a manual operation: you shove each individual sheet between the rollers, lining it up to be sure that it's square so the printer doesn't peck diagonally across the sheet.

The simple mechanism has its advantages and disadvantages. By itself, it's truly a manual affair because you must stand by giving the printer your constant attention, always ready to roll in the next sheet. But you can usually roll in any standard paper, including your own watermarked, raised-lettering, linen-textured stationery or preprinted forms like W-2s or 1040s. The friction-feed process is easy to understand and handle, but it's tedious if you want to print out a lengthy document.

An automatic *sheet feeder* (occasionally called *bin feed*) can relieve the tedium at a cost that matches the sophistication of the machinery—high. Too, although most bin feeders accept standard-sized forms and plain paper, many are designed to make a single copy at a time, no carbons allowed.

For jobs that involve many pages, continuous-form paper should be used. So that the "endless" sheet does not go askew and to make moving it easier, the edges of the paper are decorated with sprocket holes, like camera film. *Pin-feed* and *tractor-feed* mechanisms carry this special sprocket-holed paper through the printer.

Pin feed generally means that the sprocket that fits into the holes in the paper is permanently affixed to the edges of the platen roller, so it can handle only one width of paper. The two-sprocketed tractor-feed mechanisms mount on bars or rods and adjust to handle nearly any width paper that will fit through the printer.

As the names imply, *unidirectional tractors* only pull (or sometimes push) the paper through your printer in a single direction, perferably forward. *Bidirectional tractors* allow your printer to move the long sheet both forward and backward, often helpful for graphics and special text functions.

The mechanisms of some printers can move paper in very small increments, a fraction of an inch at a time. With such a machine, you can print letters at eight lines per inch for more punch per page and manuscripts at six lines per inch for shortsighted editors. Other printers, lacking this adjustable line height, offer only whole lines or half lines.

Printers also vary in the degree of control that you have over the amount of space between characters. Some printers put the space between each letter under the control of the software you're using, so you can change character pitch (number of characters per inch), even to the extent of making it vary for every character. This variation is called *proportional spacing,* and you can see an example of this in text in which a wide letter such as an *M* is allotted more space than a thin letter like an *I.* Less expensive printers usually cog between each character position mechanically exactly like an old-fashioned typewriter does.

Just as Junior is making monitor screens more colorful, the IBM Color Printer is adding hues to printouts. Other printers also have color capabilities. Some machines offer a minimal improvement over black-on-white by using standard two-color ribbons. Others add extra bands of color to a single ribbon like IBM's or shift between four individually colored ribbons. One fancy printer even uses a combination of both techniques with one black and one multicolored ribbon, knowing that you'll use black more often and so should wear out the colored ribbon less often. Dozens or hundreds of colors can be printed on some machines by using various combinations of ribbon and characters.

All of the ribbon shifting of most of these multihued printers is controlled by software containing special commands called *escape sequences.* That means, for the most part, you'll need to run special programs designed for these machines to take advantage of their rainbow of features.

The Printer Connection

No matter what brand and model of printer you choose to use with Junior, you've somehow got to connect the two machines together. Like every seemingly trivial chore in the realm of computing, making even a single connection may not be so simple. First, you've got to decide on the style of *interface* that you want to use.

Although the IBM printers are easy to add to your system, they will serve to illustrate the interface choice. The Compact Printer plugs into the *S* connector on Junior's back panel. The Graphics Printer requires you to add the printer adapter module to Junior, and then it plugs right into that. The two printers plug into different places because they use different interfaces; that is, they use different arrangements of the digital codes that transfer data from Junior to the printer.

A printer receives its instructions through an interface. A *parallel interface* sends all the data bits necessary to code each character simultaneously, each signal through an individual wire. A *serial interface* rearranges these same data bits and sends them one after the other, in a series, down a single pair of wires.

Of the two, the parallel interface is usually the easiest to use with Junior. The IBM printer adapter module adds a connector to Junior that acts like the parallel output of the IBM Personal Computer. Any printer that will plug into the IBM PC's parallel printer port will probably immediately spring to life when attached to a properly connected printer adapter on Junior.

The IBM PC parallel interface operates as an industry standard called a *Centronics* interface. However, because IBM chose to use an entirely different connector than is normally expected at the standard interface, cables and connectors labeled as "standard Centronics" will not work with the connector on Junior's printer adapter. (While IBM's decision seems illogical, there are extra wires attached to Junior's parallel printer connector that add control functions to the Centronics standard to increase versatility and convenience.) You must be careful, then, when buying a parallel printer. Unless it says it has an "IBM parallel interface," you'll probably need a special adapter cable to connect the printer to Junior's printer adapter.

Although it may take two elements to get things working—an adapter module and an adapter cable, you're nearly guaranteed success when you use a parallel interface. A serial interface is a bit trickier to set up.

There is a single standard specified for the style of serial (*asynchronous*) interface used by Junior—the ubiquitous RS-232 combination—but the so-called standard allows a wide variety of variations. Speeds, codes, and schemes of synchronizing signals and protecting against errors all can be changed to suit a given situation. If your printer is not set up to understand data the same way you set up Junior to send it (using DOS's MODE command), the result is likely to be a garbled printout or no printout at all. Even experienced programmers can be reduced to head scratching when attempting an unusual serial mating.

Serial interfaces can be set up at different data rates, which is the speed at which bits are shoved down the data line. The ability to work at different speeds makes serial interfaces more versatile. The longer the connecting cable and the poorer its quality, the lower the data speed it can reliably handle. Data transfer can slip through a serial connection as slowly as 5 to 11 characters per second in difficult situations. Yet when cables are short and the connections are good, a serial interface can whip through data at twice the speed of Junior's standard parallel interface.

Most printers have a limited amount of digital memory—called a *buffer*—built in so that they can look into the future at the data and plan their printing moves ahead of time. They can even print in both directions by looking at a chunk of their memory backward and printing the characters in it in reverse order on every other line. The buffer also allows the printer to receive data in quick bursts at speeds that do not correspond to the rate it puts characters on paper. The printer may absorb 2,000 characters in a fraction of a second and take 10 or 20 seconds to print them all.

When Junior sends a printer a long document made from more characters than the printer's buffer can hold, a problem arises. Because the Junior (or almost any computer) is capable of sending data many times faster than most printers type it and empty the buffer, some means of preventing Junior from racing ahead of the poor printer (and overflowing the buffer) is necessary. One way is to use a speed through the serial interface that is slower than the lowest rate at which the printer might be expected to absorb it and print it out, say about 300 baud (bits of data per second), equivalent to 30 characters per second. But 300 baud is very slow, indeed—and can mean that it will take several minutes to send a full page of text.

Another way to prevent data loss caused by the computer talking faster than the printer can listen is through *handshaking signals*. Essentially, the printer sends a signal to the computer that it has received all the data it can handle. The computer then stops sending data until the printer signals that it is ready again.

Handshaking can be achieved with hardware, most often through the use of an extra wire on which a change in voltage alerts the computer that the printer is busy or not (usually through pin number 20 on RS-232 cables). Junior's parallel port and the standard configuration of its serial port use hardware handshaking.

Handshaking can also be controlled by the software. The two most common software handshaking systems are called ETX/ACK and XON/XOFF for the code names of the signals that are used to control the data flow, each pair translating to "send me some data"/"hold on for a moment, I'm

up to my ears in data." These software signals are sent down the same pair of wires as is used by the data itself, and it's the computer's job to sort them out from the normal data flow.

If the handshaking used by the computer and the printer do not match, the result is usually that chunks of copy get lost in the shuffle and never make their way into print. If the computer does not recognize the XOFF half of the handshake, more characters than can be used will be dumped into the printer, and most of the excess will be lost. If the XON half of the handshake is lost, the printer is likely to dash through the printing of a few words, sentences, or paragraphs and stop, and then no amount of coaxing will get it started again.

By making a simple adapter cable to use in conjunction with IBM's RS-232 serial adapter and setting up your serial printer properly, you can get it to work as if it were IBM's Compact Printer without using the MODE command or worrying about any serial communication problems. The combination suggested in the box on the following page will help you communicate with serial printers faster than 120 CPS, the maximum that can be squeezed through a serial port operating at 1,200 baud, will not operate at full speed with this connection, however. (Note: the speeds claimed by most printers, including IBM's, are as much as 50 percent higher than the actual speed. Many so-called 120 CPS printers run at real speeds of about 80 CPS.)

Printing Under Computer Control

Most printers actually have their own internal computers because the instructions that they receive from the computer telling them which characters to type next bear little or no resemblance to the job to be done. All that Junior sends to its printer is a list of characters known as ASCII (American Standard Code for Information Interchange) symbols. Each ASCII symbol consists of seven digital bits (plus an eighth digit to check the first seven) that code the letters of the alphabet, numbers, other characters, and special commands. From that scant information, the printer must figure out exactly which characters to print and exactly where on the page each one goes.

Often specialized application programs, like word processors, make the printer's decoding job a bit easier by incorporating *printer driver routines,* which give the printer the precise location of each letter, how far to travel between characters, and similar information. With other programs, the printer must sort through an alphabet soup of text and break it into separate lines, arrange each line properly on the page (without wandering past the edge), and then advance to the next sheet.

Getting a Generic RS-232 Printer to Work with Junior

Make this RS-232 printer adapter:

Junior end: Female DB-25 connector pin numbers		Printer end: DB-25 connector
2	----------	3
3	----------	2
(optional)		
4 to 8		
5		20
6 to 20		
7	----------	7

Printer settings (to work without invoking MODE command):

Baud rate:	1200
Word length:	8 bits
Stop bits:	2 (however, 1 works)
Parity:	Mark
Handshake:	DTR

The baud rate, word length, and parity can be changed to whatever you want with DOS's MODE command. If you don't use the MODE command, Junior will assume that its serial port should be set to the foregoing specifications.

If you do not have a parallel printer or an internal modem attached to Junior, the printer that you use will be recognized as PRN, COM1, or LPT1 by Junior and the programs you run. LPRINT and LLIST commands in BASIC will go directly to your printer. If you have an internal modem installed, your printer will appear as PRN, COM2, or LPT1 to DOS, and normal BASIC printer commands will find it without a problem. Connecting a parallel printer to Junior, however, will make the parallel printer the default printer for both BASIC and DOS. LPT1 and PRN calls will go to the parallel printer.

When an application program sends commands to the printer or when you want to take direct control through the keyboard, you must send messages to the printer using special characters that do not print; these are called *command codes* and *escape sequences.*

Command codes are extra individual ASCII characters that have a special meaning to printers and other computer-controlled devices. Two

of the most ubiquitous command codes are *line feed* (LF) and *carriage return* (CR), which instruct printers when to roll up to the next line and when to send the print head back to the leftmost column.

An escape sequence is a series of ASCII characters (printable, nonprintable, or both) that begins with the escape (Esc) symbol, ASCII character number 27. The escape code is always first in the series and it warns the printer that the next few characters represent a command and, therefore, should not be printed. Typical escape sequence commands tell the printer to change type fonts or pitch, or to start using the graphics mode. Although the American National Standards Institute (ANSI) has published an official list of escape sequences, most printer manufacturers have adapted the standard to suit the special needs of their own machines. Seemingly no two printers share the same complete list of command codes.

Printers differ in the amount of independent thought they can handle. Some don't even know when they reach the edge of the paper. Left to their own devices, such machines would rattle through the printing of an entire encyclopedia on a single line, even if the paper is only 8 inches wide. Other printers have built-in microprocessors that may be as powerful as Junior's own and are capable of doing nearly anything with text that Junior can, including appropriately breaking text into lines, proportionally spacing each line of characters, fully justifying both the left and right margins, and supplying the proper margins at the top and bottom of each printed sheet.

Because their internal microprocessors can perfectly time every aspect of printing, the smart printers are usually the fast printers. When blessed with "logic-seeking" abilities, printers can race through lines in both directions, printing one line forward and the next in reverse so no time is wasted on carriage returns, and skip over blank spaces without stopping at every line.

Using the IBM Compact Printer

IBM's Compact Printer is easy to use and inexpensive. Its controls are simple enough—electronically speaking you get an on/off switch and a paper-feed control; mechanically, you get a paper-release lever.

If you've dealt with printers before, "paper feed" translates into "line feed." Pressing this tiny membrane switch advances the printer's paper one line at a time. The longer you hold the button down, the more paper it feeds through the printer.

The printer has a single visual indicator, the "ready" LED that glows

when the printer power switch is turned on and during the printer self-test. This indicator flashes if for some reason the print head (which actually forms the characters on the paper) becomes jammed.

Installing paper in the Compact Printer is relatively straightforward. First, raise the transparent plastic lid and pull the paper-release lever forward. Put the paper roll inside the bin. You might find it easier to do the opposite of what IBM recommends—insert the roll right side first and then push the plastic lever next to the left end of the roll to the left, then snap the lever back and let its dowel end slide into the paper roll. Alternately, do it IBM's way by using the roll to push the dowel on the left side over, then let the roll snap itself into place on the right.

Slide about a foot of paper into the slot just in front of the paper bin until it stops moving, then push the paper-release lever back. Close the cover and hold down the paper-feed button until the paper appears in the right place, behind the print head.

The left margin of the Compact Printer is not adjustable, as a typewriter's is. Rather, it is fixed at 8 millimeters, or about 5/16 inch.

You can check that the Compact Printer is operating properly by running its self-test. To run the test, hold down the paper-feed button while you turn the power on. The test will run continuously, printing line after line of characters, until you turn the power off again.

In normal operation, you should turn Junior on before you turn on the Compact Printer, to avoid wasting paper by inadvertently printing a character or two when you turn the computer on.

When you turn the power on, the Compact Printer remembers its place on the paper, and if you send a form-feed command to it from Junior, it will advance the paper to the next multiple of 11 inches from that position. To send a form feed from within BASIC, use this command:

```
LPRINT CHR$(12)
```

The other commands that you can send to the Compact Printer in the same way while BASIC or your own programs are running are as follows:

Compressed print. To switch the Compact Printer to its narrow typeface while in BASIC, send the following command:

```
LPRINT CHR$(15);
```

The compressed typeface will continue to be used until the printer is turned off or commanded to stop using compressed print. To stop com-

pressed print and return to the normal typeface, use the following command in BASIC:

```
LPRINT CHR$(18);
```

Double-width printing. To switch the Compact Printer to its double-wide typeface while in BASIC, send the following command:

```
LPRINT CHR$(14);
```

Double-width print mode will remain in effect only for the balance of the line in BASIC. A carriage return or line feed will cancel the double-width command. Double-width printing can be canceled sooner and print mode returned to normal type with the following command:

```
LPRINT CHR$(20);
```

Underlining. To start underlining mode while in BASIC, use the following command:

```
LPRINT CHR$(27);CHR$(45);CHR$(1);
```

Underlining will remain in effect until either the printer is turned off or the following command is given:

```
LPRINT CHR$(27);CHR$(45);CHR$(0);
```

With the foregoing commands, you can mix and match type styles and modes to a limited extent, underlining standard, double-width, and compressed type, and double-width/compressed type. You can also compress double-width type for a (very slightly) bolder typeface.

The Compact Printer's biggest shortcoming is that the thermal paper it uses is not permanent. All thermal papers either fade or discolor as time goes by. If you want to preserve your printouts, make a photostatic copy of them. You can prolong the life of your copies by keeping them in the dark, as exposing them to sunlight will speed fading. An image on one sheet can also accidentally be transferred to another if the two sheets are stored together.

The Graphics Printer

The IBM Graphics Printer has four indicator LEDs that help you monitor its condition. The "power" light glows when power is turned on. The "ready" light glows continuously when the printer is ready to print from the computer *or* from your command, that is, when the printer is not aware of any problems inside itself. This light flickers as the printer receives data that is to be printed. The "no paper" light goes on when the printer is nearly out of paper. As an additional warning, the printer will also beep. The "on-line" light glows when the printer is ready to receive data.

The Graphics Printer also has three direct controls that put you in immediate command of its operation. The on-line button switches the printer on and off the data line from Junior. Pressing the button once puts the printer on-line and causes it to listen for instructions from Junior arriving at its parallel port. Pressing the button again switches the printer off-line; it will then not react to any instructions that Junior sends.

Pressing the form-feed button will advance the paper in the printer to the top of the next page. The printer remembers the position the paper is in when you turn it on as the top of the page, and uses this reference for all subsequent form feeds. To reset the top of the form, turn the printer off, position the paper at the top of a sheet, and turn it on again.

Pressing the line-feed button causes the printer to advance the paper one line at a time.

The Graphics Printer's ribbon is one of the easiest to install among printers. First, turn the printer off, raise the cover, and lift the metal print scale toward the rear of the print-head chamber. Remove the new ribbon cartridge from its box. Holding the cartridge by its raised "fin," drop it into place in the printer, making sure the guide tabs on the left and right fall into position in V-shaped slots at either side of the print-head chamber. Press on the two outer edges of the cartridge to seat it firmly into place.

Once the ribbon cartridge is installed, take up slack in the ribbon by rotating counterclockwise the knob on the left side of the cartridge. Using the eraser end of a pencil, slide part of the unsupported section of the ribbon between the nose of the print head and the ribbon shield. Check to be sure the ribbon is in its proper place by carefully moving the print head back and forth. Finally, lower the cover and turn the power back on.

To load continuous-form paper into the Graphics Printer, first prepare the machine by turning the printer off, removing the cover over the print-head chamber, pulling the shiny metal print scale forward, and opening the tractor covers (lift the inside edge of each tractor and snap

each open). Take the free edge of the paper and slide it into the opening between the chrome-plated wire forms rack and plastic roller. Tilting the forms rack up will help guide the paper.

Carefully push the paper through the printer until about 3 or 4 inches appears in front of the platen. If you're changing the paper width from the previous size you used, adjust the tractors by pulling forward the gray lever on the outside edge of each tractor to unlock them, and slide them left or right to fit the width of the paper.

Push the paper down so that its sprocket holes fit neatly into the pins in the tractors. Close the tractors and lock them into place by pushing the gray levers back. Using the platen knob on the right side of the printer, roll the paper through the mechanism until the top of any single sheet coincides with the top of the print head. Push the print scale back against the paper and replace the cover. Turn the printer on to set the top of the form in its memory.

You can send commands to the Graphics Printer to change its typefaces, much as you can change typefaces with the Compact Printer. The Graphics Printer has a wider variety of print modes:

Compressed print. To switch the Graphics Printer to its narrow typeface while in BASIC, send the following command:

```
LPRINT CHR$(15);
```

The compressed typeface will continue to be used until the printer is turned off or is instructed to stop. To return to the normal typeface, use the following command in BASIC:

```
LPRINT CHR$(18);
```

Double-width printing. To switch the Graphics Printer to its double-wide typeface while in BASIC, send the following command:

```
LPRINT CHR$(27);CHR$(87);CHR$(1);
```

To cancel the double width, use the following command:

```
LPRINT CHR$(27);CHR$(87);CHR$(0);
```

The next command will initiate a *temporary* double-width print mode that will remain in effect only until the rest of the line is printed in BASIC. A carriage return or line feed will cancel the following temporary double-width command:

```
LPRINT CHR$(14);
```

Temporary double-width printing can be canceled sooner and normal type reinstated with the following command:

```
LPRINT CHR$(20);
```

Underlining. To start underlining mode while in BASIC, use the following command:

```
LPRINT CHR$(27);CHR$(45);CHR$(1);
```

Underlining will remain in effect until either the printer is turned off or the following command is given to stop it:

```
LPRINT CHR$(27);CHR$(45);CHR$(0);
```

Emphasized printing. To switch the Graphics Printer to its emphasized print mode in which the print head travels at half speed and prints two dots for every one of a normal character, which produces darker characters, send this command to the printer while in BASIC:

```
LPRINT CHR$(27);CHR$(69);
```

Emphasized print mode will remain in effect until either the printer is turned off or until the following command is sent to the printer:

```
LPRINT CHR$(27);CHR$(70);
```

Double-strike printing. To switch the Graphics Printer to its double-strike print mode in which the print head hammers out one column of dots, advances 1/216 inch and hammers out a duplicate column of dots, to produce boldface characters, send the following command to the printer while in BASIC:

```
LPRINT CHR$(27);CHR$(71);
```

Double-strike mode will continue until either the printer is turned off or the following command is sent to the printer:

```
LPRINT CHR$(27);CHR$(72);
```

Superscript printing. To print superscript characters, which are smaller than normal and raised above the main line of type (like exponents), send the following command to the Graphics Printer while in BASIC:

```
LPRINT CHR$(27);CHR$(83);CHR$(1);
```

All the characters entered after that command will be printed as superscripts until either the printer is turned off or the following command is sent:

```
LPRINT CHR$(27);CHR$(84);
```

Subscript printing. To print subscript characters, which are smaller than normal print and below the main line of type (like the numbers in chemical formulas), send the following command to the Graphics Printer while in BASIC:

```
LPRINT CHR$(27);CHR$(83);CHR$(0);
```

All the characters sent after that command will be subscripts until either the printer is turned off or the following command is sent:

```
LPRINT CHR$(27);CHR$(84);
```

As with the Compact Printer, many of the foregoing commands can be combined. Underlining and double-width printing can be used in any print mode. Obviously, normal, compressed, and double strike cannot be combined, and you cannot print superscripts and subscripts simultaneously. In these cases, the last command sent to the printer will take precedence.

Choosing a Printer

In theory, with the addition of the IBM printer adapter module or IBM's serial RS-232C adapter cable, nearly any printer can be put to work with Junior. Often, though, some programming expertise may be necessary to put your non-IBM-printer through its proper paces under Junior's control. Nevertheless, there are good reasons to use a non-IBM printer.

Many printers run faster than the limited selection of IBM printers. IBM's fastest claims 80 CPS; other manufacturers' printers are twice as fast for under $500. IBM's most affordable model, the Compact Printer, is advertised at 30 CPS but it actually pokes along at about 20.

Furthermore, a lot of machines produce type that is more readable than that of any of the IBM printers. In fact, no official IBM Personal Computer printer claims to be "letter quality." However, for as little as $400 you can add another manufacturer's letter-quality printer to your Junior and get text that is indistinguishable from that made on a typewriter.

Some printers work more quietly than the official IBM machines. Granted, the IBM Compact Printer is quiet, but it's painfully slow. The Graphics Printer is quick, but it sounds as friendly as a dentist's drill. The ideal is a printer that is quiet, fast, and high quality.

If you plan to use a variety of papers, there are better choices than an IBM printer. The more affordable IBM printers handle paper only 8½ inches wide after trimming; other machines in their price range will print on 14-inch computer paper, which is particularly useful for business applications such as electronic spreadsheets. Furthermore, the Compact Printer requires that you use rolls of special thermal paper, which quickly discolor and can be expensive.

Some printers have extra features that the official IBM printers don't have. Certain color printers will accept "vector" commands so that they can easily draw pictures, in combination with certain software packages. And some can even draw on transparencies.

On the other hand, IBM printers have special, nonstandard graphics character sets to match the special characters that Junior can display. Printers made by other manufacturers are apt to print totally unexpected characters when they receive Junior's graphics commands.

Here's a selection of printers that will happily plug into Junior—one way or another—and print away at your command.

DEC Letterprinter 100

If you're looking for a really *fast* printer and don't mind spending as much as or more than the price of your whole computer system, take a good look at DEC's Letterprinter 100.

DEC, self-proclaimed number two in the computer industry (behind IBM, of course) has had a reputation for quality, and the Letterprinter 100 follows through. Even with a plastic case and a lightweight feel for such a large machine, it seems truly solid. One novel feature—two slots for removable ROM cartridges—lets you change typefaces as easily as swapping balls on a typewriter.

High-speed printing is done at a breakneck pace, tested at 170 CPS, so fast that even at 1200 baud the Letterprinter 100 pauses occasionally to let the computer catch up with it! Letter quality (at 30 CPS) looks much like a fresh nylon ribbon had been put into a faithful old SCM typewriter.

Dynax DX-15

The Dynax is an inexpensive but versatile daisy-wheel printer with many of the features of a more expensive machine—all it lacks is speed. It creeps along at 13 CPS, even with the help of a genuine bidirectional print mechanism.

Equipped with the proper printing element, the Dynax will handle three pitches, 10, 12, or 15 characters per inch, as well as proportional spacing. Front panel membrane switches select pitch, line-spacing (single-, line-and-a-half, and double-spacing), form feed, line feed, and on and off.

The print head travels smoothly across the paper without cogging as some inexpensive daisy-wheel printers do. As it chugs along, the machine sounds amazingly like a plain old typewriter, but faster. Its beige plastic case is a good match for Junior, and its built-in tractor feed works well with almost any continuous-form paper. You can also use single sheets of plain stationery, if you like. Ribbons are loaded in quick-change, snap-in cartridges.

The on-paper quality is superb. While the lightweight mechanism is perfect for children's schoolwork and occasional business printing, this is not a machine for heavy-duty office work.

Epson MX-80 and RX-80

The Epson RX-80 and the older MX-80 are cousins of the IBM Graphics Printer—all the machines are likely made in the same factory. The RX-80 is improved in many ways over its precursor the MX-80 and the IBM, but the differences are hard-to-find design changes subtly locked inside the case. Even the mechanically similar MX-80 and the IBM have their differences, mostly in the ROM memory chip that gives each printer its identity. The IBM machine has special characters in its memory that correspond exactly to what you can put on the screen. The Epsons have their own interpretation of your nonalphanumeric musings. Of the Epson machines, the RX-80 is probably the best choice for attaching to Junior because it's the latest design and has the lowest price.

The RX-80 will dash out characters at roughly the same speed as the IBM, about 60 CPS (notwithstanding the rate of 80 per second that is claimed). To look at its work on paper, you'd swear it was done by an IBM. The price difference is the principal reason for choosing an Epson over the official IBM machine. You can probably save $100 or more by buying an Epson, though you will sacrifice some convenience.

Epson also offers a line of printers that use essentially the same mechanism but have a wider carriage that accepts 14-inch computer paper

(which is well beyond the RX-80's 10-inch limit). These machines are indicated by a -100 in their model designations.

Gorilla Banana

The Gorilla Banana is one of the most inexpensive impact dot-matrix printers available, but you'll probably find that you get what you pay for. Though it is a dot-matrix machine, the Banana's print head slowly crawls across the paper at about 20 CPS, hardly better than a cheap letter-quality printer. And the characters that the Banana makes are hardly letter quality, lacking true descenders, only marginally readable, and difficult to underline. Although the Banana does not require specially treated paper as a thermal printer does, it will not handle single sheets. The print mechanism is designed for light duty, so it has a hard time pulling fanfold paper up from a box positioned below. Finally, it's capable of making only two copies at a time, an original and one carbon. Essentially, the Banana is a minimalist printer for minimal needs.

Infoscribe 1100

This sturdy but expensive machine looks sleek, uncluttered, and elegant—and it works the way it looks. Even inside there is no mechanical clutter, just a simple, straightforward mechanism. The Infoscribe has a solid feel that implies dependability.

The case is more than just good-looking, it does an exemplary job of deadening sound, making the machine among the quietest dot-matrix printers this side of thermal machines.

The Infoscribe has three print modes: high speed, tested at 130 CPS with readability as good or better than most other nine-wire print-head machines; somewhat better "correspondence quality," dark but rough; and near letter quality, sharper than most (but that very sharpness makes the individual dots of each character stand out like the grain of a photograph).

Two type fonts are built into the Infoscribe, one with serifs and one without. You can download a third.

This is not a machine for single sheets, as it does not have a rubber platen like a typewriter. Tractor-feed paper is mandatory.

C. Itoh Prowriter II

This printer uses the same mechanism as do machines sold by Apple and other familiar manufacturers. It has a nine-wire print head to make

high quality 7 × 9 dot-matrix characters with true descenders. The Prowriter II uses a snap-in fabric ribbon cartridge that mounts on the moving print-head assembly and is smaller than most matrix printer cartridges, more like a typewriter ribbon than that of a typical printer. The ribbon even automatically reverses when it reaches its end, exactly what you'd expect from a typewriter. Its friction-feed roller mechanism will handle single sheets, roll paper, or fanfold, continuous-form paper from 4 to 10 inches wide. In operation, it is quite quiet compared to most dot-matrix printers, and reasonably quick—tested to a true 72 CPS on normal text. As non-IBM printers go, it is a sturdy, well-built machine and an excellent choice.

Juki 6100

The Juki 6100 may be the best choice for a low-speed letter-quality printer. It offers a good combination of speed and price. For the same price as printers that trace out characters at 12 per second, the Juki will race through an honest 18. Quality with a film ribbon is crisp and clear—exactly what you should expect. It handles all three standard type pitches—10, 12, and 15 per inch—using the proper print wheels. Tractor feed is optional.

The mechanism is servo-controlled and cog-free, and the print head chases back and forth, printing in both directions. Overall, the machine itself in its plastic case seems filled with helium. It doesn't weigh much. In fact, it feels like it's all plastic (but it's not)—it doesn't instill faith that it will churn out paper at several reams a day, but will more than adequately serve to type out correspondence when you need it and general light duty around the office.

Mannesmann-Tally MT-160 and MT-180

The Mannesmann-Tally MT-160 and MT-180 differ only in width—the former being narrower. Tally is an old name in the printer business, but it's not well-known. Its products are sold worldwide under other brand names.

The MT-160 and 180 are rugged, high-performance, and surprisingly affordable printers. They aren't built from the standard stamped-metal parts, but from sturdy die castings; the only part that seems flimsy is the tractor feed. Either machine will work like a charm with single sheets or continuous-form paper. And both parallel and serial interfaces are standard equipment.

Print quality is good but not fabulous. High speed looks like high

speed, made at a rapid 130 CPS in actual tests. Correspondence quality looks rough—you'll immediately recognize that it was made with a matrix printer, but it's very readable and nothing to be ashamed of.

Mannesmann-Tally Spirit-80

Although the corporate parent Mannesmann-Tally is located in Germany, and most of the company's machines show a strong Teutonic heritage, this low-cost entry hails from the Orient. In appearance, it's the white sheep of the Tally family, an elegant, wedge-shaped machine. It differs in construction, too, being made from stamped-metal pieces rather than heavy castings.

The Spirit-80's on-paper image is what you might expect from an inexpensive dot-matrix machine—readable but nothing you'd mistake for the work of a typewriter. One unusual feature of the Spirit-80 is its film ribbon (in a cartridge) that prints dots darker and more defined than cloth ribbons do. Nevertheless, the characters still show their matrix heritage. Although it packs the standard six lines per inch, its printed characters *look* a bit smaller. It's reasonably quick, generating nearly 60 CPS of normal text, and it's about as noisy as you'd expect a small dot-matrix printer to be.

NEC PC-8023A-C

The NEC PC-8023A-C claims to be neither outstandingly fast nor letter quality. Rather, it is a small, sturdy printer introduced to compete with the IBM Graphics Printer to do a reasonable job on everyday chores.

Print quality is typical of a single-pass 9 × 7 dot-matrix machine—characters are ragged with true descenders. Underlining, however, is rather dissatisfying, with the bottom line blotting out the lowest line of descending characters.

Although the NEC will handle single sheets and continuous-form paper, you'll have to disable the "paper-out" sensor to use the last few lines on individual sheets.

Though the NEC is not particularly noisy *nor* particularly quiet, its sound is different and some find it unusually annoying.

Okidata ML-80

Among the less expensive alternatives to the IBM Compact Printer is the Okidata ML-80. After several price cuts, this model has appeared on the market at $150 or less.

The ML-80 is quicker than the Compact Printer. Eighty CPS is claimed, though its rate is closer to a still-respectable 50 CPS. And it's noisier than most dot-matrix impact machines. Among other things, sound deadening, bidirectional printing, and true descending characters were left out when the price was sliced.

Although the ML-80 has a parallel input port, it is not compatible with most standard IBM-to-Centronics cables, and using an unmodified Centronics cable may damage the printer, your Junior, or both. Fortunately, many standard cables can be adapted easily for proper operation by disconnecting and taping up the wires connected to pins 33 and 36 of the plug at the printer end of the cable. Because design changes may be made (and not all "standard" Centronics cables are really standard), check with your computer dealer about the need for and suitability of this procedure before attempting it.

Okidata ML-92 and ML-93

As replacements for the IBM Graphics Printer, both of these machines deserve consideration because of Okidata's understanding of what complete IBM compatibility comprises. Besides this competent duo of printers, Okidata also sells IBM *Plug 'n Play* kits, everything you need to make an ML-92 or -93 act exactly like the official IBM printer. The kit includes ROM chips that you slide into sockets inside either machine to make it recognize and print the entire IBM extended character set. Another box in the kit includes the proper adapter cable to plug into Junior's parallel printer adapter.

Both the ML-92 and ML-93 claim print speeds of 160 CPS, but you should expect to see normal text print at about 120 CPS in high-speed mode, and at about 30 CPS in a reasonably good-quality "correspondence" mode. Besides the different typefaces offered by the IBM machine, these two also feature a proportionally spaced type font. The ML-92 is an 80-column, 9½-inch paper printer; the ML-93 handles 16-inch paper and 136 columns.

Quadram Quadjet

This new ink-jet printer from Quadram is one of the least expensive color image makers available, and the only non-IBM printer that now works with IBM's PC*jr* *ColorPaint* program. That makes it a top contender to connect to your Junior. While the Quadjet doesn't use all the IBM extended character set, a special program (included with the printer) lets you copy any of your graphics screens to paper.

Ink-jet printing means that it's quiet, and the seven-color palette (black, red, green, yellow, cyan, blue, and magenta) available from its four jets almost exactly matches the capabilities of *ColorPaint,* so you can get inexpensive but impressive hardcopy of your on-screen drawings. Colors are bright and saturated.

The Quadjet handles text, as well, with about average dot-matrix quality, dark but a bit fuzzy, for your choice of standard width or enlarged characters in normal or boldface. Its 37-character-per-second claimed speed, however, may disappoint you, so you may want to use it just for special, colorful projects—like business charts—and use another machine when you need to churn out reams of data and text.

The ink for the various jets is contained inside disposable cartridges, so there's little muss and bother when the supply runs out. Maximum paper width is 8½ inches, and because the machine uses a rolling platen, there's no trouble with feeding individual sheets or continuous paper rolls. You'll want to use its special paper (or transparency sheets) because normal bond absorbs its thin ink like a paper towel, blurring the image.

Qume Sprint 11

High speed among letter-quality printers means about 40 CPS. The Qume Sprint 11 claims that goal and comes very close at a price that is reasonably affordable for an American-built, heavy-duty machine (it will print all day and not even get warm). The base is a solid metal casting, the top heavy plastic. The platen knob is noticeably weak, however.

As you would expect for a machine that costs somewhat more than a fully equipped Junior, the actual printing mechanism is bidirectional and fully programmable for nearly any line or character spacing you want. Readily accessible DIP switches allow you to preset the most common pitches (10, 12, and 15 per inch and proportional spacing) and page lengths so that you don't have to be a software hacker to take control.

Single sheets up to 15 inches wide are no problem, but the tractor is optional. The interface, too, is optional, and you have your choice of serial, Centronics parallel, or an IBM interface (parallel with the proper connector to plug right into Junior's parallel printer adapter—you won't even need a cable).

Perhaps the biggest advantage of a Qume is its wide use in business and industry and the ready availability of supplies like print wheels and ribbon cartridges. Often both are available as house brands for a fraction of the cost of official products. The Qume also rates at the top for print quality and versatility.

Smith-Corona TP-1

First among inexpensive daisy-wheel printers is the TP-1, from a company well-known for its inexpensive typewriters. The TP-1 shows its typewriter heritage through and through. Just like a typewriter, it cogs to each character position. Skip a dozen spaces, and the TP-1 stops at each one anyhow, albeit momentarily. At the end of each line the print head zips back to the left margin exactly as if someone had slammed a typing hand against the carriage of an old manual typewriter.

Because of the fixed mechanical character cogging, the TP-1 comes permanently preset with your favorite pitch, 10 or 12 characters per inch. You adjust line spacing exactly as you would on a typewriter, by moving small levers near the end of the platen roller. The rubber platen does a good job on single sheets. Continuous forms are another matter—it was more than a year after the TP-1 was introduced before a tractor feed became generally available.

You can buy a TP-1 with either a parallel or serial interface. Although the serial interface can be adjusted to work with Junior as a pseudo Compact Printer, accessing the DIP switch that sets the baud rate requires a good deal of disassembly.

Although the actual printing is slow and noisy, the on-paper print quality produced by the TP-1 is excellent. The low price of the machine makes all of its minor problems endurable, if not bearable.

Star Micronics Gemini-10X

From a standpoint of speed versus price, the Gemini-10X is a top choice, putting nearly 80 characters on paper every second. It's sturdily built and makes a sharp, dark dot-matrix impression. But it's noisy, too—one of the noisiest small dot-matrix machines you're likely to encounter.

The biggest disadvantage of the Gemini is that it does not use ribbon cartridges, so ribbon changing can be so messy you'll starting seeing purple, particularly on your fingertips, and its plastic lid snaps down rather than being hinged. Perhaps it might not seem like a major point, but inevitably the snaps snap off, usually the first time you forget and try to lift the cover.

On the positive side, the Gemini will wrap either single sheets or continuous-form paper around its rubber platen and offers more special printing features such as different type sizes and styles than most other machines in its price class. If you're willing to forgo a few minor conveniences in favor of a good buy, you'll probably find your match in Gemini. (A similar model with a wider carriage is available as the 15X.)

Star Micronics STX-80

In concept and execution the STX-80 seems like a dead ringer for the IBM Compact Printer. Both are quiet thermal machines that are happiest rolling through a spool of paper in a special rear chamber with a transparent cover. Like the IBM, the STX-80 prints quite slowly, about 27 CPS, and it is eerily quiet. The thermal-made matrix dots actually look more sharply defined than most impact dots, and the printed characters appear darker and denser than those made with most inexpensive impact matrix printers.

But the STX-80 lacks the advantage of an official IBM ROM and may produce characters different from what you expect if you venture away from the alphabet and numbers. Too, since the standard STX-80 uses a parallel port, it will cost you more to bring to life than the IBM machine simply because of the additional money you must spend on the adapter module.

Toshiba P-1350

With a twenty-four-wire print head, rather than the usual nine or so, you might expect this Toshiba to be a miracle worker. Unfortunately, it's rather ordinary. Neither large nor small, it doesn't print rainbows or high-quality letters, and it won't run through paper faster than it can be made. Nevertheless, it does everything a printer should do without a lot of fanfare and bother. It prints reasonably quickly (119 CPS in a test of its high-speed mode), and it doesn't make you pay for features that you'll never need.

Its letter-quality print looks like the typing of a thirty-year-old Remington— black but a little smudgy. The additional print wires make some characters look a bit odd in high-speed mode, but for the most part, the output is as good or better than ordinary matrix renderings.

Single-sheet operation is easy; a bidirectional tractor for continuous forms and a sheet feeder for reams of paper are both optional.

Communications

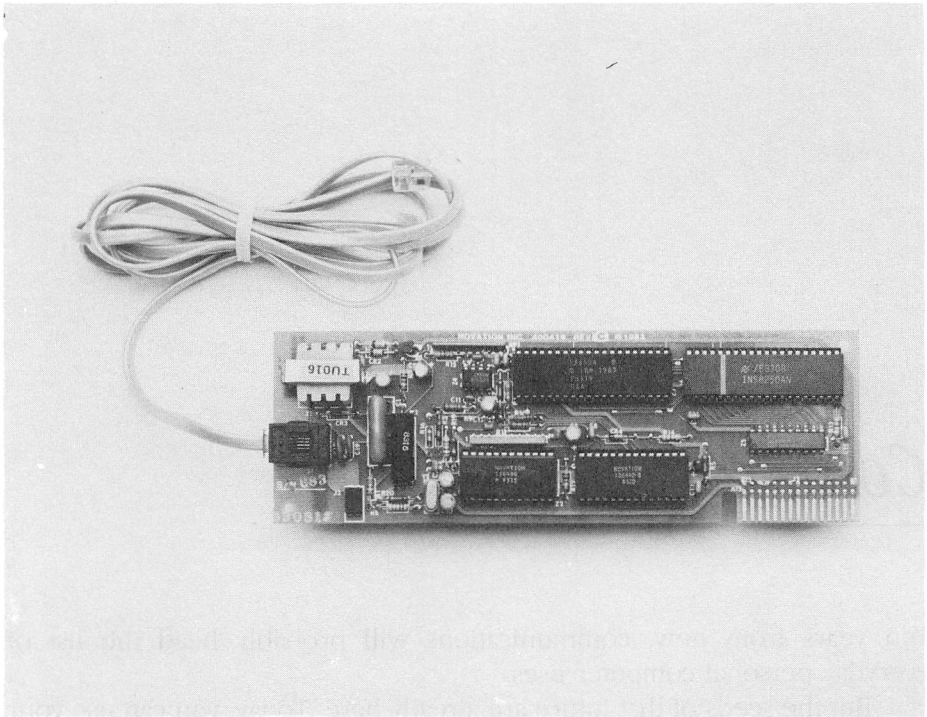
Ten years from now, communications will probably head the list of everyday personal computer uses.

But the seeds of that future are already here. Today, you can use your Junior to send messages across the country at the speed of light. You can shop or pay bills with Junior. You can browse through the latest newspapers, magazines, and even some books with Junior. And with Junior you can research almost any subject faster and in greater depth than ever before.

You can connect your Junior to a telephone to perform any of these tasks quickly and easily. The connection is so natural that IBM has made it internal and intrinsic to Junior's makeup. The modem, the device necessary for a computer to use telephone wires, fits right inside Junior, unlike the clumsy add-on devices that must be used with most other computers.

Using the modem is simply a matter of mastering software. Some programs, like IBM's official *Personal Communications Manager,* let you operate your modem with a menu, from which you select your preferred function. Other programs are command-driven, such as the much less expensive *PC-Talk.* To use them you press keys to issue commands, relying mostly on your memory to be sure of using the right one.

Either way, the automatic dialing of your phone under modem control is only a couple of keystrokes away. In a few seconds your Junior is a complete data terminal, able to send commands and receive information from around the world.



Internal 300-baud modem allows PC*jr* users to communicate with bulletin boards and other users over standard telephone lines.

About Modems

Modem is a contraction of the words *mod*ulator/*dem*odulator, a functional description of the device. A modem converts your Junior's digital signals into tones that can be sent over normal telephone lines, much as your Junior itself converts digital signals into tones that can be recorded on cassettes. The modem must handle the conversion predictably, reliably, and affordably.

The problem faced by the modem does not arise from the characteristics of telephone wires themselves. Junior's digital codes can reside in a variety of media. After all, the wires inside your Junior do a yeoman job of digital signal handling, and similar wires were used for long distance *digital* communications long before the telephone was invented—Morse's first telegraph was essentially a digital communications system. When you use your Junior's serial port, a digital code is sent down a pair of wires to printer or plotter, bereft of any modem's benefits.

The reason you need a modem is that the signals may have to travel a long distance, which they don't do well. They won't flow through any medium—air, ether, or wires—forever. With each quantum step a communication signal takes, it gets weaker and disperses on its way to entropy.

The telephone company's solution to signal fading is the amplifier, an analogue signal booster. Amplifiers are spaced periodically along telephone circuits (every several dozen miles or more) to increase the strength of the signals as they weaken. However, because the telephone system was designed to use only voice signals, the telephone amplifiers were built to handle voice frequencies only, and even then only signals of 300 hertz to about 3,000 hertz, the range of "intelligibility."

Squeezing high-speed digital signals through a telephone amplifier is like trying to put a gallon of sauerkraut into a quart jar. Everything just won't fit, and if you don't do some fancy tricks, you'll have a mess on your hands.

When the telephone company filters away all the signals below and above the 300- to 3,000-hertz voice range, it distorts the sharp transitions between the on-and-off digital states. The limited bandwidth of the telephone system therefore requires that the digital pulses conform to a certain (and not necessarily desirable) rate.

Other characteristics of telephone wires make them even more inhospitable to digital signals. Noise on the phone line often occurs in the form of sharp pulses that are exactly like digital pulses. Your computer can mistake a noise pulse for data, resulting in errors. Echoes of digital signals bouncing back and forth down the wires (and through space to satellites) further confuse computers.

To avoid problems, modems convert digital signals into analogue tones, which are happy to traverse the telephone system. A tone can carry digital information because the digital code is entirely independent of its medium, and it doesn't even require electricity be used to make or transmit it. Digital bits can correspond to anything that exists in two states—switches that are on or off, red or black checkers, or tones of two different frequencies. One tone can be used to represent a 1 digital bit, and another tone could represent a digital 0. The data bits in the digital code would then change the tones or, as radio and scientific folk would say, modulate their frequency. When a pair of frequencies within the working range of the telephone system is used, the modulated tones can transport digital data over normal telephone circuits for vast distances without suffering from the agony of analogue signal processing.

Toning Up

Although simple in concept, in actual application the modem's use of tones is more complex. With telephones, it is better to give *and* receive than to do either one alone. You talk and listen on the telephone, often at the very same time. So do computers. Hence, modem conversations generally go in two directions—every modem must be as good a listener as it is a speaker. However, if a modem listens to the same tone pair that it sends, it might become enamored of its own voice and think it was receiving worthwhile data coming from the far end of the telephone wire. Most modems, therefore, use two pairs of tones, one pair for signals they send and another pair for the signals that they listen to.

Strict one-way communications, like the signals sent by a television station or radio transmitter, are called *simplex,* because they are the simplest way of communicating.

Two-way communications on a single channel or telephone line are called *duplex.* If information is alternately sent and received over the same channel the way data is written and read from a cassette recorder, the scheme is called *half duplex.* Each mode of communicating, sending and receiving, uses the communications channel essentially half the time.

Schemes in which two signals move in opposite directions simultaneously—one sending and one receiving—are termed *full duplex.* Computer communications are therefore generally full duplex.

There's another "plex" called *echoplex,* which is often used by microcomputer terminals to talk to mainframe computers. In echoplex, the terminal sends a data character to the computer, which immediately relays a duplicate copy of the signal back to the terminal, "echoing" it. The terminal displays the echoed character on its screen. If a signal becomes garbled during the two-way trip, or if it disappears entirely, the error or omission will appear on the screen of the terminal, alerting whoever might be operating it of the transgression of the data.

One ramification of using full-duplex and echoplex schemes for computer communications is that the modem mechanism must necessarily be more complex than is needed for use with a cassette recorder. Whereas one set of tones is enough to communicate one way at a time to or from the cassette, to communicate in two directions at the same time over one telephone line some means must be used to separate the coming signals from the going signals, otherwise your computer might listen to its own words and react to them, often with undesired (and

repetitive) consequences. Most modems, therefore, use two sets of tones, one set for receiving and one for sending.

Standardization is another problem. Obviously, if modems are to understand one another, they must all use the same tones. However, if all modems sent data on the same frequency and received it on another set frequency, they would never be able to communicate. One modem must listen to the frequency that another talks on, and vice versa.

To keep things reasonably organized, most modems have two modes—*originate* and *answer*. To properly communicate, one modem must be in originate mode and the other in answer. In the originate mode, most modems send data out on tones centered at a frequency of 1,170 hertz and listen to data coming in at a frequency centered at 2,125 hertz. Conveniently, modems in answer mode listen to tones centered at 1,170 hertz and send frequencies centered at 2,125 hertz.

At low communication speeds, the modem's manipulation of tones to transmit data is easy to understand. It receives data from your computer as a serial string of digital pulses, essentially one long line of signals that are either on or off. Whenever the modem detects an off signal in the digital stream, it sends a frequency lower than the nominal 1,170 or 2,125 hertz standard. When it hears an on signal, it sends a frequency higher than the nominal one. Because the frequency of the tones sent out by the modem is constantly changing to reflect the data, this transmission scheme is called *frequency-shift keying* ("keying" harks back to the telegraph when an electric current was actually turned on or off by operating the telegraph key).

Frequency-shift keying allows modems to send and receive data over normal telephone lines at speeds up to 300 bits per second (baud), about 30 characters per second. Higher speeds are not possible with simple frequency-shift keying because of the transmission qualities of telephone lines. However, faster data speeds can be achieved using different modulation schemes (like *phase-shift modulation*), but the equipment required is more complex and expensive and more transmission errors are inevitable. The IBM PC*jr* internal modem operates at a data rate of 300 bps.

Faster Talking

High-speed modems can be connected to Junior either through the serial port (the *S* connector on its back panel) or by buying an outside supplier's high-speed modem that will plug into Junior's internal modem slot. Many

of the modems that fit Junior's serial port operate at 1,200 bps as well as at 300 bps (and lower speeds). The advantage of higher speed is most obvious when communicating long distances—1,200 bps transfers data four times faster, which means that long-distance calls will be only one-fourth as long.

Even faster modems are possible; modems that communicate at data rates of 9,600 bps are commercially available. Not only are such superfast modems even more complex and expensive than the 1,200 bps variety, they are worthless to Junior—which cannot operate that fast!

Junior has its own speed limit. Its internal microprocessor must divide its time among a variety of chores: it listens to the keyboard and runs the video display at the same time that it processes data and deciphers what's going on at the serial port. Consequently, its serial port has a speed limit of 4,800 bps in sending data and 1,200 bps in receiving it. Further, it appears that when a program puts Junior to work, it might occasionally miss a character or two. Although Junior will have no problem receiving and displaying data at 1,200 bps, it may have trouble manipulating that data at the same time (in a buffer, perhaps).

Bits and Parity

Besides the speed at which the serial port operates, different communications systems and some software packages handle digital data differently. Some concentrate only on printable characters—letters of the alphabet and numbers—and use a 7-bit digital code to handle that information. Other systems and software handle data in nonalphanumeric formats—like for Junior's extended character set—and use an 8-bit digital code. By using a single additional bit, twice as many characters can be coded.

An additional data bit, called a *parity-checking* or just *parity* bit, is often added to the serial code to help detect errors. To assure against transmission errors, some computer systems count the number of on data bits in each 7- or 8-bit word of code they send. They then add another bit to the digital word to make the total either an even or an odd number. System parity is described as odd if the on bits are made to add up to an odd number and even if they are made to add up to an even number.

If something happens to the digital signal to accidentally change a single bit in any data word, the parity of odd or even that has been set will not properly match the total of the on bits in the data word, and the receiving computer will know that there is a problem. A parity bit alone,

however, does not provide sufficient information in itself to allow error correction. Usually, the word in which the error occurs (or the block of data containing it) is retransmitted until it is properly received.

There are three special kinds of parity: *mark, space,* and "*no.*" Mark parity always adds an extra on bit no matter what the parity bit is. Space parity always adds an extra off bit, also disregarding the checksum. No parity systems don't bother checking parity.

The sending computer also adds *further* bits, called *stop bits,* to the word being transmitted to indicate that it has finished with the data bits of one word. Commonly, one or two stop bits are used in each digital word.

These conditions—speed, word length, parity, and number of stop bits—are called the communications parameters. When you use a modem, be certain that you set Junior's serial port communications parameters the same as those of the computer system you try to talk to. Different communications systems may use different settings. For the most part, it won't matter exactly what the settings are; the important consideration is that the settings of your modem match those at the other end. Two of the most common settings of serial communications parameters are shown in the following:

	Text & Graphics	*Text Only*
Baud rate	300	300
Word length	8	7
Stop bits	1	1
Parity	none	even
Duplex	full	full

(Use the left set for transferring *WordStar* files and those with graphics, and the right settings for communicating with most data bases.)

These settings, the parameters that Junior uses for serial communication, are set either through the communications software that you're using or with the MODE command in DOS.

Modem Extras

Most modems do more than just translate data. Many can dial the telephone for you by mimicking the signal a telephone makes when it is dialed, a featured called "auto-dialing." Auto-dialing modems may use

either the dialing method equivalent to that of rotary dial telephones, called pulse dialing, or the scheme used by pushbutton phones, called tone dialing. Most auto-dialing modems are capable of both functions and can be switched (perhaps by software command) to match the dialing scheme used by your telephone system. More sophisticated modems listen for other than data tones on your telephone lines. An "auto-answer" modem listens for the signal that rings the bell of your telephone, and will automatically answer the phone (and connect itself up) for you. And the most intelligent modems can listen to the telephone line after they dial and determine when a connection has been made and whether the phone at the other end is ringing or giving out a busy signal.

Although modems can handle all of these chores, they cannot do any of them alone. They need expert assistance and guidance to tell them what numbers to call and how and just what to do next. Such instructions are sent from Junior to the modem in the form of control codes, special digital patterns that represent the instructions electrically. The control codes are another language that both your computer's software and the modem must understand to work properly together.

Selecting a Modem

The biggest influence on what modem you choose is the software that you have available to you. It must match the command codes used by the modem for changing modes, dialing, and all of its other functions. While no organization has defined a standard set of control codes for modems, many modems made by lesser-known companies use the same codes as those of the major manufacturers; they are said to "emulate" the other modems. Probably the most emulated of modems, because it has popular control sequences, is the Hayes Smartmodem.

There are several important things to check before you buy your modem. You must be sure that the modem you choose is compatible with your software, that your communications program can be modified or installed to adapt to the control codes of the modem that you choose, or that your modem comes with its own software. When you get "free" software with the modem, you should be certain that the software will run on Junior and that the software does everything that you require. Both the software *and* the modem itself must be able to handle all the modem features that you intend to use. Should you select a 1200 baud modem, it's important to be sure the software that you choose will reliably work with Junior at that speed.

Deciding on whether you really need a 1,200 bps modem is mostly a matter of trade-offs. If you only plan to chat with local friends or use a WATS line for long-distance chattering with a system that does not charge for its time—for instance, communicating with your home office—there's no good reason to go faster than 300 bps. Should you deal regularly over the long-distance lines, a modem that works at 1,200 bps can lop 75 percent off your monthly telephone bill. Even though most services that you can dial charge more for 1,200 bps services than for those at 300 bps, the price difference won't be a factor of four—and that makes 1,200 bps the better buy.

Junior gives you a choice of two ways of connecting modems. Although there is a slight difference in convenience, the more important difference between the schemes is the price. Internal modems can be slid into an expansion slot built right into the computer; just pry off the lid and slide in the circuit board. When you start attaching an external modem to the jack on Junior's back panel, suddenly some hidden costs become apparent. In most cases you'll need to buy an IBM RS-232 serial connector and a serial connecting cable. (You'll want a cable with a male DB-25 connector at one end and a female DB-25 connector at the other.)

An internal modem does not need a case or housing of its own, or its own power supply, so they should be less expensive to build and buy. Nor will you have an extra box cluttering your desk. Moreover, when you buy an internal modem, you won't need either the IBM serial connector or a connecting cable between computer and modem—and that will save you about $50 in supplies in hooking everything up.

Internal modems do have shortcomings, however. Although they don't need their own power supplies, they suck valuable watts from Junior's already strained 33. Further, an internal modem will work only in Junior, so you cannot use it with another computer. The internal modem will also generate extra heat inside your computer and it won't show you any blinking lights to reassure you that the product is properly working.

External modems are more versatile than internal models, as they can be used with nearly any computer in addition to Junior. They offer a bigger selection of products and features to choose from. The choice is yours—the data that you send and receive won't care which type, or even which modem, you use.

Junior's Own Modem

IBM's modem for Junior is a good product that handles most functions with only a few shortcomings. It is limited by its 300 bps speed, which means that it is quite slow. On long-distance lines it can be costly to

use, especially if you are hooked up to the modem for extended periods of time.

Simplicity is the key reason for considering IBM's modem. You know that it will work and that it will fit properly in its socket and cause no disruption elsewhere in the system. You know that you can get software to run the modem, from the same IBM dealer who sold it to you. It has no switches to adjust and has no trouble running other software.

All you do to hook up the modem is plug your telephone line into the jack on the back of Junior. First, disconnect the telephone from its modular wall jack, and plug the wire IBM graciously gives you with the modem into the wall jack and into Junior. Simple. Impossible to do wrong. No loose ends.

Oh? What do you do with the wire leading from your telephone to nowhere? How do you bring your telephone back to life? How do you and your computer share the same telephone line?

You need an adapter for your telephone jack, a little plastic module that slips into the jack and doubles it, giving you a place to plug in both your modem and your telephone. Alternatively, you can easily install an extra telephone jack. Simply connect the screws on the connector block inside of the new jack with the corresponding terminals on any existing (and working) telephone jack in your home or office. To avoid any problems, connect all four wires (though only two may be used by the telephone) color for color on each end: red to red, green to green, yellow to yellow, and black to black.

If you don't use communications software to set the modem to auto-answer mode, you can leave both the computer and telephone connected to the line all the time. (Don't forget to notify the phone company that you've connected your computer to their wires.)

Unless you tell it to, Junior will never interfere with your telephone conversations. And you should not interfere with Junior's conversations, either. If you pick up the telephone while it's transmitting or receiving, you'll confuse the signals. If you plan to use your modem for serious work, such as doing research on databases, be very careful that nothing interferes with Junior's use of the telephone line.

One of the more common phone-line-related problems is interference from the phone company's optional "Call-waiting" service—the signal that you hear while you're using the phone that tells you you've got another call. Call-waiting's warning signal interferes with Junior's digital dialogue and can cause errors, or it may even disconnect the call.

Bad connections are an even worse problem because they can attack any line at any time, totally unpredictably. A good line can go bad in the

middle of a call, reducing data communication to mental consternation. The only solution is to abandon the call and try again, perhaps waiting a few minutes for a better line to be available.

Other Modems

If you're the impatient type, you will quickly tire of 300-bps communications. Odds are you'll be able to read faster than your display fills, making the chore of wandering from menu to menu on some remote computer system a tedious if not painful experience. With Junior's internal modem you cannot go faster than 300 bps, but you can quadruple your speed with another manufacturer's modem.

At least two companies—Microcom (1400A Providence Highway, Norwood, MA 02062) and Tecmar (6225 Cochran Road, Solon, OH 44139)—sell high-speed, 1,200-bps modem designs that slide right into Junior's internal dedicated modem slot. Installation is as quick and easy as with the standard IBM product.

Most standard modems made by other manufacturers connect to Junior's serial port, *S* on the back panel. With IBM's short RS-232 serial interface adapter, you can plug almost any modem directly into Junior if you can maneuver it around in a convenient place behind the computer. IBM is hardly generous with the cable that it supplies with the adapter; it's all of 4 inches long. If you have more sense than the IBM engineers, you'll probably want to add an extension cable to the standard adapter so you can put your modem in a convenient position, like under your telephone where it belongs.

Because Junior's serial port is set up as Data Terminal Equipment, you will not need a modem cable (or "null modem") cable to adapt your computer to most modems. A plain extension cable is required, whose pin assignments at both ends should be the same.

If you do succumb to the lure of high speed, keep in mind that as modem speeds increase, the incidence of transmission errors also increases. One bad bit makes errors in several bytes. Be sure that the software that you use for important transmission at high speed has some form of error checking or error correction—and use it. The most common error-checking scheme is called *Xmodem*, and is used by a wide variety of programs. IBM's *Personal Communications Manager* uses a more advanced error-checking scheme developed by Microcom Corporation.

Communications Software

A modem is not enough to get you communicating with other computers over the telephone lines. You need the right program to make the modem work. Junior can run a variety of programs that will turn it into your personal computer communications center. Except for the first, the following programs all require a disk drive and 128K of memory.

TERM. Perhaps the cheapest and easiest to use communications program for Junior comes built into cartridge BASIC as the direct command TERM. Once you have *cartridge* BASIC running, just type in the magic word TERM and press the Enter key. Junior will then automatically switch to a function called "terminal emulation." As a terminal emulator, you can use Junior's internal modem or serial port to communicate with other computer systems.

The first screen you see when you enter the terminal emulator is a menu of communications parameters. Using it, you can change baud rates and parity without bothering with MODE. To change any of the settings, just type in the number of the setting you want to change, a comma, and the new value you want. Then press the Enter key. Any value you give after the function number 6 will be sent to the modem when you transfer from the menu to actual terminal mode. (Check your modem's manual to find specific instructions. For instance, to automatically dial a number with Junior's internal modem, type in 6,DIAL 555-1234, then press the Enter key.)

To switch to actual terminal mode, just press Fn then 1. Junior will send the instructions listed after number 6 on the menu to the modem and start to communicate.

The BASIC terminal emulator is rather minimal. You cannot save any of the incoming information to disk or send a disk file to another computer using it. Once the printing scrolls off the screen it's gone. (You can, however, use the Fn then Prt Sc key combination to send the data to your printer.) In other words, it's all you need to use a service like CompuServe or get stock quotes, but you can't use it to send programs and files to friends.

Asynchronous Communications Support is IBM's cheap, basic program. It won't auto-dial or otherwise automatically control a modem, but it will turn your computer into a terminal that can talk with other computer systems. From the program's menu, you choose the communications

parameters and the kind of terminal you want your computer to behave like and then you're pretty much on your own. To use the program effectively, you should memorize the manual that comes with your modem so that you get used to sending it commands directly. IBM does not recommend this program for Junior, but it does work. (Available through IBM)

CrossTalk XVI is a program originally written for the IBM PC that will run on Junior. You'll need an 80-column monitor, but don't expect a colorful display.

The program itself is basically command-oriented, which means that you give it commands rather than selecting from a menu. Although that requires you to learn and remember what to do and when, command orientation makes *CrossTalk* work quickly. *CrossTalk* does have a "help" facility that can be invoked at any time to describe the function of each of its commands, but you must have a general idea what the command is about before you can even ask for help!

CrossTalk has some very valuable features in addition to operating all the functions of a modem (auto-dial, auto-answer, file capturing and sending, and so on). It's complete enough that you can use it as an automatic dialer for your telephone if you want. However, it will not automatically handle the various intricacies of electronic mail or some of the other advanced convenience features of *Personal Communications Manager* (described later).

You can use *CrossTalk* to send data directly through Junior's serial port or to operate a computer system by remote control. Other versions of *CrossTalk* are available for non-IBM, non-DOS computers, which let you transfer files between different computer systems without a great deal of hassle using their serial ports—once you have the right version of *CrossTalk* running on each machine. *CrossTalk* even lets you operate your computer from another machine by remote control.

However, *CrossTalk* favors several modems other than Junior's, including the Hayes Smartmodem and Novation Cat series, so you'll have to give the program special instructions to use your internal modem. (Microstuf, Inc., 1845 The Exchange, Suite 205, Atlanta, GA 30339)

Personal Communications Manager (PCM) is Junior's official IBM communications software (written by Microcom), so it operates with the fewest problems—almost none if you stick to IBM's internal modem, though it will happily operate other modems.

Although the program will run Junior's internal modem at 110 and 300 bps, it can also be used at higher speeds with other products, including those modems made by Hayes, Novation, and Microcom (the company that wrote the program) at 1,200 bps. It seems to have trouble capturing data when running on Junior at 1,200 bps, however.

Above all, *PCM* is designed to be easy to use. It's menu-oriented, so you select what you want to do from a list of on-screen choices. Although that means you may have to wend your way through several layers of menus just to make a call, because many menu choices let you dial a whole series of numbers such as long-distance service, ID number, phone number, and log-on sequence in one fell swoop, it can still be quick to use.

Although convenience and the ability to be operated by people who have trouble with sophisticated electronic tasks are *PCM*'s strengths, it also incorporates one of the most sophisticated communications protocols used in personal computer systems. This means that it can detect and correct transmission errors better than most programs, providing both ends of the call use the same protocol and are running *PCM.*

PCM was written especially for Junior and it understands all of the shortcomings and strengths of the computer, like the lack of direct memory access to its disk drive resulting in its inability to "listen" and "remember" at the same time. The program is smart enough to prevent you from losing any information (at 300 bps) should you try to put material you receive from the modem link directly on disk, a task other communications programs seem to have problems with. (Available through IBM)

PC-Talk III has a big advantage that has made it a standard among IBM PC owners—it costs nothing if you copy it from a friend but its author asks for a $35 donation if you are pleased with the program. Although many programs are copied illegally, *PC-Talk* is *legal* to copy; the author of the copyrighted program (it is *not* in the public domain) has given blanket permission to anyone to copy it for personal use.

To run *PC-Talk* on Junior, you'll need an 80-column monitor. The program lets you change the on-screen display to any normal Junior color combination that pleases your eye.

PC-Talk is a good, general-purpose, command-driven communications program that works reasonably well with Junior. Its dialing directory holds sixty phone numbers, and the program can convert characters that it receives to make files more compatible with your other software (such as *WordStar*). It supports most of the automatic features that you'd expect, like auto-dialing and auto-answering, and lets you save and transmit files

from disk, except that it may inadvertently lose characters if you try to route the information that you receive directly to disk. (Presumably this will be fixed by the time Junior becomes popular.)

PC-Talk prefers to work with a Hayes modem. To use it with Junior's internal modem, you'll have to take a couple of minutes to change some of the commands—an easy, menu-choice procedure. The program is normally supplied with both compiled (fast-running) and interpreted BASIC versions, and complete documentation on one disk, which you must supply. Printing out the manual fills about seventy full pages.

At times *PC-Talk* is unforgiving, but it does work, and more people swear by it than swear at it. Its price alone may have made it the most popular communications program for the IBM PC. (Freeware, PO Box 862, Tiburon, CA 94920)

Number, Please: Whom Do You Talk To?

Once you've got your modem installed, your communications software up and running, and your telephone bill paid, you face the vast wilderness. What number should you tell your modem to call for you? What waits at the other end of the telephone wire for the frequency-shift-keyed digits from your Junior? Now that you have a voice, will anybody listen—and will anyone talk to you?

There are a lot of computers out there that want to listen, and share data with you. Some share their bytes only for a price, while others will say hello and give away programs and the like for free.

The free services are for the most part electronic bulletin boards operated by dedicated computer hobbyists intent on advancing their art, or just showing their stuff to the outside world. They graciously connect their machines to their phone lines like hopeful fishermen casting for trout, never knowing what will find their lines. Bulletin boards are your modem's greatest resource. But the economics of running a bulletin board can be overwhelming and many appear and then disappear more quickly than word travels about their existence. Published lists are quickly outdated. The best way to track down a bulletin board is through users' groups and other electronic sources.

The commercial information services, however, show no sign of going away. In fact, electronic data is a fledgling industry looking at a nearly unbounded future. Already you can dial up the answer to nearly any question that you have—medical, legal, business, or whatever—page through next month's magazines, and get the latest quote on bonds, stocks, and cattle.

Moreover, as the computer becomes more involved in communicating, other services are turning their ears from words to bytes and eyes from paper to the monitor screen. Some people already prefer their mail to be electronically delivered through popular services like the Source or CompuServe. Shopping and bill paying by computer are becoming commonplace.

The following list contains the most popular general-interest commercial computer information services:

BRS
1200 Route 7
Latham, NY 12110
(800) 833-4707

CompuServe
CompuServe Computer Information Service
5000 Arlington Center Blvd.
Columbus, OH 43220
(800) 848-8199
(614) 457-8650

Delphi
General Videotex Corp.
3 Blackstone St.
Cambridge, MA 02139
(800) 544-4005
(617) 491-3393

Dialog
Dialog Information Service, Inc.
3460 Hillview Ave.
Palo Alto, CA 94304
(415) 858-2700

Dow Jones News/Retrieval Service
Dow Jones and Company, Inc.
PO Box 300
Princeton, NJ 08540
(800) 257-5114
(609) 452-1511

Nexis
Mead Data Central
9333 Springsboro Pike
PO Box 933
Dayton, OH 45401
(513) 859-1611

Orbit
SDC Information Service
2500 Colorado Ave.
Santa Monica, CA 90406
(800) 421-7229

The Source
Source Telecomputing, Inc.
1616 Anderson Road
McLean, VA 22102
(703) 734-7500

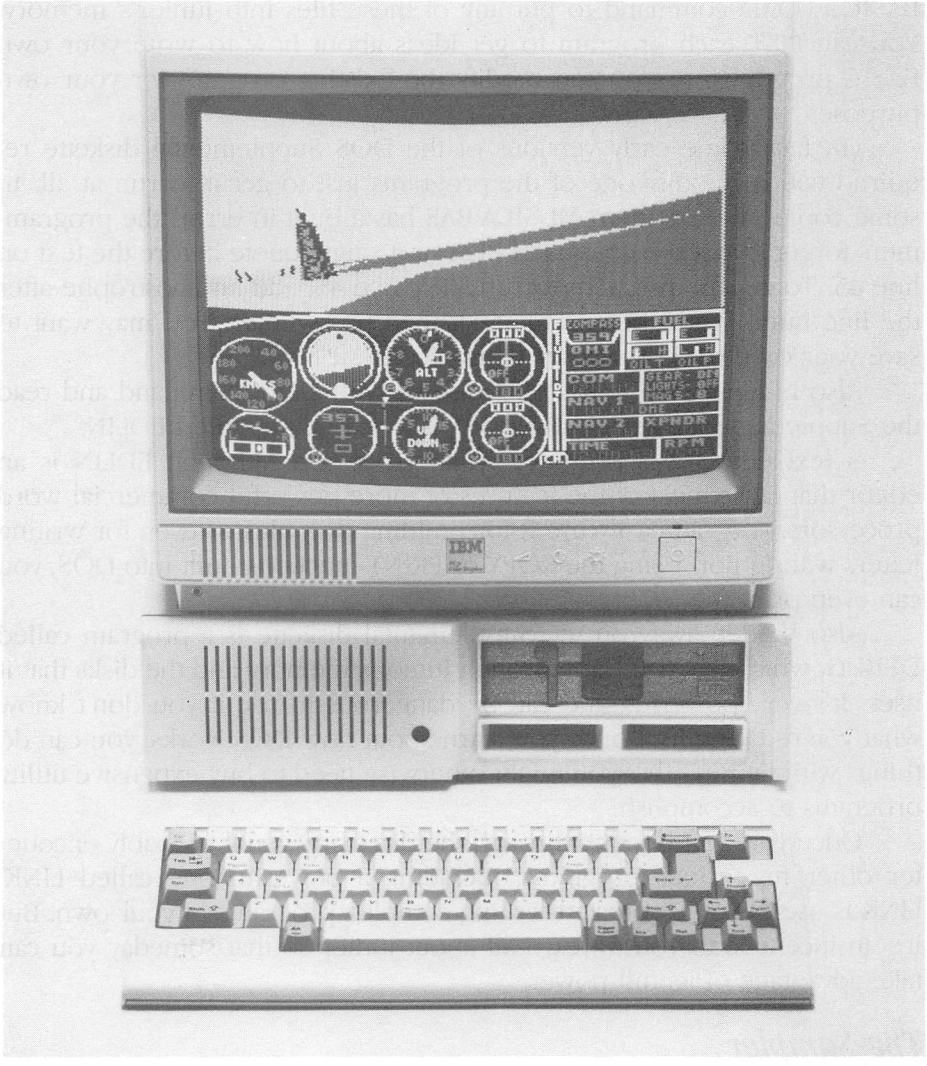
CHAPTER TEN

Hidden Treasures

When you bought an enhanced Junior with a disk operating system (DOS), IBM actually gave you a hidden, undocumented bonus. Included free with your new Junior were several worthwhile programs and a game, all packed on the sampler disk packed in the big box with the computer. You got a word processor and several working BASIC programs, which are not only useful in themselves but can be taken apart and rebuilt to suit your needs or used as models for programs that you write yourself.

In addition to these hidden treasures, Junior offers advanced programs that will let you hunt through every nook, cranny, and sector of your disks to locate data, and then move the information somewhere else. It's true. These programs are all free—you just have to find them and figure out what to do with them.

The trick is knowing where to look. When you buy IBM's disk operating system, you get two disks packed full of programs. By now, you should be aware of the DOS programs themselves, which, among other things, control the disk drive, copy files and disks, and change the time and date inside the computer. Probably, though, you've not yet had occasion to venture onward to the Supplemental Programs diskette. The programs recorded there are designed only for experienced engineers and users of more powerful IBM computers, right? Wrong. Even though the BASIC language included on the DOS disks will not work in Junior (it's functional only in the larger IBM PC, PC-XT, the Portable, and AT),



The PC*jr* is not all strictly business. Its superlative graphics abilities allow ultra-realistic simulation programs such as Microsoft's *Flight Simulator.*

along with the language itself you get several programs that you can easily load into Junior's BASIC and use to your heart's content.

Type a DIR command when you have the Supplemental Programs diskette in the drive, and you'll see a list of programs with the file-name extension .BAS. These are demonstration programs that you can run with your BASIC cartridge simply by typing the word BASIC followed by the file name (without the extension .BAS) and pressing Enter. Or you can use

BASIC's LOAD command to put any of these files into Junior's memory. You can LIST each program to get ideas about how to write your own BASIC programs, or you can modify the existing program for your own purposes.

(In fact, some early versions of the DOS Supplemental diskette required you to modify one of the programs just to get it to run at all. In some copies, the program MUSICA.BAS has a built-in error: the programmers forgot to put the command REM or a single quote before the text on line 45. To get the program to run, LIST line 45, add an apostrophe after the line number but before any text, then press Enter. You may want to save your corrected version by typing SAVE"MUSICB.)

Also hidden, until you get inquisitive with a DIR command and read the Supplemental diskette's directory, is a text editor called EDLIN.

A text editor is a primitive word processor. Although EDLIN is an editor that lacks most of the features of more powerful commercial word processors, you can easily use it for building disk files or even for writing letters with Junior. Using the COPY or PRINT functions built into DOS, you can even print the letter.

Also hidden away on the Supplemental diskette is a program called DEBUG, which lets you hunt through Junior's memory and the disks that it uses. It's very powerful, and can be dangerous to use if you don't know what you're doing. But once you learn about how DOS works, you can do things with DEBUG that you might otherwise need to buy expensive utility programs to accomplish.

Once you start hunting through the directory, you'll probably encounter other mysterious programs, including a powerful one called LINK. LINK is useless until you start writing complex programs of your own. But it's an incentive to you to learn all about Junior so that someday you can take advantage of its full power.

The Sampler

The IBM PC*jr* Sampler, included with enhanced models of Junior, is another gift from IBM. The Sampler was designed to get you computing—and loving it—as soon as you get Junior home. Just slide the Sampler disk into the drive slot and boot it up, following the instructions in the Sampler folder. On the disk you'll find a good selection of programs that demonstrate the computer's abilities and put you in control. Some of the programs are nearly as good as or better than software you can buy.

The Sampler is so easy to use that it truly needs no explanation. In fact, the biggest problem is that you're stuck using the Sampler disk itself

when you want to run any of its programs. But you can get around this by copying the programs on the Sampler that you find useful onto a separate disk with your own programming creations. To do this, all you need is the secret of how the Sampler programs work and depend on one another.

The programs in the Sampler are written in a language called *compiled BASIC*, in which the high-level language instructions written by the programmer have already been translated into the machine language that Junior understands. This means that the programs can run much faster than regular BASIC programs—but also that Junior requires a special reference library in which to look up the meanings of some of the commands the compiled programs send it.

That reference library is in a file called BASRUN.EXE. Consequently, you must also move the file BASRUN.EXE to any disk that you copy the Sampler programs to. If you don't, the program will probably ask you to give it the letter of the drive where BASRUN.EXE can be found. (If it does, you can get the program running simply by specifying drive B and inserting a disk with BASRUN.EXE on it—like the Sampler disk—when Junior asks for the drive-B disk.)

The files on the Sampler disk that contain the actual programs all have the file name extension .EXE. From the file names, you can usually decipher which file corresponds to which program. For instance, the Sampler shopping list program is contained in the file SHOP.EXE. Obviously, then, you must copy the program file to the disk that you want to run it from.

But that's not enough. The shopping-list program, for example, looks for another file on the disk you're using—a file called SHOPPING.LST, which contains the actual list of items that are displayed for your shopping list. If SHOP.EXE cannot find SHOPPING.LST, you'll get a warning message that the disk you've copied SHOP.EXE onto is not the Sampler disk. So you'll have to copy SHOPPING.LST to your destination disk as well.

After you've copied the files that you need to a new disk, just type the transferred program's name, without its file name extension, to run it. To run the shopping list program, simply type SHOP.

You're not done yet, however. When you press Esc to leave the program, you'll get the message warning you that you're not using the Sampler disk and you won't be able to get rid of it. Your computer will stubbornly refuse to do anything until you put the real Sampler disk in the drive slot. If you merely press Esc again, the same message will reappear.

The reason for the obstinate warning is that the Sampler programs all look for another program called MENU1.EXE, which displays the menu on the screen and lets you choose options. Every time you exit a program,

the program's last act is to try to start MENU1.EXE running. If it's not there, you get the error message.

If you copy MENU1.EXE to your new program disk, you'll face the starting menu every time you leave a program you want to run. (You can then also start the program you want to run from the moved menu if you've transferred that program's file to your destination disk.) But then you'll be stuck with a menu you can't get rid of.

You can fool your program into thinking that you've put MENU1.EXE on your disk by making a new MENU1.EXE program. To do this, just create a file (using DOS's COPY command) with that name that contains either no data or only a single character. Then, when you exit your program you'll get the message "Error in EXE file" on the monitor screen, but you will be returned to the operating system and you can go on from there and do what you please. Obviously, you'll never return to the menu, but by that time you'll be so proud of the professional way you can handle Junior, you may never want to look at a menu again.

The World's Cheapest Word Processor

How much should a word processor cost? Is $500 too much? How about $50? Or is $5 closer to your budget? If you are interested in getting the most for your money, start poring through that disk you had to buy to get your disk drive running. On that DOS disk you'll find a program called EDLIN.EXE.

When you first start out, EDLIN might not seem like much. Type in EDLIN, and you'll be told that you haven't given Junior a file name. What possible good could a program that just does that be?

Actually, EDLIN is both good and powerful. It's a line editor that lets you change lines of text or data in a disk file. It also lets you easily make and alter disk files, type in vast amounts of text, change it as you need to, and print it out using other simple DOS commands. You can easily create batch files using EDLIN, and if you're persistent, you can even write letters, turning your screen into a typewriter. That makes EDLIN as good as a word processor, doesn't it?

Well, not quite. EDLIN lacks many features that are expected from even the most primitive word processor. It doesn't format text or wrap lines around without your mashing down the Enter key at the end of each one, and it won't even let you double space without hitting the Enter key twice at the end of each line. But it will let you retype lines without a lot of hassle, it will search and replace, it will read text from other files on your disks, and it even automatically makes backup copies of the files you

edit, just in case. In other words, EDLIN is a bit cantankerous and clumsy to use, it has its quirks and limitations, but unless you resort to piracy you won't find a piece of software to compare to it at twice the price.

To get going in EDLIN, all you have to do is type the program name followed by the name of the file you want to create or edit. (That's why it didn't work when you first tried it—it demands a file name, existing or not.) You can even specify a file on a different disk.

If you give EDLIN the name of a nonexistent file, it will create a new file for you and tell you when it's ready to go with the message "New file." If the file already exists, EDLIN will open it up and let you tinker away at the contents, giving you the message "End of input file," meaning that it has read the whole file into memory. You may get a different message if your file is a long one, but you shouldn't be using long files with EDLIN, anyhow.

EDLIN shows you that it is ready to start editing by putting an asterisk in front of the cursor. The asterisk is EDLIN's prompt. Note its position by itself at the far left side of the screen. At this point the prompt means that EDLIN is waiting for a command, not for you to start typing in text.

EDLIN has two operating modes, indicated by the prompt position. When the prompt is on the far left, EDLIN is in its command mode, awaiting your instruction. Edit/entry mode is indicated by the appearance of line numbers (indented a bit from the left) and the prompt appearing to the right of the numbers. When you're in command mode, you tell EDLIN what to *do*. When you're in entry mode, you tell what you want to *say*.

To put information into a new file, you must move from command mode into edit mode by typing the letter *I*, either in uppercase or lowercase, in response to the far left-hand * prompt. The following helpful legend will immediately appear on your screen if all is well:

```
1:*
```

The 1: is the line number. It will help both you and EDLIN keep track of where you are in the file that you are editing. The asterisk is, of course, the prompt. It indicates the *current line,* which means either the line that you are editing or the line to which any commands that do not specify a line number will apply.

Once you are in edit mode, you can just type whatever you want into the current line, as if you were typing on a typewriter. Note, however, that unlike genuine word processors, EDLIN does not wrap text; it will not automatically end a line and start another. Although it may look like

EDLIN is wrapping because text will be brought from the right edge of the screen back to the left side of the screen, EDLIN will remember everything you enter as one long line. You must signal the end of a line to EDLIN by pressing the Enter key, the way you would press Return on a typewriter. Pressing Enter also puts the line into EDLIN's memory and advances the active line one (shown on the screen by increasing the line number by one). (If you don't press Enter, the longest line you can type is 255 characters. After that EDLIN will refuse to accept any more characters from the keyboard.)

If you make a mistake while typing in a line, just press Backspace or the ← cursor-control key to move back to the error, then retype the rest of the line.

To go from edit mode back to command mode, press Fn then *B* (Break) or hold down Ctrl and press *C* on the line after the last line you want to enter into your file. The Break command will cancel the line that it appears on. If you use either keystroke combination before you press Enter on the last line of your file, you'll lose the last line.

To exit from EDLIN, saving what you typed onto disk in the file that you specified when you entered EDLIN, press *E* (upper- or lowercase), then Enter in response to the far left-hand * prompt. To exit from EDLIN without saving your file to disk, press *Q* (upper- or lowercase), then Enter.

If you haven't noticed, you could have accomplished the same feat of daring using nothing more than DOS's COPY CON commands, probably with much less bother and confusion. EDLIN's abilities really don't show until you want to revise or edit a file.

To learn about EDLIN's full capabilities, therefore, you must have a reasonably long file of at least a dozen lines. Make one full of whatever text pleases you—if you like, copy a few lines from this book. But if your fingers turn to jelly when you type, there is an easier way, EDLIN's COPY command.

EDLIN can copy one or more lines as many times as you want, at your command. At the command prompt, the * in the far left-hand column of your display, just specify the numbers of the first and last lines of the block of text you want to copy, the line number at which you want the first line of the block's copy to appear, then (optionally) the number of times the block of text is to be copied. If you don't specify how many copies should be made, EDLIN assumes you want only one. To avoid confusing EDLIN, you must separate each line number from the others with a comma. Finally, to tell EDLIN that you are specifying the copy command, you must follow your list of line numbers with the letter *C,* upper- or lowercase.

For instance, the command:

```
6,12,34,2c
```

will make two copies of all the lines between 6 and 12 (including lines 6 and 12) starting at line 34.

You can fill a file with text to demonstrate the rest of EDLIN's commands by typing in a single line, then using this command:

```
1,1,1,20c
```

which will make twenty copies of the line you type in.

If you want to change any single line in a file, all you need to do is type the number of the line you want to change in response to the * prompt in the far left-hand column. EDLIN will display that line and enter its edit mode. If you don't want to change the line, just type Enter before doing anything else. If you want to change the line, type in the new line, then press Enter. As soon as you strike Enter, EDLIN will return you to command mode.

If your memory is not as good as your computer's, you might have a problem remembering exactly what words you have recorded in each line, particularly if your file is more than one line long. You might not even know what line you want to edit. Fortunately, EDLIN can list the lines in a file for you with its L (for List) command.

Merely typing L in response to the left-hand column * prompt will display twenty-three lines on the screen (if there are at least that many in your file). EDLIN will attempt to put the current line in the middle of the twenty-three—that is, it will list the eleven lines preceding the current line if there are at least that many lines there. If not, EDLIN will list the first lines of the file, marking the active line with an * following its line number.

You can make EDLIN show you any block of text. If you specify a line number and follow it with an L for List, EDLIN will show you that line and the twenty lines that follow it. If you specify two line numbers separated by a comma, EDLIN will display those lines and all those in between, scrolling the first few off the screen if you ask for more than twenty-three lines. A comma followed by a line number followed by an L will list all the lines before the specified line number.

Similar to the L command is the P, or Page, command, which functions identically except for two twists: Page changes the current line to be the last line displayed, and if no block of lines is specified to display in the

command, P defaults to showing the line *after* the current line and the twenty-two lines that follow it. These simple refinements over L allow you to page through an entire file simply by repeatedly giving the P command. You'll roll through one screenful of text after another in sequence (with overlap between screens).

Even the simplest word processor must have some mechanism to let you add what you forgot in the first draft. EDLIN, too, helps you revise your material. If you want to insert text between lines, just specify the number of the line that falls immediately before the space where you want to make the insertion, followed by the familiar I command. I, then, stands for Insert. If you don't specify a line number with the I command, you'll start inserting text *above* the current line.

After you complete one line after entering the Insert command, EDLIN will let you continue to add lines until you press Fn-Break or Ctrl-C. EDLIN will also automatically change the numbers of the lines after the insertion, so that all the line numbers in the file are consecutive.

You can subtract as well as add lines with EDLIN's D, or Delete, command. Just specify the line or lines (by indicating the first and last line numbers) that you wish to delete followed by the letter D (upper- or lowercase) and press Enter. If you do not specify a line number, EDLIN will delete the current line and make the following line the new current line.

You can even move a block of text, like more expensive word processors do, with EDLIN's M, or Move, command. You bring it to life by specifying the first and last lines of the block of text to be moved, followed by the new line number that you want the first line of the moved block to have, and finally the letter M in upper- or lowercase. Of course, you must separate the various line numbers in the command with commas.

For instance, the command:

```
34,36,2m
```

will move lines 34, 35, and 36 to become lines 2, 3, and 4.

EDLIN's S, or Search, command allows you to search for characters, words, or phrases either throughout your entire file or simply through a delimited block of text. When EDLIN finds the text you send it searching for, it makes the line where it finds the text the current line so you can quickly and easily start editing it.

To use S, first specify the block of text you wish to search through by specifying the numbers of the first and last line of the block (separating the two numbers with a comma, of course). Next, type the letter S in

upper- or lowercase, and *without skipping a space* type in the character, word, or phrase you want to hunt for. EDLIN recognizes your last keystroke before you press the Enter button as the last character of the search for a character, word, or phrase. EDLIN will look for an exact match, so here the difference between upper- and lowercase is important. In a search, "Bill" and "bill" are two different individuals.

S allows you several options. If you type an S command without specifying a character, word, or phrase, EDLIN will hunt for the same thing it did the last time you gave it an S command. If you do not specify a first line number, EDLIN will commence its search on the line following the current line. If you don't specify the number of the last line you want searched, EDLIN will hunt until it reaches the end of the file. If you add a ? between the line numbers and the S command itself, EDLIN will ask "OK?" after it finds the searched-for text. If you press *Y,* then and only then does the line the text appears on become the current line. The command:

```
1Sfish
```

would cause EDLIN to search the entire file for the line in which the word "fish" first occurs and make that line (if found) the current line. Pressing only S subsequently will find later occurrences of the word. The command:

```
30,40?sBig flappy feet
```

will make EDLIN search lines 30, 40, and those in between for the first occurrence of the phrase "Big flappy feet." Before it makes the line in which that phrase first occurs the current line, EDLIN will ask "OK?"

EDLIN will also replace the text you search for with whatever other text you specify when you use the R, or Replace, command instead of S. To indicate the text you want to replace the sought-after text with, press Fn then 6 (or hold down Ctrl and press *Z*—either of which will display ^Z on the screen) instead of Enter after you type in the text you want to search for, then type in the replacement text, then type Enter. For instance:

```
25,400?Rgood, warm puppies^Zcold, evil leftovers
```

will make EDLIN search the block of text between lines 25 and 400 inclusive for good, warm puppies and replace them with cold, evil leftovers. The prompt "OK?" will appear, and only if you type Y will the change become permanent. To look for the next occurrence and replace it, merely type R again.

EDLIN also lets you transfer the contents of another file into the one that you're editing using the T, or Transfer, command. You must specify the line number before which you want to insert the contents of the transferred file, the command itself in either upper- or lowercase, then the complete file name you want transferred. The command:

```
200tB:ELECTRIC.EEL
```

would insert the contents of a file called ELECTRIC.EEL (on another disk) into the file you're editing before line 200. If you've used DOS's hard-disk subdirectory commands, take note that the Transfer command does *not* know how to find its way through elaborate paths but confines itself to the directory you specified in your DOS command that started EDLIN editing your file and the current (necessarily the main directory in a one-drive Junior system) directory of other disks.

EDLIN has other intricacies, like variations on commands and complex methods of handling files longer than the memory that is available in your Junior. Odds are, however, that you're either already discouraged by the immediate difficulty of using the program and don't ever want to hear about it again, or are intrigued enough to explore its greatest depths in minute detail in the DOS manual itself.

Digging In with DEBUG

DEBUG is a special program included on the Supplemental Programs disk included with IBM's disk operating system. Its primary purpose is to help experienced programmers working in low-level languages, like assembly or machine language, find elusive flaws in their programs and correct their mistakes before putting the software into the hands of other human beings.

Even if you don't intend to program in machine or assembly language on Junior, however, DEBUG can be a fascinating tool that will unlock many of the mysteries of your machine and the programs it uses. DEBUG can peer deeply into the hidden recesses of Junior's memory and reveal to you deep, dark disk secrets that only special programs can normally probe.

Although DEBUG has many commands, only a few of them are important for a beginner's peek. Like EDLIN, each command in DEBUG consists of a single letter, perhaps followed by an *argument* of additional letters and/or numbers.

D, according to the DOS manual, stands for Dump, but is more

understandable as Display. It will cause Junior to display the numbers stored in any given memory location anywhere on disk, in a cartridge, or in Junior's internal solid-state memory.

E stands for Edit. It allows you to edit or change the contents of any address in Junior's RAM memory.

L stands for Load. Using L you can load any sector you want from a disk into Junior's memory, then look at an image of the sector's contents using D.

Q is the all-important Quit command that lets you stop the DEBUG program from running without doing any damage to your disk or machine.

W stands for Write, and is a very, very dangerous command, which, if used foolishly, can make an entire disk unusable. It allows you to write the data from specific bytes of Junior's solid-state memory to specific sectors of a disk.

To start DEBUG running, just type the program's name—DEBUG—when a copy of the Supplemental Programs DOS disk is in Junior's disk drive. In a few seconds, a hyphen will appear on the screen; that's DEBUG's prompt and it means that the program is awaiting your command.

To see how DEBUG displays Junior's memory contents, start by taking a look at some of the data stored in one of Junior's cartridges. With your BASIC cartridge in the right-hand cartridge slot, try typing this command in response to DEBUG's prompt:

```
D E000:8088
```

The numbers following the Dump command indicate which memory location you want to investigate. The number to the left of the colon indicates which 64-kilobyte segment of Junior's microprocessor's 1,024-kilobyte addressing capacity you want to investigate. The E in the number is the hexadecimal representation of the value 14, and it indicates that you want to peer into the second highest segment of Junior's memory, which is always assigned to the right-hand cartridge slot. (Hexadecimal is a number system based on groups of 16 rather than normal decimal-notation 10s.) For access to the left-hand cartridge slot, the address is one segment lower, so the command would be D D000:8088. The 8088 in the command indicates which memory addresses within the specified segment you want to look at. The similarity between this number and the numerical name of Junior's microprocessor may not be coincidental.

After you're certain that you haven't made a typing error, press the Enter key. The block of two-digit numbers that should appear in the middle of the monitor screen are the contents of each memory location in hexadecimal notation. In Junior's 40-column display mode, a translation of

these numbers into displayable letters and numbers will appear below each line; in 80-column mode the translation will appear on the right side of each line. The left-hand column gives the address of the first byte of each row of the display.

The characters you should be able to read are not the copyright message that appears on your screen when BASIC first starts, but rather a copyright message that identifies IBM's ownership of the rights to the instructions in computer code inside the cartridge—words meant to strike fear in the hearts of program plagiarists.

You can look in another interesting place, Junior's built-in ROM memory, with this command:

```
d f000:ff7f
```

which will display up to the penultimate byte of Junior's ROM memory. You'll note that the hexadecimal code for the last byte that is displayed is FD. This is the "system identifier" that indicates what kind of computer the ROM chip is (supposed) to be installed in.

At first, you might think this information would be useless—after all, you can identify your computer just by looking at it. Programs, however, need to know what sort of computer they are running in, so they know what characteristics of the computer they can depend on. The programmer can send the program to this memory location so that the program itself can identify the machine. FF in this memory location would indicate an ordinary IBM PC; FE indicates an IBM PC-XT or an IBM Portable Personal Computer. (The date preceding the system identifier is the release marker of the ROM chip, indicating when its programming was created.)

Press *D* and Enter again and you'll bounce back to the beginning of the F000 segment of memory, where you'll again see IBM's part number and the copyright notice for the ROM chip.

You can use DEBUG to peer at disk memory in two ways. To look at the contents of any given disk file, simply specify the file you want to examine after the name DEBUG when you start DEBUG running. For instance, the command:

```
DEBUG B:ROSESARE.RED
```

will automatically load the contents of a disk file that is not on the same disk as DEBUG into Junior's memory (after you play shift-the-disks) for examination at your leisure. You'll be able to read the file contents when

you DEBUG a text file, and you'll get all sorts of strange notations when you DEBUG a program or other nontext file. The numbers in each memory location of a program file are the actual commands in machine language that tell Junior how to run the program.

The other way to stick your nose into disk memory is to use the L command to load from disk into RAM chunks of memory that are defined by sector location on the disk rather than file name. Although the information stored on each disk is different, every disk shares certain common aspects—the location of the file directory, the file allocation table, and the hidden files of the disk operating system, for instance. To load the sectors containing the disk directory, type this command:

```
L CS:0100 0 3 4
```

then press Enter. Your disk drive will start to spin for a couple of seconds, then stop, and a new DEBUG prompt—another hyphen—will appear on the screen. The command tells DEBUG to load into Junior's memory register CS, starting at location 0100, information from the A: disk drive (0 in the third position stands for the A: drive; 1 there would mean the B: drive) that is recorded on sectors starting with sector 3 and ending with sector 4. (Disk sector numbers, like most counting systems in computing, start with 0, so you've specified the fourth and fifth disk sectors.)

To see the contents of the sectors you've loaded, you must ask the computer to display them. Type this command:

```
D CS:0500
```

to display the contents of the CS register starting at location 0500, where things begin to get interesting. You'll see the first few bytes of the disk directory. Each directory entry takes two full rows of display (in 80-column mode) with the individual bytes assigned as follows:

1. The first 7 bytes indicate the file name. (If the first byte contains E5, a lowercase *e,* the file has been previously erased.)
2. The next 3 bytes are the file-name extension.
3. The next byte indicates a file attribute. It can indicate that the file is read-only, hidden, or a disk name (or "volume label," in IBM terminology), or it can reveal other specific information about the file. For instance, if this byte is 08, the file name is actually the volume label.
4. The next 9 bytes are not currently used.

5. The next 2 bytes code the time when the file was last updated. The number is coded backward, with the least significant bytes first.
6. The next 2 bytes code the date that the file was last updated, again with the least significant byte first.
7. The next 2 bytes tell where the beginning of the file is recorded on the disk. (Later portions of the file can be found, by either you or DOS, only by consulting the file allocation table.)
8. The last 3 bytes encode the size of the file.

If you are really adventurous, you can load the directory of an *expendable* disk into DEBUG and change any of the just-mentioned bytes in any file's directory entry using the Edit command. Type E followed by the memory location you want to change, skip a space, then list out the new contents of that byte either as a two-digit hexadecimal number or as a string enclosed in quotation marks. You can change several sequential bytes by following your E with several two-digit hexadecimal code numbers (separated by spaces) or a long string. For instance, this command:

```
E CS:0500 01 01 01 01 01 01 01 01
```

will change the file name of the first directory entry to eight "smiling faces" symbols, and make the file totally unusable.

However, the E command does not make your changes permanent. To do that, you must record the changes onto your disk by using the Write command to write what you've stored in DEBUG to the same place on the disk you originally got the data from. All you need to do is substitute W for the L of your original Load command, like this:

```
W CS:0100 0 3 4
```

Such procedures are not for the faint-hearted because the slightest mistake can make a file or a whole disk totally unusable. Nevertheless, if you're courageous or foolhardy, you can put some really strange features into your directory to confuse and amaze your friends—or you can make files invisible and/or unerasable to suit your own purposes. To become adept at such procedures, you should invest in IBM's DOS *Technical Reference Manual.*

Taking Care of Junior

Junior is a complex precision instrument, but it's also an amazingly hardy machine. It's designed to be able to withstand the misdeeds of children, both at home and in the classroom. Odds are, Junior will be tossed around, sat on, and generally beat up, and it will still be expected to work flawlessly. It's likely that it will, too. And if the Freeboard inadvertently gets wet, just wipe it off and it will probably work without a hitch.

But despite Junior's resilience, there are steps you can take to assure a long, trouble-free life for your computer. With just a little extra care, you can guarantee that Junior will stay looking—and working—like new.

The extra care is called "preventive maintenance," and the few extra seconds it takes can do a lot to prolong the life of your computer. Preventive maintenance is the same as changing the oil in your car more often than the factory recommends or polishing the chrome every week—doing a little extra work to keep your investment in tip-top shape. You may be tempted to try the same loving care with Junior, only it has no oil to change or chrome to polish. What can you do to keep it happy?

Obviously, the minimum you can do for Junior is absolutely nothing. Ignore it and let it fend for itself, but don't worry about your thoughtlessness because IBM's own recommendations for Junior are minimal. The truth is that Junior's microelectronics (or those of any computer) require no mechanical care or maintenance. Junior has no bearings that need grease, no cams and levers to adjust, in fact, no moving parts at all that can

wear out. It will work fine with no effort from you other than turning it on when you need it.

Modern solid-state electronics don't simply grind down slowly, they fail catastrophically. One moment they work, the next nothing is left except your frustration. Worse than that, no matter what you do, there's a chance your Junior will someday, perhaps very soon, suffer a catastrophic failure.

Fortunately for the space program, undersea research, and the electronic ignition of your car, such catastrophic failures are rare. After the marginal components built into any modern electronic device have been weeded out, disasters rarely occur more often than once in tens of thousands of hours of operation—years of normal use. Junior's one-year warranty is more than enough to carry it through the initial danger period, the first weeks of weeding and weaning.

In fact, if a failure is going to occur in a solid-state device, it will usually happen within the first few *hours* of operation. If your Junior works after you turn it on for the first time, and perks happily along for a few days, it should cogitate for years to come without you lifting a finger except to massage the Freeboard.

The logical thing to do when you become the owner of a brand-new Junior is, then, to set it up and test it with data that you wouldn't mind losing. Don't trust the computer with the keystrokes of your first novel or your budget or any other irreplaceable data until you're certain it won't suddenly fail and destroy all your work. Use the "burn-in" time for familiarizing yourself with the strange new Freeboard, wander through the programs IBM supplied, or blast away at a game.

Junior had two problem areas when it was introduced: the disk drive and the solid-state RAM memory. A problem in either one spelled disaster to any data that was typed in. To be sure your Junior doesn't have these potential problems, simply leave it turned on for a couple of days. When you go to bed, switch off the television or monitor, but leave Junior running. Keep tabs on it. Occasionally fire up the monitor and take a look. Run a quick program. Make sure that in your absence it hasn't suddenly frozen up, that things haven't changed all by themselves in its memory or on the monitor screen, and that programs that ran yesterday will still run today without a turn-off-and-on reset. If something changes while you're gone *and no one has touched Junior,* it's likely that your machine has a flaw that may bring on a digital disaster. Report it to your dealer (or any authorized IBM dealer).

A Happy Home

Once your Junior has proven itself, you can take added precautions that go beyond the factory's nearly nonexistent recommendations and assure yourself that Junior will remain in perfect shape—and give the machine a longer and more trouble-free life in the process.

Mandatory care for Junior amounts to common sense. The best advice you can follow is to be reasonable. Even though Junior won't complain about its surroundings, it is nevertheless a bit particular about them. Its internal circuitry is designed to work within a specific temperature range, from 60° to 90° F, and within a humidity range from 8 to 80 percent. Although a temporary excursion outside those limits while the machine is on may not permanently injure Junior, it may damage your data. When Junior is not on, the limits are a bit wider, 50° to 110° F. Submitting Junior to temperatures outside of that range may, in fact, damage it, particularly if you hit an extreme. So it's probably not a good idea to set up your computer in a dank basement or overheated attic.

A rule of thumb is that computers generally operate well under any conditions of temperature and humidity that humans would be happy with. That does not mean that a computer needs the same perfect temperature that you might demand for absolute comfort. Rather, the best preventive measure is to avoid putting a computer in a hostile environment.

Although the contacts inside the Freeboard are somewhat waterproof, they may succumb to airborne irritants. In highly humid or chemical-laden environments, contact corrosion may occur. Damp basements are great for mushrooms but could spell early retirement for Junior; the beach is best for bunnies but torture for a computer.

Power Line Problems

Even though electricity is Junior's lifeblood, it can be its worst enemy. Not that a steady supply of electricity is bad for the computer; in theory, at least, a solid-state computer won't wear out electrically no matter how long you leave it plugged in. But the current supplied by your local electric utility company is not pure enough to guarantee that theoretically endless life.

Supplied free of charge (and free from recourse) along with the nominal 110–120 volt, 60 hertz (cycle per second) alternating current delivered to your fuse box is a collection of spikes, surges, pulses, glitches, noise, and absences of very brief and prolonged durations. Any deviation from the steady current your computer expects is potentially life threatening to the machine or its data.

Glitches and spikes are most damaging. Both terms describe essentially the same phenomenon: a temporary pulse of abnormally high voltage on the power line. "Temporary" can mean anywhere from a few millionths of a second to nearly a second long. Abnormally high voltage can range from just a couple of volts above what you are supposed to get to 10,000 volts and higher. Because of the brief durations of spikes and power-line glitches, they are invisible. They won't cause the slightest flicker to your lights, and only specialized equipment can determine whether or not they are present on your power line.

Spikes and glitches can be caused by anything from lightning striking near a power line to brief pulses of power added to the electric line when the large motors in household appliances or industrial equipment switch on and off. Any single large spike can be enough to permanently damage the microcircuits in Junior.

Some studies have shown that the effects of spikes can be cumulative, similar to a human getting small doses of poison over weeks or months. Each spike can slightly damage the tiny internal structure of Junior's microcircuits. When enough damage has been done over time, the computer may mysteriously stop working or work erratically, maybe refusing simple functions or giving strange answers to obvious questions.

The best way to protect Junior from power-line spikes and glitches is with a *surge suppressor.* Many brands and styles of suppressors are available.

Although operating principles vary, the most common type of suppressor uses a novel electronic device called a *varistor,* which effectively swallows up most spikes and glitches and prevents them from reaching your computer. One suppressor is sufficient to protect Junior and all its peripherals.

Closely related to spikes and glitches are surges and overvoltages, which are voltages that are slightly higher than normal but that last slightly longer to much longer than spikes or glitches do—from a fraction of a second to several seconds. Surges may last long enough to cause the lights in your home to flicker or get momentarily brighter.

The low voltage rise of surges means that most inexpensive spike or surge protectors actually do little good in eliminating them; only the best (and most expensive) "ferroresonant" types of surge suppressors or voltage regulators offer genuine protection.

Fortunately, in most cases such protection is overkill. Practical experience indicates that Junior is probably impervious to most minor surges. It does have a limited amount of built-in protection from them, and complete protection is likely to cost more than Junior itself.

While power problems that result in low voltage or none at all

probably won't harm Junior, they may wreak havoc with the data and programs in its RAM memory. When voltage dips low enough long enough, the power-line condition is equivalent to turning the computer completely off. That means everything in its RAM memory is wiped out, and when power again reaches a normal level the turn-on cycle will begin anew.

When this happens, you must reload whatever was in RAM, wasting seconds, if you had been using a program safely stored on disk, or hours, if you were developing a lengthy program that you had laboriously entered keystroke by keystroke and not bothered to save.

The anguish caused by voltage drops can be prevented by using a voltage regulator, which automatically adjusts the power supplied to your computer to the optimum level. Your computer's memory can be kept fresh even through complete power outages by using an *uninterruptible power supply* (UPS), a device that has built-in batteries with the electronic circuitry necessary to convert their output to normal line current. The UPS ensures that the computer's power supply remains constant.

Even the least expensive regulators and uninterruptible power-supply devices are likely to cost several times more than the price of Junior. A more economical solution to power problems is to save your work to disk early and often, and keep backup copies of everything you do.

Static Protection

The other electrical enemy of any computer is static electricity—the blue sparks you feel after shuffling across the carpet and touching a doorknob on a dry winter's day. Static electricity is genuine electricity, as the tingle at your fingertips tells you. The voltage generated by a short shuffle can be tens of thousands of volts.

There is little current behind this voltage, so static shocks are mostly only annoying and rarely fatal, except to computer circuitry. To delicate microcircuits, a static spark can be more deadly than a spike on the power line. The tiny sparks of static can be enough to blast a silicon microchip to data heaven and run the repair bill into hundreds of dollars.

The only protection from static is prevention. If you don't make sparks, they won't hurt anything. Dozens of products are available to help keep static under control—special carpets, chairs, mats, and sprays. There's little doubt that they work. But while some form of static control may be necessary in the business computer room where one byte of data can mean a million dollars, expensive static protection in your home is probably unnecessary.

Junior itself does a good job of warding off static sparks. Sparks

usually jump from finger to metal; that is, from your body, which can store a great deal of static electricity, to a conductor that can quickly drain that electricity off. Touching Junior's insulating plastic case doesn't drain off the static charge fast enough to generate a damaging electrical spike.

The best static prevention is inexpensive and good for you—it's worth doing even if you don't care about your computer's health! Static electricity can only build up in your body when the relative humidity is low enough that the moisture in the air, all by itself, does not slowly drain off the static charge as fast as it is created. Raising the relative humidity of your home to 30 percent or higher by using a humidifier will effectively prevent static shocks to you and your computer.

Disks and Cassette Care

The worst enemies of Junior's magnetic storage media—disks and cassettes—are invisible. Certainly tapes and disks are vulnerable to the same enemies that plague your important documents, the spilled coffee and dropped cigarettes. But compared to the records kept on paper, magnetic media require extra thought and care in both use and storage. A host of unseen enemies that you've probably never had to deal with are poised and ready to wreak havoc with them.

The disks used by Junior need the same special treatment that all computer floppy disks demand. You've probably seen the warnings a dozen times already. Keep diskettes away from magnetic fields such as those created by hi-fi speakers, power transformers, electric motors, and the bell in a telephone. The errant magnetic fields from such devices can alter or erase irreplaceable stored data.

Always take care to keep diskettes inside their protective sleeves when they're not in the disk drives. That way dust, airborne contaminants, grease, and grime won't pollute the sensitive magnetic surfaces.

When handling a floppy disk, guard particularly against touching its magnetic surface where it shows through the slots in its black plastic case. Tiny drops of oil in a fingerprint can be enough to mess up a disk and gum up the works of a disk drive.

Cigarettes pose a particular danger. Not only are they harmful to the health of smokers, but they can have lethal effects on computers, too. Stray ashes have a devious way of finding high-precision parts, such as disk drives and even the diskettes themselves, to collect on. A speck of ash or even a single particle of tar from a cloud of cigarette smoke can be enough to foul some disk units. Try to avoid smoking when using Junior.

Cassettes and cassette recorders are generally more robust than disks

and disk drives and are therefore able to withstand much more abuse. For years stereo cassettes have been tossed around, taken to the beach, and left inside cars on summer days with few ill effects. Tape players have withstood similar treatment without complaint.

Nevertheless, carelessness with cassettes can lead to disaster. Touching the tape surface with your fingers will pollute it with oil that will attract dirt and shorten the life span of both the cassette and the recorder you play it on. Throwing a cassette loaded with important programs and files into a desk drawer containing a pair of magnetized scissors is inviting ruin.

Taking the same care of computer tapes as you would stereo tapes is necessary to keep your programs and files healthy and long-lasting.

Cassette Recorders and Disk Drives

Although there is no standard recommended care for disk and cassette drives besides proper protection, they can be safeguarded by regular and preventive maintenance.

Both disk drives and cassette recorders work exactly like other magnetic recording machines and use "heads" to actually record and play back (write and read, in computerese) their signals to and from the disks or cassettes. Just as with stereo tape recorders, cassette and disk-drive heads on computers can get dirty and magnetized and thereby damage the signals they read and write.

Although stereo tape players begin sounding muddier and more muffled as dirt and magnetism accumulate on their heads, computer cassettes and disk-drive heads use digital signals that give no hint that the heads are encumbered by dirt until they actually misread or miswrite data. Obviously, it's recommended that you clean computer cassette and disk-drive heads regularly to avoid data errors.

Although various sources disagree on how often head cleaning is necessary, once every ten to fifty hours of use should be enough to prevent problems, and can extend the lifetimes of disks and the drives themselves.

Head-cleaning cassettes and disks make this routine chore simple. Just unplug your cassette recorder from Junior by unplugging all three plugs on the cassette adapter cable from the tape unit (so that you can operate its controls manually), and insert the head-cleaning cassette in the cassette recorder. Run the head-cleaning cassette for a few seconds, then remove it and reconnect the recorder.

To clean the head of your disk drive, first dampen one of the many

available head-cleaning disks with its special cleaning solvent, then run the disk for a few seconds in the disk-drive unit. One way to exercise the disk drive so that the head sweeps across the cleaning disk and cleans itself is to instruct Junior to try to load an imaginary file—and any file on the nonmagnetic head-cleaning disk will be imaginary! Just type any letter or character in response to Junior's A> prompt and the disk drive will spin for a few seconds, cleaning the heads, until the error message pops up on the screen.

If you don't want to deal with sloshing solvents around, you'll welcome products like the Datalife system of head-cleaning disks made by Verbatim, in which a pretreated, presoaked cleaning disk is provided in a sealed envelope. The disposable disk is put into a reusuable jacket and inserted into the drive slot for a fast, easy, and safe cleanup. These head-cleaning products are available from many sources, including your favorite IBM dealer.

If you're more adventurous, you can go right to the heart of the matter and directly clean the read/write heads of your cassette and disk drives.

For cassette recorders, a cotton swab is a suitable cleaning tool. Soak the swab in a suitable head-cleaning solvent and scrub the heads of the cassette recorder to clean them.

You can clean the heads of your disk drive similarly, but you must be very, very careful. If you are too vigorous in your cleaning, or just clumsy, you can ruin the disk-drive mechanism and run up a repair bill of $100 or more. Do *not* use cotton swabs on disk-drive heads. A strand of cotton can come off the swab and foul the head, possibly destroying it. Instead, use a special video-cassette head-cleaning applicator or a photo chamois or similar lint-free cloth.

Be sure to turn Junior off and disconnect all peripherals before you begin. Then remove its top cover to gain access to the disk-drive mechanism and its read/write heads.

The recommended solvent for cleaning either cassette or disk-drive heads is tape head cleaner, a special fluorocarbon solvent that can be bought in nearly any stereo store. Alcohol, either isopropyl (ordinary rubbing alcohol—a 90 percent solution is preferred but 70 percent is okay) or methanol (methyl alcohol or shellac thinner!) can be substituted as a tape or disk-drive head-cleaning solvent. Even high-proof vodka will do in a pinch.

As with a stereo cassette recorder, computer cassette drives and disk drives benefit from an occasional head demagnetization. The constant rubbing of the magnetic disk or cassette past the read/write head can cause the head to become magnetized. If enough residual magnetism is

built up in the head, its ability to respond to high data rates (at the innermost sectors of disks, for instance) can be impaired. And magnetized heads gradually erase the recording medium that passes them by.

The magnitude and extent of the effects of possible head-magnetization problems in personal computer systems are subject to debate, and the procedure requires a great deal of care. Unless you have some experience with servicing computer components, it's probably not a good idea to attempt to demagnetize disk-drive heads yourself. However, if you're familiar with the assembly and disassembly of precision equipment you may want to regularly demagnetize the read/write heads in your computer system just to be on the safe side.

Any good-quality tape head demagnetizer designed for use with stereo recorders will probably do a reasonable job on Junior's heads. For cassette drives you can use the kind of specially designed cassette head demagnetizer (also called a "degausser") that is built into standard cassette shells and operates on batteries. For disk drives you'll need the plug-into-the-wall variety of demagnetizer that has either a single rodlike extended "pole piece," or two flat pole pieces that curve together. No matter which style of pole-type demagnetizer you prefer, make certain that the pole pieces that extend from the main body of the demagnetizer are covered with a soft plastic compound to reduce the chance that you might accidentally scratch the read/write head.

Using any pole-type demagnetizer or degausser takes less than a minute and is a good adjunct to your regular head-cleaning routine. That means you'll have to disconnect and disassemble Junior to get at the disk drive.

First, turn off and unplug the unit to be demagnetized. Then bring the degausser near (or lightly touching) the head and *only then* turn it on, slowly removing it from the proximity of the head (to about 3 feet away) and turn if off. The key to the demagnetizing process is the sudden turn on and slow reduction of the alternating magnetic field of the degausser.

Good Housekeeping

A clean computer is a happy computer. Dirt is the mortal enemy of any computer system. Not only does dust injure Junior's pride, it can also damage mechanical parts: it cakes on print mechanisms and causes them to slow and bind, and slowly grinds away like sandpaper at delicate disk-drive parts. If your printer and disk drive are so layered with dust that you need to hire an archeologist just to find them, you'll probably soon be searching out the warranty cards and inspecting the phone book for repair shops.

The best way to keep dust from becoming a problem is by developing a regular cleanup program for Junior. As often as necessary, simply vacuum the danger away. Use the finest nozzle available to suck dirt from inside the disk drives and from between the keyboard keys and within the printer. Never blow the dust away, as you're just as likely to blow it into a precision assembly. Blowing merely rearranges dust; vacuuming removes it from the area.

The dust that inevitably collects on your monitor or television screen can be removed with a soft cloth and ordinary window cleaner. Use paper towels and elbow grease as if it were just an ordinary piece of glass, but spray the solvent on the cloth rather than the display screen to avoid the inevitable overspray from seeping into the ventilation holes of the monitor. If you feel wealthy and want to lavish extra care on Junior, you can buy special CRT cleaners, but in the end the results will be about the same as if you'd just used window cleaner.

Window cleaner is also perfect for removing smudges and grime from the cases of Junior and its peripherals. Again, take care if you plan to use a spray cleaner. Don't allow overspray to find its way into *any* openings into the computer, including the disk-drive slot or between the keys of the Freeboard.

Be wary of using strong solvents like acetone, or nail-polish remover, which frost, glaze, or melt plastics. They can ruin the textured finishes of computer equipment and might even melt holes in it. If in doubt about a cleaning fluid, dampen a cotton swab with it and test it on the back of the cabinets where damage will be hidden.

If you accidentally spill liquid on the Freeboard, let the excess drain off and wait for the keyboard to dry. Although the Freeboard is more immune than most computer keyboards to such shenanigans, a sugary fluid will turn it into a sticky mess. If you spill plain tap water on the Freeboard and you react fast enough to prevent drips into its circuitry, the Freeboard will probably work fine after it's dried out.

For severe spills such as soda or coffee, try diluting the damage by flooding the Freeboard (after removing the batteries and/or unplugging it from Junior) with plain water and then drying it out. Wait at least overnight before replacing the batteries and trying it out. Don't try to hasten the drying process by putting the Freeboard in the microwave oven or by blow-drying it. This cleanup is dangerous and its results are not guaranteed, nor is it recommended by the factory. Use this emergency procedure at your own risk.

Although it is a good idea to protect electronic devices from the ravages of airborne dust and grime, never cover Junior with any sort of

cover *while it is turned on*, as the cover will prevent air from circulating and cooling Junior off properly. As we've seen, high temperatures will shorten a computer's life.

When Disaster Strikes

Sometimes things just don't go right. Remember the times when no matter what key you pressed you got a "bad command or file name" or "syntax error" message, your monitor picture shrunk to a tiny dot in the center of the screen, or you keyed in "LOAD" and discovered the 10,000-line program you finished at 3:00 A.M. yesterday morning was nowhere to be found?

When these mishaps occur, your first urge may be to get an axe to gently persuade Junior back into line. Often, however, a cool head can prevail. Most of the problems that Junior suffers are caused by minor slips of the programmer's fingers, forgotten routines, missing quote marks, or having the wrong disk in the drive unit. In these cases, just retrace your steps and you will probably find the transgression.

Sometimes, however, your mistakes or omissions can be subtle, illogical, unreasonable, or obscure. To overcome them, you must first determine in what part of the computer system the problem lies. Is it in the hardware or in the software? Hardware problems can lead to the repair shop if they're serious. But if software is the problem, and you wrote the program that crashes the system, you'll probably be able to fix it. Obviously, then, the first thing to do whenever you run into trouble with your computer is to determine whether it's a hardware or software error.

If a program goes awry, first press Fn then Break and stop the action immediately. If that doesn't work and you're still losing ground fast, press the infamous Ctrl-Alt-Del trio and start from scratch. If your keystrokes have no effect, turn Junior off for five seconds and then back on again. This will usually put you back in control.

Then give everything a quick check and if all seems well, you've probably run into a software problem. Breathe a sigh of relief and prepare to spend a while trying to find the bug in your program.

If resetting Junior doesn't work, try running a different program (after resetting). If that program works, your computer is probably alive and well, but the first program is ailing.

To be certain that the problem is in your software, you may want to run Junior's built-in Diagnostics, a special program that checks major parts of Junior's circuitry to find real and potential problems.

To start Diagnostics, press the Ctrl, Alt, and Ins keys simultaneously. You should be greeted by the Big Blue IBM logo and a quick count of available memory. Then the screen will turn into a display of your diagnostic choices—different parts of the system that you can check, each represented on the screen by an icon. Pressing almost any key, except Enter, will step you through the range of devices to check, moving the cursor to a position beneath the next icon. Although you cannot step backward, you can go forward in a loop.

Using the spacebar or the right-arrow key, move the rectangular cursor to the item you want to check, press down on the Enter key, and the diagnostic for that part of the system will begin. Diagnostics are detailed in your IBM PC*jr Guide to Operations,* section 6.

Hardware Problems

Some common problems you'll encounter with Junior can't be checked with the Diagnostics. For instance, how can you check your monitor with the diagnostic program if the screen won't even light up? Fortunately, most such "equipment failures" are merely inadvertent maladjustments or common omissions, such as the wrong switch thrown or a tape in the wrong place, which can be quickly corrected if you know what to look for. The hard part is finding exactly where the problem is.

To help you on your way to computer troubleshooting proficiency, here's a quick run-through of some of the most common hardware-related problems that you're likely to encounter.

Symptom:

Television screen filled with TV program rather than computer image.

Probable Causes:

a. Wrong channel selected on television set. Make sure your set and the connector for TV are tuned to the same channel.

b. Switch on connector for TV not in "computer" position. Flick the switch to the "computer" position, then make sure the proper channel is selected.

Symptom:

Television screen gray or filled with static.

Probable Causes:

a. No power—Junior switched off. Switch it on!

b. No power—loose plug. Be sure the entire system is properly plugged in. Check power cord from black transformer to wall, and from black transformer to Junior.

c. Wrong channel selected on television set. Make sure your set and the connector for TV are tuned to the same channel.

d. Connector for TV in "television" rather than "computer" position. Adjust switch properly.

e. No signal reaching television—check cable from Junior to TV set. Make sure all connectors are in solidly. Check that wires are solidly screwed to terminals both on connector for TV and on television. Try wiggling connectors—if picture flickers and you can detect part of an image, the cable probably is not making good connection or is defective.

Symptom:

Monitor screen white or gray, no image.

Probable Causes:

a. No computer power—Junior switched off. Switch it on!

b. No computer power—loose plug. Be sure the entire system is properly plugged in. Check power cord from black transformer to wall, and from black transformer to Junior.

c. *RGB* monitor not properly plugged into Junior—check cable from Junior to monitor. Make sure all connectors are in solidly. Try wiggling connectors—if picture flickers and you can detect part of an image, the cable may be defective. Replace it.

Symptom:

Monitor screen gray or black, no image.

Probable Causes:

a. Monitor switched off—turn it on!

b. No signal getting to *composite* monitor—check cable from Junior to monitor. Make sure all connectors are in solidly. Try wiggling connectors—if picture flickers and you can see part of an image, the cable is probably defective. Replace it.

Symptom:

Suddenly all the keys stop working. Pressing keys does not cause characters to appear on screen. Even the Ctrl-Alt-Del combination doesn't help.

Probable Causes:

a. Junior cannot see Freeboard or dead batteries in Freeboard. Point Freeboard more directly toward Junior. If that doesn't work, turn Junior off and back on and see if it then responds. If Junior still fails to respond, replace batteries in Freeboard or use a direct cable connection. If that doesn't work, run Diagnostics.

b. Software hemorrhage—whatever you did has preempted Freeboard control. Turning Junior off and on should clear the problem.

Symptom:

Canned program running. Can't stop program by pressing Fn then Break or any other key combination.

Probable Causes:

a. That's how the program was designed! The canned program can't be stopped because the software company doesn't want you digging in and seeing how it works, or copying it.

Symptom:

Shift keys work backward. Uppercase characters appear on monitor when Shift not pressed, and lowercase characters appear when Shift is pressed.

Probable Causes:

a. Wrong keyboard mode—you accidentally pressed the Caps Lock key. Press this key again to return to normal, unshifted mode.

Symptom:

Cartridge won't work, though it has worked before.

Probable Causes:

a. Cartridge not inserted fully in socket. Press it in firmly.

b. Dirty contacts—the contacts that connect the cartridge to Junior can become contaminated and not make good contact with the connector on the computer. (The contacts are the small metal areas on the edge of the cartridge that is inserted into the computer cartridge slot.) Clean contacts by gently rubbing a pencil eraser over them. The eraser scrubs off the contamination so that good contact is made. Be sure to remove the eraser shavings before plugging the cartridge back in.

Symptom:

New cartridge won't work—strange patterns on screen or no image at all.

Probable Causes:

a. Defective cartridge. Try running another cartridge or program. If the other program works, your new cartridge is bad. Take it back.

Symptom:

Program on cassette fails to load into Junior.

Probable Causes:

a. Cassette recorder not plugged into Junior. Turn off Junior and make sure connector is securely in.

b. Tape problem—wrong cassette. You may have accidentally put the wrong cassette into the tape machine. (The results of that error are quite predictable.)

c. Tape problem—partial erasure. If the tape cassette has not been properly stored, it can be partially erased and might never be loadable again.

d. Dirty heads—if your cassette unit is not properly maintained, the heads can get so dirty that errors result in loading, or tapes may not load at all. Cleaning the heads of the cassette recorder often solves this problem.

e. Jammed tape. If a cassette jams or fails to run properly through the cassette machine, resulting in improper loading or saving, the problem may be caused by a bad cassette (toss it) or by a badly wound cassette. The latter condition usually shows up after a very long tape has been

rewound from one end to another. The actual problem is that the tape inside the cassette is unevenly wound and is binding against the cassette shell.

You can often fix such cassettes by holding them in the palm of your hand and rapping them sharply against a solid object, but not so sharply that the cassette shatters. A measured amount of shock causes the tape to fall back into line better and frees up the sticky mechanism.

Symptom:

After a save to tape, a program cannot be loaded; no trace of it can be found.

Probable Causes:

a. You forgot to press down the Record button when you saved the program. All you can do is try again, remembering to press the button.

Symptom:

Cannot save to tape. Record button on cassette recorder won't press down.

Probable Causes:

a. No cassette in recorder.

b. Write-protect tab missing from cassette—use self-stick tape to cover the write-protect holes on the cassette.

Symptom:

Disk drive will not operate properly.

Probable Causes:

a. Disk in drive not formatted—format disk using FORMAT command. See chapter 5 on DOS.

b. Write-protect tab on disk—remove tab.

c. Disk improperly inserted in drive—remove disk and insert properly.

Symptom:

Joystick won't work. Has no effect on program.

Probable Causes:

a. Joystick plugged into wrong jack on back of Junior. Unplug joystick and insert it into other *J* jack on back panel.

Symptom:

Internal modem fails to communicate or strange characters appear on screen when using modem.

Probable Causes:

a. Communications parameters improperly set. Check the parameters in effect in the communications program you are using. Use DOS's MODE command or the facility built into the communications software that you're attempting to use.

Various communications systems use various parameter settings. The only important consideration is that both ends of the system use the same settings.

Two of the most common settings are:

Baud rate	300	300
Word length	8	7
Stop bits	1	1
Parity	none	even
Duplex	full	full

b. Modem in wrong mode—answer or originate. Generally, if you dial the telephone, you should use the originate mode. More specifically, when two modems are communicating, one modem must be in answer mode and the other in originate. If not, neither modem will listen to the other one, and communication won't take place. Your communications software controls the modem's mode.

Software Troubleshooting

Junior isn't so bad. It's not out to reduce you to a frustrated hunk of quivering protoplasm when you try to write your first program. Its BASIC is ready, willing, and able to solve your toughest problems. DOS will try to make everything run smoothly. Both programs have built-in aids to help you find errors that you make in programming. If something goes wrong, neither DOS nor BASIC will just desert you and let you wonder what happened. Instead, both will give you an *error message* to tell you what went wrong.

The most common mistakes you'll make in DOS are simple typographical errors and getting disks mixed up. If you don't type your DOS commands exactly the way DOS wants to see them, you'll get an error message—most likely "bad command or file name." If you have the wrong disk in your disk drive when you give the right command, you'll probably get the same error message. Look for the typos first, then check your disks. In case you run into other error messages, check your DOS manual. You'll find a complete listing of error messages, conditions causing them, and the corrective action to take in Appendix A of the manual.

When BASIC encounters a problem like a syntax error, it will not only tell you the line number in which the problem occurs, it will also print out the line number and enter its editing mode so that you can instantly make corrections. With other problems, it may tell you only the line number in which the error occurs, and a few error conditions won't tell you where in the program the error occurs. Often the best strategy is to LIST the program (or just the indicated line) and see if you can figure out what you did wrong, or what Junior thinks is wrong. A complete list of error messages and the conditions that cause them is given in Appendix A of the PC*jr* BASIC manual.

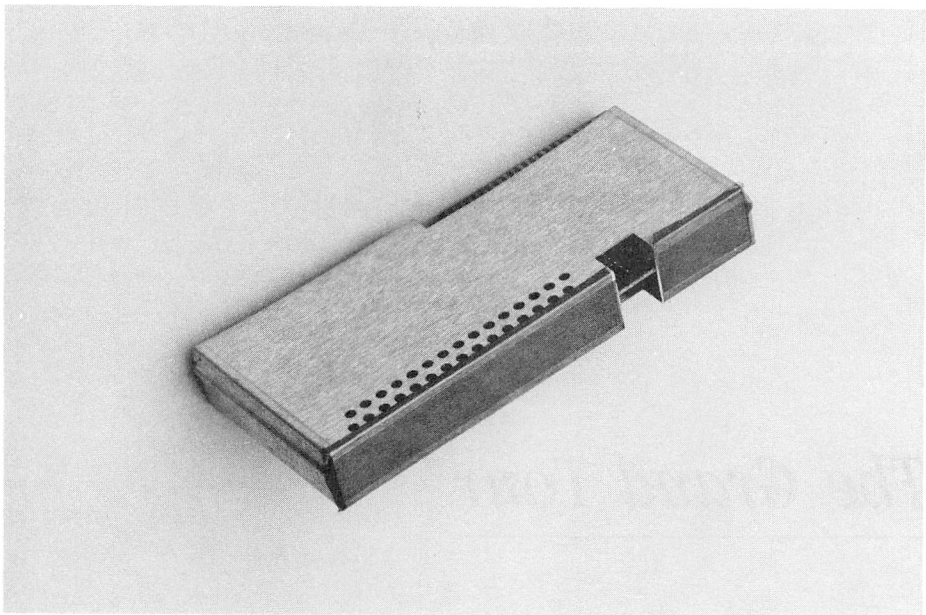
Remember, the error messages presented on the screen are meant to help you find mistakes or potential problems and not make you feel stupid. Learning what error messages mean and the possible problems that they point to can speed your program debugging and get you into error-free computing faster.

The Grand Tour

Most people hold computers in awe and some are even frightened by them, simply because they do not understand how they work. Moreover, they don't understand how they could possibly work. The system unit cabinet is filled with mysterious parts that work even more mysteriously carrying on strange processes almost beyond human comprehension. And if you'd never bought a computer before, you probably wondered when you set out to make your purchase how you could be sure to select the right one.

Even after you've bought an IBM PC*jr* some doubt might remain—did you do the right thing? After all, if you've given any thought to buying a small computer and you've looked around at what's available, you've probably been struck by how similar all the different computers claim to be in their abilities and attributes, and how they all promise the best of everything for everybody. You don't have to be a skeptic to know that all of them can't possibly be the greatest, and more likely, none of them actually is.

What makes a computer, and particularly Junior, good? How do you decide? The problem is that most people don't even know what the words used to describe various computers *mean*, let alone how to compare the values of different terms and numbers. Although they embroider their conversations with all the jargon they can remember from the magazines they've read, few people even grasp the meaning of fundamental terms and why they are important—like the elusive byte. Usually the people



The PC*jr* 64K Memory and Display Expansion allows the PC*jr* to run programs that require more than the standard amount of memory—and gives the PC*jr* its unparalleled graphics abilities.

who banter the terms the most have the least idea of what they're talking about. What can you do to sort sense from the nonsense around you?

Simple—you've got to learn a little about computers, a few fundamentals of the language. Then you have to dig into the machine itself.

The language really isn't hard. In fact, you'll probably be surprised at how easy it is to understand once you cut through all the misunderstandings and misapplications of terms.

The easiest way to understand computers is to use one to learn on. Because you've either bought or are considering buying Junior, it's probably the best example to use. And as a nice side benefit, along the way you'll learn what makes Junior the superior machine it actually is.

Journey to the Center of the System

The operational and philosophic center of almost every small computer is a tiny fingernail-sized chip of etched silicon, an integrated circuit and an electronic brain called a microprocessor. The microprocessor inside Ju-

nior is known to numèrologists and engineers alike as an Intel 8088. The numbers don't really have any meaning in themselves—they just form the name that identifies a particular circuit design.

IBM's choice of that particular microchip for Junior is both remarkable and unremarkable. It's unremarkable because the 8088 is a common circuit that has been used for years. In fact, it is the same processor that has been at the helm of the PC (not a particularly revolutionary computer in itself) for several years. IBM's chip choice is remarkable for the same reasons. Junior is the first application of the powerful Intel 8088 in a computer affordable enough to dress up a kitchen table or line a row of desks in a school or small business. The 8088 microprocessor inside Junior means that for the first time a relatively inexpensive, typewriter-priced computer can share the programs—even many of those written in fast-running "machine" language—with the IBM PC.

To many serious computer users, the number of *bits* a computer uses to think is an important measure of the computer's quality. The microprocessor inside the computer determines how many bits of digital data the machine can process at one time. More bits are better (in general), but more bits also means more costly. The number of bits a computer uses at once must be a compromise between price and power.

All the letters of the alphabet, numbers, and punctuation marks (as well as over one hundred more symbols) can be named or "coded" using 8 bits. Eight bits is enough to code television pictures with quality high enough for network broadcasting. Eight bits is enough to allow the use of BASIC and most high-level computer languages. In other words, 8 bits is the minimum required to handle most things you're likely to want to do with a home computer.

The one most important aspect of using 8 bits of data at a time has nothing to do with pure computer power, however. Integrated circuit chips designed to handle 8 data bits at a time have been around in engineers' minds, hands, and circuit designs for years. They're well proven and often called upon. Moreover, they're so popular that they are made in huge quantities and have become quite inexpensive.

Sixteen-bit computers (the usual step up from 8) can do some things better than 8-bit machines. All else being equal, they can crunch numbers faster—handy if you want to solve three-dimensional simultaneous mathematical equations—but 16-bit power has more down-to-earth, practical value, too. More bits means that a computer can directly address more memory. More memory means that the 16-bit computers can handle huge programs hundreds of thousands of bytes long much faster than 8-bit machines. (That's not to say 8-bit computers can't use really big programs,

but to do it they have to spend precious time shifting portions of programs from one part of their memory to another.) More bits also enables a computer to handle graphics better. Although drawing pictures on a computer screen seems simple, the computer must go through thousands of calculations just to draw the simplest figures. The faster math abilities of 16-bit computers make them able to draw sharper pictures faster than 8-bit machines.

IBM boasts that Junior is a 16-bit computer, but the staunch supporters of other computers will adamantly insist that Junior's 8088 microprocessor only uses 8 bits. Who's lying and who's telling the truth?

They both are. Although the 8088 microchip is often labeled a "16-bit" processor, it deals with digital data as 16 bits only inside the confines of its black epoxy case. To the outside world, it acts as an 8-bit computer, handling data and words exactly 8 bits (or 1 byte) at a time. Because Junior is built from such a switch-hitting microchip, for some purposes it acts like an 8-bit computer and for others it behaves like it has 16. In computer parlance, Junior does internal 16-bit processing and has an 8-bit data bus.

A data bus is the vehicle through which information is put into and taken out of the microprocessor. You can think of it as a real bus that carries the computer's data bits as its passengers, but the bus is really only a set of wires that transports data as electrical signals. In Junior's case, data is moved 8 bits at a time, which requires the use of 8 separate wires. A true 16-bit computer requires 16 wires in its data bus, and with twice as many wires, data can be moved around twice as fast. Because Junior's data bus is only 8 bits wide, programs that push a lot of bits through the bus wires will probably run no faster on Junior than on a plain old ordinary 8-bit computer. Once the data gets off the bus and gets inside Junior's microprocessor, however, it can calculate and think as fast as a 16-bit machine: it can literally think twice as fast as older computers that only think 8 bits at a time.

Nowadays a growing number of microprocessors proudly claim to be 16 bit all around; they both handle data 16 bits at a time inside and wield 16-bit data buses. Why did IBM make Junior—and the original PC—with the strange combination of 8- and 16-bit capabilities? Although it was long rumored before Junior was announced that the new computer would have the more powerful 8086 or newer 80186, both full 16-bit microprocessors, as its brain, IBM stayed with the old 8/16 hybrid. Why?

The same good reason answers both questions. In general, all of the other integrated circuits inside a computer must use the same number of bits as the microprocessor uses for its data bus, so inside both the PC and

Junior all the other integrated-circuit "support" chips that are needed to make a complete computer only need to use 8 bits. Although there's usually only one microprocessor in a computer, there can be dozens of support chips, and 8-bit chips are much less expensive than 16-bit chips. You don't have to be too quick to figure out that a computer with an 8-bit data bus is less expensive to both build and buy than one that uses 16-bit chips.

The clever engineers at IBM had another reason for their circuit choice. At the time the PC was introduced, a large number (in the thousands) of programs written for an all 8-bit microprocessor similar to the Intel 8088 already existed. Restricting the PC to an 8-bit data bus meant that all of those existing programs could be easily translated into the PC's own native tongue. Yet, with 16-bit internal processing, the PC—and now Junior—was more powerful than its computing peers.

The reason for soldering the same Intel 8088 microprocessor into Junior is just as easy to understand. When Junior was introduced thousands of programs written for the IBM PC had already been written, and, because of the identical microprocessors in the two computers, the great majority of them would run *without modification* on Junior. Unlike other new computer designs that enter the world without programs to run, Junior joined us with a cast of thousands.

Junior was by no means the first 16-bit computer affordable enough to use in the home. The late Texas Instruments home computers, the TI-99 series, took that honor with a TI-manufactured full 16-bit microprocessor. Its ultimate failure in the marketplace was probably based more on philosophic than technical considerations. Texas Instruments wanted to be the only supplier of hardware and software for their computer, holding onto total control of the market. As we've seen, IBM has adopted the opposite strategy, called *open architecture.* They have published the complete technical specifications for their entire personal computer family. Consequently, a vast amount of software and hardware made by outside suppliers to add on to IBM computers is available for Junior.

Speed's the Thing

If your only experience with computers has been with cheap home machines, the biggest difference you'll notice when you put Junior to work is how fast it gets jobs done. Compared to most other small computers, and even some business machines, Junior may resemble a

speed demon. It transfers programs to and from floppy disks very, very quickly and races through the most difficult problems.

Then again, if your experience in small computers has been with the older members of the IBM family, Junior often seems like a slowpoke. If Junior were scheduled to race big brother PC, the little computer might as well hang up its sneakers. In many applications Junior will be noticeably slower than the PC, particularly when it uses its disk drive.

Computing speed is not determined solely by the number of bits that a computer uses to think; that's only one of the factors that determines speed. Another major influence is the speed at which the computer's internal clock runs and how many times the clock must tick before the machine thinks of something.

All computers have built-in clocks that control the exact instant each "thought" courses through each part of the machine, how fast each decision is made, and how quickly each number is added. The faster this clock runs, the faster the computer computes, and the harder it is to design and keep working properly.

Rather than ticking off minutes and seconds, these computer clocks run at rates of *millions* of cycles (ticks) per second, or megahertz. The higher the megahertz rating of the computer's clock, the faster it can work its way through computations.

Junior's clock, like those in the rest of the IBM personal computer clan, operates at 4.77 megahertz. Most home computers operate in the range of only 1 megahertz. Although clock speed is only a rough indicator of how fast a computer actually thinks (for instance it takes Junior five clock ticks to have a single thought—other computers may take more or fewer ticks per idea), Junior's higher-hertz design does help make it quicker than other home computers. Junior can cycle through instructions roughly four times faster than the Apple II, for instance.

You may not notice Junior's high-speed approach to computing if you've never used another computer or when you're running short programs. But if you've ever waited eons for a $200 computer to load a disk or run a program, fleet-footed Junior will be a blessing, indeed. Although in most applications Junior still is not fast enough to be able to generate Disney-like animated cartoons, you'll be able to race through the thorniest programs and do things that the dawdling 8-bit computers only dream of.

Although all IBM personal computer clocks tick at the same tempo, Junior will often lag behind its brethren because it devotes some of its thinking time to drawing on its monitor screen. Other IBM machines have special circuits that take over the artistry so the main brain can spend its

time doing real computing. Sharing its thinking time between painting the monitor display and genuine creative thoughts causes Junior to process data about 20–25 percent slower than the bigger, more expensive IBM personal computers.

Junior also slows down when it puts its optional disk drive to use because it lacks a feature called direct memory access (DMA), which the more expensive, more powerful IBM machines have. A DMA is essentially a subbrain that takes control of a computer's disk drive so that the main brain—the microprocessor—can devote its full time to more productive thinking. The DMA-caused speed difference between PC and Junior will be most apparent when you use programs that run the disk drive quite often, such as *WordStar*, which swaps program parts from disk to memory and back quite often because the parts are too big to fit completely into the computer's memory at one time.

A Bit about Memory

More important than the number of bits in the address bus is the number of bits that are used for addressing memory. A computer's memory is arranged into a number of different locations or addresses, each of which is capable of storing a single character (1 byte, or 8 bits) of data. The number of addresses that the computer can recognize determines the maximum memory the computer can handle without resorting to black magic or complex programming. Most home and business computers introduced before Junior and the rest of the IBM family had 16-bit addressing—meaning that the internal microprocessor in those computers could directly address 2^{16} addresses, or 64K of memory. Computers with 8-bit brains must think twice about where things are going to come up with the proper codes for their 16-bit addressing.

Junior and the rest of the IBM family use 20-bit addressing for 2^{20} addresses, or the capability to handle 1,048,576 bytes of information directly. That million bytes of memory handling makes Junior a very powerful computer.

The total addressing capacity of a computer cannot be used by programs. Although a computer might have 64K of addressing ability (and might even have 64K of memory), not all of that capacity is usable for running programs. Often some of the available addresses are used for storing information that is sent to the computer's display and for holding the programs that tell the computer how to run other programs. That's

A Bonus in Every K

About the time you thought computers were designed to straighten out this chaotic world, you probably bumped your head into the term *kilobyte*. If you know anything about the metric system, you know that the prefix *kilo-* means "thousands of." How come a kilobyte is 1,024 bytes?

Computer engineers don't think like normal human beings. Instead of counting on their ten fingers, they count in "base two"—meaning any number over one confuses them—because they deal in digital *bits*, each of which can only be either a one or a zero. Rather than using a number system based on tens where each column of digits represents a power of ten (the way we and the metric people think), computer engineers figure in powers of two. The golden number for the *K* of bytes, 1,024, just happens to be an even power of two (2^{10}), and they thought it was close enough to 1,000 to get away with calling it a "kilo." Besides, most computer engineers would have one heck of a time trying to say "kilodidecaquadbytes."

true for Junior, too. Junior's design allows for only 640K of memory to be devoted to programs. (Although Junior's internal endowment only goes up to 128 kilobytes, add-on modules from IBM and other companies will expand Junior's memory up to that full 640K bytes, ten times more memory for programs than other popular computers like the Commodore 64 and Apple II.)

Digging In

Hobbyists who think nothing of poking around inside a broken color television set (never indiscriminately play around inside a television, as deadly voltage can be present even when the set is turned off and unplugged!) are afraid to peek through the slots of their computers, fearing that an errant touch—or even an errant thought—will destroy the data-processing abilities of its delicate circuits.

Certainly, Junior is not as reassuring as a twenty-year-old television, filled with warm, glowing bottles of electricity that made repair so easy: if a tube stopped glowing, you could pull it out, buy a new one at the drugstore, and plug it in. When you turned the set on, you'd see the

orange sparkle of the tubes as they came to life. Turn on Junior and almost nothing happens. Only if you have a disk drive does it give any outward sign of life, other than on the monitor screen.

But Junior is no finicky creature. It was designed from the start to be accessible, welcoming the fresh air when you pry its lid off. IBM was careful to keep dangerous voltages out of Junior's case, so the highest voltage you'll find is 16 volts, about a thousand times less than the level inside the typical color television set. It also made everything easy to get at, change, and manipulate. You can install any standard internal IBM accessory—including the entire disk-drive system—into Junior with no tools other than your fingernails. And if you exercise reasonable care and a few precautions, you can expect that Junior will suffer no harm from such intrusive surgery.

However, there is a part of Junior that you shouldn't toy with—the rectangular black epoxy plastic microchips and a few other more colorful parts. Though they look the same, nearly every one of these parts is different, and very hard to find and replace. To pull them out, you need a special soldering iron and other tools. If you do manage to get one of these pieces out, you'll never find another one at the corner drugstore.

But thank goodness you probably won't have to. In fact, unless you want to add something to Junior, there's no reason at all for you to crack open Junior's case to see what's inside.

All that said, now's the time to pry Junior open and see what makes it go. You won't see a marvellous collection of interlocked gears, shifting levers, and ratchets on the run. Instead, you'll find the fascinating miracle of modern microelectronics. And though you won't learn how to build one just by peering into its insides, you will learn not to be afraid of your Junior, the high-tech computing machine.

Careful Preparations

For your first venture inside Junior, you'll want to be very, very careful. Although Junior is mechanically rugged, it is susceptible to strange voltages appearing in the wrong places inside its circuitry. The biggest precaution to take to avoid any possible damage is to make sure NO voltages get anywhere near Junior while you're poking around inside. And the best way to do that is to completely unplug Junior from every attachment to the outside world.

Disconnect every peripheral and accessory—even the keyboard cord and television adapter. Make sure there are no cables leading into Junior's back panel. Moreover, make certain that electricity doesn't get anywhere

near Junior after you pry its lid off. Play it safe: unplug the black power-supply brick from the wall. You can take even greater precaution by unscrewing the fuse that powers the outlet that you connect Junior to. Or have the power company turn off all the electricity to your home. Or the whole neighborhood. Or pray for a statewide (or even nationwide) power blackout. Do you get the point?

Clear off a small tabletop. That means no coffee cups around and definitely no cigarettes or ashtrays. You should avoid wearing jewelry, which could fall off and nestle out of sight inside the machine.

The first step is the incision, by which you open Junior up so you can stare inside. Pry off the lid. Junior's top cover just snaps into place. The lid has a front lip that slides under the top of its front panel, so do not start your prying near the front panel. Rather, start to lift the cover at its rear edge, at one or the other rear corner.

Wedge your fingernails in the groove between the cover and the rest of Junior's case. If your fingernails are too short to grip or too long to risk, you can use a coin or screwdriver to pry up the cover. If you decide to use a tool, be very gentle with it. Metal is much tougher than Junior's plastic case and you might damage the case by prying with a metal object.

Once you have a good grasp of a rear corner, pull it slightly upward. It should snap free. Work your way to the other rear corner, lifting slightly as you go along. The cover is held down by three snaps, one in each rear corner and one in the center. Once all three have snapped free, lift the rear of the cover slightly, then slide it backward until the front edge of the cover is free from the lip of the front panel. Now remove the cover and lay it aside.

Note the silvery coating inside Junior's plastic case. Believe it or not, your computer actually has a silver lining! It's really a thin layer of pure silver designed to keep potential radio interference inside the otherwise leaky case.

Like any silver, this coating may discolor with age, turning brown and then black. Although the transfiguration may offend your aesthetic sense, don't worry about the looks. The discoloration is caused by oxidation when the silver metal combines with the oxygen in the air to form silver oxide. The oxide coating works almost as well as pure silver in preventing interference. Don't try to clean the oxide off, as you're apt to scrub off the silver as well, robbing Junior of the interference shield.

Note the contents of the case. You are now staring at Junior in all its glory. Depending on whether you bought the entry or the enhanced model of Junior, you'll see variations on the interior trimming. In the entry model, without enhancements, you'll see a mostly empty case with

one large circuit board lining the bottom and a vertically mounted circuit board on the left.

Enhanced Junior is chock-full of things. You should find at least three circuit boards inside (one will be in a gold-colored metal case) and the disk drive.

For the best view of Junior's works, remove all the inner enhancements. Don't be afraid to pull out any of the circuit cards. You will disassemble and then reassemble Junior as you learn how it works.

No matter what model of Junior you have, first remove the circuit board on the far left, the one with the large black cylinder and wire-covered doughnut mounted on it. It is the power supply card that converts the 16 volts of alternating current supplied by the black power-transformer brick into the direct current that powers Junior's circuitry and some accessories.

If you have an enhanced Junior, first unplug the two cables connected on the card nearest Junior's front panel. The flat, black wire carries power to the disk drive, and the smaller pair of wires carries power to the disk-drive fan.

Now slowly lift the power supply card from its socket. Be careful! Besides the long socket near the front panel, the power supply card is also connected to two thin gold pins that rise from the big bottom circuit board near the rear panel. Be careful not to bend these pins when you remove or reinsert the power supply card.

In enhanced Juniors only, the next circuit card to the right is the memory/display enhancement card. Inside its golden metal shell is 64 kilobytes of random access memory. Remove the module by gently pulling it straight up.

If you have an enhanced Junior and two circuit cards still remain in your computer, or if there is another circuit card in your entry-model Junior, the next circuit card to the right is Junior's internal modem. Remove it by pulling it gently upward. As with the power supply card, be careful of another connection near the front panel. The modem card also connects to two gold pins on the big bottom circuit board near the back panel.

The card furthest to the right in enhanced Juniors is the disk-drive controller card, which holds all the circuitry that controls the operation of the disk drive. A wide ribbon cable connects this card to the disk-drive unit itself. With one hand, hold the card in place and disconnect this ribbon cable by placing your index finger and thumb on each of the short edges of the rectangular black plastic connector on the circuit card and gently pulling to the left, wiggling it slightly, if necessary, to make it part. Next, carefully lift the card from inside Junior.

The last part to remove from inside enhanced Juniors is the disk drive itself. Believe it or not, the disk drive also snaps into its place. To remove the drive, grasp the unit under its rear edge. Slide your fingers behind the disk drive, then underneath. Do not try to remove it by pulling up on any part other than the bottom, as doing so may damage the disk drive. With your fingers underneath, gently lift the rear of the disk drive. It should abruptly snap free. Lift it up further, slowly withdrawing it toward the rear, up and out of the system unit. Put it aside.

You should now be staring at a large printed circuit board dotted with the rectangular black integrated circuits that line the bottom of Junior's case. What you see is the essential Junior. These black integrated circuit chips do all of Junior's heavy-duty thinking. They interpret your commands, manipulate signals, and even generate all the pictures and sounds that Junior makes. Each circuit chip contains the equivalent of hundreds of thousands of individual electronic components all working together to give Junior its elusive electronic thoughts.

Hello, Mr. Chips

This single circuit board actually *is* Junior—it's a single board microcomputer. Everything necessary to make a complete computer is soldered to this one board, no larger than a legal pad. This one circuit board is the foundation upon which you can build an entire computer system.

Each of these integrated circuits has a specific function in bringing Junior to life. Size or shape makes no difference in how vital each one is. All of them are important—the bigger ones are just more complex.

The proper place to begin is with the main brain, the type 8088 microprocessor that gives Junior its versatility and power. If you have problems finding the various integrated circuit chips from the numbers in this discussion, see the box on page 221.

The microprocessor stands tall, one of the few circuits inside Junior in its own socket, so that it can be replaced easily if there is a problem. When you orient your Junior so that its front panel is closest to you, the microprocessor sits halfway back on the far right side of the main circuit board.

A microprocessor works very much like a trained animal. When you give it the proper command in the form of a number representing an instruction in machine language, it responds by either altering electrical signals inside itself or by sending another signal back out to the real world.

A Note on Numbers

Perhaps one of the biggest mysteries of modern science is decoding the strange legends appearing atop integrated circuits. When you go hunting part numbers, you're likely to be confronted by several prospects, none of which looks entirely convincing.

Usually the "generic" identification number of an integrated circuit is surrounded by several letters as part of a coding scheme particular to one manufacturer. Most often, the letters that precede the part number can be ignored. Trailing letters, usually from the very front of the alphabet (like A, B, and C), often indicate updated versions of the breed of integrated circuit.

Most integrated circuits are emblazoned with other numbers, usually four digits long and beginning with 83-- or 84-- (and soon 85-- and 86--). Actually, these numbers are a date stamp that codes when the circuit was made, sort of like the expiration date on a carton of milk. The first two digits of these numbers represent the last two digits of the year the chip was made. The last two digits code the week of the year the chip was manufactured, from 01 to 52.

Usually the microprocessor is instructed to find a certain signal that represents a byte of data somewhere in the electronics connected to it, manipulate that signal inside itself, and then send the result back out to the same place or a different place in the circuitry. A typical machine-language command to the microprocessor might be to find a number in the computer's memory and hold it inside the microprocessor in a temporary storage area called a register. The next command would be to find another number from a different location and add it to the number already in the register. Finally, the result would be sent out to another memory location.

If the microprocessor first gathered two numbers from the Freeboard and sent the result it got back to the screen memory for display on Junior's monitor, the computer would have functioned as a simple adding machine. Besides addition, however, Junior's microprocessor knows hundreds of other instructions allowing it to do all the things you expect a computer to do.

Although a microprocessor is a computing machine in itself, it requires many other circuits to handle signals and put its power to use. The extra circuits on the main board are the "support" chips that help the microprocessor function.

In the right rear corner of the circuit board is the system clock, an electronic subsystem that does nothing but send carefully timed pulses of electricity through the circuit board. The pulses are used to synchronize the operation of all the circuits on the board so that each knows when it is supposed to perform its function. The microprocessor could not think without this supply of carefully controlled pulses.

The shiny aluminum can in this area of the circuit board contains a crystal of quartz that determines the frequency at which the system's clock operates, with the same precision found in a quartz watch or clock.

The white cylinder next to the crystal is a trimmer, which is used to fine-tune the frequency that the crystal operates at to exactly 14.31828 megahertz.

The small black integrated circuit in front of both the crystal and trimmer is called a clock generation chip. Inside it is all the circuitry to make the necessary system-controlling pulses from the vibrations of the crystal, including circuitry that divides the 14.31828 megahertz crystal frequency by 3 to create the 4.77 megahertz operating frequency of the rest of the computer system. (The 14.31828 megahertz crystal frequency was chosen for the crystal so that it could also be divided by 4 to create a 3.58 megahertz signal necessary for the operation of standard color televisions and composite monitors with Junior.)

Diagonally forward and to the left of the clock generator is a larger microchip called an 8253-5 programmable interval timer. This circuit actually has three built-in timer circuits, which are used for separate computer system functions. One timer continuously counts signals generated by the system clock and converts them into new signals that Junior uses to keep track of the time of day. Another of the timers is used to decode the pulses Junior receives from the Freeboard—the timing between the different pulses of infrared from the Freeboard indicates which keys are struck. A third timer is used to change the megahertz frequencies of the crystal clock into a range suitable for use by the sound-generation system.

Between the programmable timer and the microprocessor is an 8259 interrupt controller. An interrupt is a special instruction to the microprocessor that tells it to put everything that it was working on temporarily on hold and take up a new task. When that task is over, the microprocessor picks up where it left off, without missing a beat.

Junior allows nine different reasons for interrupting the microprocessor, each one with its own level of importance and priority. This prioritizing helps Junior organize its thoughts and concentrate its attention on the most pressing matters. The interrupt controller is in effect an interpreter that, upon detecting certain conditions in Junior's circuitry, sends the appropriate interrupt message to Junior.

Junior considers a keystroke at the Freeboard the most important interrupt—no other interrupt can override it. The least important interrupt is the calling of the parallel printer. Although the needs of the printer draw Junior's attention from the normal course of its thoughts, they are superceded by all other interrupts.

To the right of the programmable timer chip is Junior's programmable peripheral interface (PPI) chip, emblazoned with the number 8255A. This special chip continuously listens to the data channel inside Junior. When it hears a command to one of the peripheral devices that it controls on the input/output bus that is shared by the other circuits inside the computer, the PPI sends the appropriate control signal *directly* to that device. In other words, the PPI translates the numberlike digital codes made by the microprocessor into direct on/off signals more compatible with the peripheral devices.

You might have noticed that an active disk drive seems to take priority over your typing even though the Freeboard has been assigned the highest priority among Junior's interrupts. The reason for this confusing result is that the keyboard's high priority can be turned off by the PPI.

The Freeboard is only given top-priority interrupt status *after* it has started sending a sequence of signals that code a single keystroke. The priority of the Freeboard's interrupt lasts only for the duration of those single keystroke signals. Once a keystroke signal is begun, nothing can interrupt it, for its duration of a fraction of a second. If, however, another interrupt has control of the microprocessor first, the PPI prevents any new keystroke from taking control. Consequently you cannot type when the disk drive is operating.

This strange order is necessary so that neither your keystrokes on the Freeboard nor the signals from other devices like the disk drive are misinterpreted. If one of the Freeboard's keystroke signals were to be interrupted in midstream, Junior would misunderstand it as the signal for another key, resulting in an error. Obviously, then, after the keystroke signal begins, it must get the microprocessor's undivided attention until it is finished.

Some other devices require the microprocessor's constant attention, too, because timing of their signals is critical; for instance, the precise timing of pulses from the disk drive determines the meaning of the data sent to the microprocessor. If anything, even a keystroke, interrupted the microprocessor's train of thought during this process, the timing of the signals from the disk drive would be upset, and the data from the disk would be misunderstood. Again the result would be errors.

Directly behind the left-hand cartridge slot lie the read only memory

(ROM) chips that tell Junior's 8088 microprocessor that it really is the brain of a special computer organized to be Junior. Permanently etched inside these two chips are Junior's instructions for carrying out a self-test whenever its power is turned on, the cassette BASIC language program, an operating system for controlling a cassette recorder, instructions for operating the input/output system on the data bus, the diagnostic procedures, enough instructions for Junior to start a disk drive and then get additional instructions from it, and patterns for Junior's assortment of block graphics characters. Each ROM chip has a capacity of 32 kilobytes, for a total built-in ROM of 64 kilobytes.

Another ROM chip lies sequestered behind the infrared-detector module, the silver box right behind the front panel to the left of the two cartridge slots. Encoded inside this chip are the patterns for the dots of every text character that Junior can display on its monitor.

Junior's native endowment of 64K of random access memory is contained in the block of eight identical circuits at the left front of the main circuit board. Each chip holds 64,000 *bits* of data; the eight of them together hold 64,000 *bytes*.

The small, colorful component to the right of each memory chip is a bypass capacitor, which prevents the individual memory chips from accidentally interacting with one another through their common power-supply connection.

The two large integrated circuits in the left-middle of the main circuit board are the largest part of Junior's video display subsystem. The circuit that is farthest back of the two is the video controller itself (number 6845, the same chip that is used in both the IMB PC's color and monochrome display adapters), which changes data to picture information that a video monitor can display. The video gate array is the large chip in front of the video controller. Its job is to control the controller, instructing it which video mode to operate in and keeping track of the border color and other mundane chores.

Directly to the left of the video controller is Junior's sound synthesizer, a relatively small circuit wearing the number 76496. Most of Junior's audio circuitry is tucked over on the left side of the circuit board with this chip, including the strange metal cylinder near the rear, Junior's internal beeper.

A couple of inches to the right of the beeper is a rectangular box about 1 inch by ½ inch and ½ inch high, perhaps bright orange in color. It is a relay that converts the minuscule current of Junior's logic signals into a switch action capable of handling the rather substantial current that controls the motor of a cassette recorder, if you attach one to Junior.

The remaining large integrated circuit, number 8250, controls Junior's

serial port, changing the parallel data on Junior's input/output bus into the serial data used by the Compact Printer and similar devices, and changing serial data coming from the connector into parallel data that can be transferred through the input/output bus to other parts of your Junior system.

Picking up the Pieces—Reassembly

All too often the reassembly instructions for a product are simplified to a single line, "Just reverse disassembly procedure." About halfway through the reassembly reversal you discover step 12 of disassembly was something akin to "scatter pieces to the four winds," which proves to be rather difficult to reverse. Although putting Junior back together is quite a bit easier than reassembling Humpty-Dumpty, some helpful instruction may make the chore easier.

First, reinstall the disk drive. With your hand underneath the rear end of the mechanism, slide the drive down and forward into position, first positioning its faceplate in the large rectangular hole in the front panel designed for it. Make sure that the front of the drive unit rests flush against the back of the front panel.

Now rest the two snap-action plastic support posts, near the rear of the bottom of the disk drive, on the circuit board. Carefully align them with the two holes in the circuit board that they should snap into. Make certain the posts are directly above and centered on the holes, then press the disk drive firmly down, applying pressure only to the plastic frame that supports the disk-drive assembly, first on one side and then the other. Each side should snap itself solidly in place.

Temporarily (and carefully) fold the wide ribbon cable attached to the disk drive across the top of the drive unit.

Snake the flat, black, disk-drive power and fan power cables along the bottom of the case, just above the main circuit board and as near to the rear of the three circuit board connectors as possible. These two cables should be kept below any circuit boards that you install in Junior except for the power supply. Lay the cables down so that their free ends are near where the power supply card will fit.

Next, insert the disk-drive controller card in the rightmost connector slot, making sure the disk-drive power cables remain under it. Line up the bottom rear edge of the card with the slotted support post molded in Junior's case. Press the card firmly into its connector by applying pressure to the top edge of the board directly above the connector.

Now slip the connector on the free end of the disk-drive ribbon cable

over the row of gold pins on the top rear of the disk-drive controller card. (The ribbon cable leads out of the upper side of the connector.) Horizontally center the connector over the gold pins and lightly press it against the pins until you're certain they all line up properly. Put your fingers on the connector and your thumbs on the opposite side of the circuit card and squeeze the connector firmly into place. Be careful not to press the cut-off leads of circuit board components through your fingers.

When replacing your modem card, if you have one, carefully check to be certain that not only does the card fit into its slot but also that the two long gold pins, rising off the bottom circuit board near the rear of Junior, fit into the matching black plastic connector on the bottom of the rear end of the modem card. Once you're satisfied everything lines up properly, press the card firmly down.

If your Junior is an enhanced model, next insert the metal-encased memory/display card into the vacant slot closest to Junior's front panel. The module is designed to fit in only one way—properly. If the module won't fit because Junior's front panel gets in the way, turn the module around, exchanging front for back. Once you have it properly lined up, press it into place.

The power supply card goes in the leftmost slot. When lining this card up with its slot, take care that the two gold pins, rising up from the bottom circuit near Junior's rear panel, slide into the matching black connector on the bottom of the rear end of the power supply card. Once the card is properly lined up, press it into place.

If you have a disk drive, plug its power cables into the connectors on the front end of the power supply card. The white connector attached to the black cable is keyed to fit into its socket only one way. If it won't snap into place, try reversing it. The black connector at the end of the two wires that power the disk-drive cooling fan fits on the gold pins below the white disk-drive connector. The connector is not keyed but it is designed so that as long as all three pins on the power supply card are plugged into the connector, the fan will work properly.

Now reinstall Junior's lid. Slide its front edge under the lip of Junior's front panel and align the back of the lid with Junior's rear panel. Snap it in place, pressing down first on one rear corner, then the middle, and then the other rear corner.

Reconnect your Junior and turn it on. If you've carefully followed these instructions step by step, no one will ever suspect that you've been peering at the works—that is, if you don't brag about what you've done.

Adding to Your System

Handling Hardware

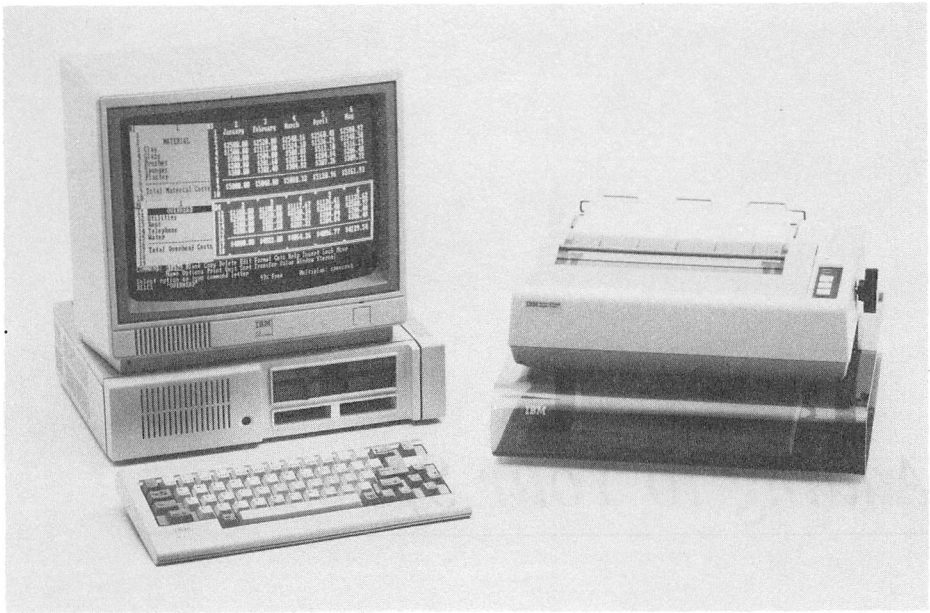
If Junior has one great virtue, it's compatibility. It is unique among the computers that are affordable enough for your living room in that it will run many programs written for the IBM PC.

But that compatibility is not complete, and you're not likely to run into its limits until you own your own machine. Although an amazing number of IBM programs will run on Junior (particularly considering how different from other IBM computers Junior's circuitry really is), many will not.

If you know and understand the limits of Junior's compatibility, however, you can avoid screens of gibberish and a cursor as reluctant to move as a '57 DeSoto *before* you blow your budget on that budget-tracking program.

The rules are simple though the reasons are complex: an enhanced Junior will run almost all programs designed for the IMB PC if all of the following are true:

1. The program requires no more than 96K of memory.
2. The program requires no more than one disk drive.
3. The program is not copy-protected.
4. The program uses (and follows all the rules of) the PC-DOS operating system and probes no deeper than Junior's BIOS (basic input/output system).

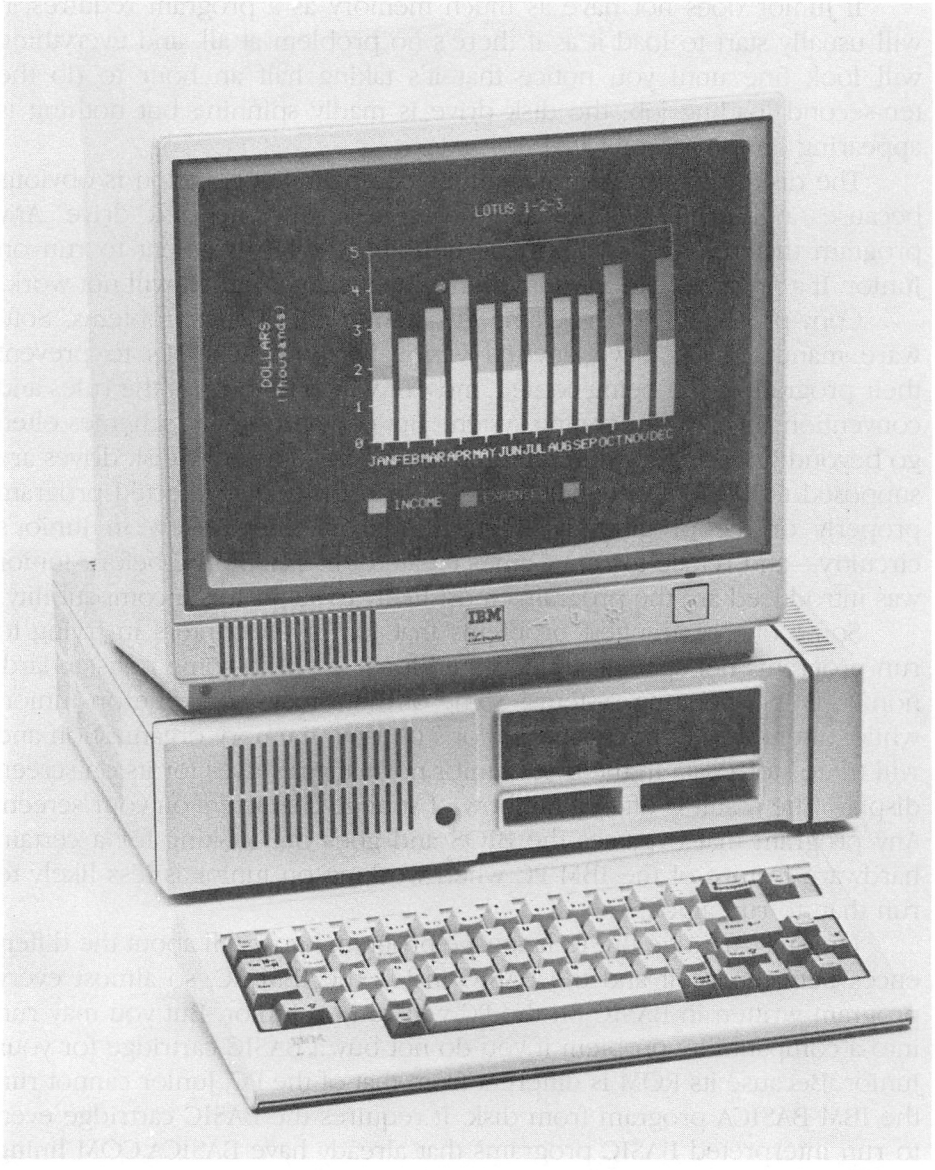


Above and right: While the PC*jr* is the perfect machine for the home, it is also a powerful business computer. Here it is shown running two of the most popular business applications, *Multiplan* and *1-2-3* from Lotus.

Certainly, many programs that do not meet this minimum standard *will* run on Junior, but such programs tread perilously near to the limits of compatibility, perhaps succumbing to an obscure error when you least expect it.

Perhaps the amount of memory that limits Junior's compatibility seems strange to you at first. Everyone knows that an enhanced Junior has a full 128K of working memory, yet programs requiring more than 96K might possibly fail. The reason for this is that Junior's design, which uses part of its working memory for holding video-display information, eats away at that total. Junior's highest resolution, most colorful video display mode swallows up a full 32K of memory, leaving only 96K for programs.

Many PC programs that state a memory requirement of 128K will nevertheless run on Junior, for a variety of reasons. Often the memory requirement of a program is rounded to a convenient memory multiple, and the current use of 64 kilobyte integrated circuits for RAM makes capacity measurements in chunks of 64K reasonably rational. The next step up in memory size from 64K is 128K, so if a program requires half a dozen more than 64K, it will probably need a 128K computer to run. Few programs use every byte of the 128K they claim to need, and those that don't stand a good chance of being within Junior's limit.



Programs written for the IBM PC are not likely to use Junior's best and more detailed graphics mode because at the time Junior was introduced, the PC could not handle such precision graphics. Consequently, programs written for the PC usually use only a lower-resolution graphics mode that will steal, at most, 16K from Junior's working memory. Hence, many programs requiring up to 112K of memory on the PC will run on Junior, barring other compatibility problems.

If Junior does not have as much memory as a program requires, it will usually start to load it as if there's no problem at all, and everything will look fine until you notice that it's taking half an hour to do the ten-second loading job; the disk drive is madly spinning but nothing is appearing on the screen.

The disk-drive problem in Junior's compatibility equation is obvious because, at best, a factory-issue Junior has only one disk drive. Any program that requires two will be unusable or inconvenient to run on Junior. If a program expects something that it cannot find, it will not work.

Copy-protected PC programs give Junior the worst problems. Software manufacturers have devised various devious schemes to prevent their programs from being copied, most of which tiptoe past the rules and conventions of IBM's operating system. But copy-protection schemes often go beyond Junior's very different understanding of the way disk drives are supposed to work, and either it cannot load the copy-protected program properly or the program itself may find something askew in Junior's circuitry—and refuse to run. Games created for the IBM PC before Junior was introduced are the programs most likely to suffer this incompatibility.

Some of the strangest problems that Junior encounters in trying to run programs not specifically designed for it are with some nonstandard, non-PC-DOS operating systems. Some of them may work fine on Junior, while others won't understand Junior's unusual memory organization and will try to store data in the bytes Junior normally reserves for its on-screen display. The result is often a rainbow of strange characters on your screen. Any program that bypasses the BIOS and goes out looking for a certain hardware feature of the IBM PC when working on Junior is less likely to run than to run amok.

Junior's cartridge BASIC language program knows all about the differences between itself and the BASIC inside the IBM PC, so almost every program written in BASIC on the PC will run on Junior. But you may run into a compatibility problem if you do not buy a BASIC cartridge for your Junior. Because its ROM is different from that of the PC, Junior cannot run the IBM BASICA program from disk. It requires the BASIC cartridge even to run interpreted BASIC programs that already have BASICA.COM lining their floppy disks. Without the cartridge, there just isn't any language available to Junior to interpret the program.

A number of interested parties believe that these compatibility problems need not be insurmountable. In fact, many of them have built an industry on making computers go beyond their factory-installed limitations. The personal computer "enhancement" or "peripheral" industry specializes in creating hardware add-ons for computers that extend the capabili-

ties of machines either less expensively or more extensively than the factory allows.

Junior quickly became the target of the machinations of these firms, many of which promised to conquer most, if not all, of Junior's incompatibilities with the IBM PC.

The first enhancements to become available were memory add-ons that plugged into Junior as "sidecars," expansion modules that attach and look like the optional IBM parallel printer adapter. By adding one or two modules, Junior could be enhanced to have the same maximum memory capacity of 640 kilobytes as the IBM PC.

More Memories

As clever as Junior's internal circuit design may be, it gives outside hardware suppliers one big headache. Because IBM designed Junior to divide its RAM memory up between programs and display images, adding memory alone will not give Junior extra capabilities. It won't even know that extra memory has been attached because once it finds its video memory, it stops looking for more.

With the right software, however, Junior can be sent searching for expanded memory. Any add-on memory hardware requires you to run a special program when you turn your computer on, usually as part of an AUTOEXEC.BAT file. Peripheral suppliers therefore must include these programs, which make the additional memory in their enhancement products accessible to Junior. Once the additional memory hardware is properly installed and the installation program run, even the largest IBM PC programs will fit into Junior and run.

Because the programs for modifying Junior's memory are very complex, so fewer companies offer add-on memory for Junior than for the IBM PC. Because hardware components, especially memory circuits, are relatively easy to design, anybody with a basement workshop can become a manufacturer of add-on PC memory. But adding memory successfully requires a good deal of software expertise, so such products are usually offered only by large, well-established firms.

Memory software can also vary in quality, working with some programs but not with others. Before you buy add-on memory, be certain that it will work with the programs that you plan to use. Don't forget to give BASIC a try, too, as it has proved to be particularly troublesome to keep cartridge BASIC working with add-on memory!

Large amounts of add-on memory can bring one very desirable feature to Junior—the electronic or RAM disk, a program that makes

Junior think that a block of its memory is a very fast disk drive. Not only does an electronic disk get around Junior's one-disk limit and make programs such as word processors run faster, it can also prevent Junior's bothersome habit of missing your keystrokes. Cycling through a RAM disk does *not* temporarily shut off the keyboard, so you can keep typing as the RAM disk is working. Suddenly Junior becomes a top-notch word processor.

(Note: remember that anything you store on an electronic disk is only *temporarily* held in electronic memory. A momentary power interruption or turning the computer off will destroy the data. Always save programs and files that are on electronic disk to a real disk before you turn Junior off. If the material on your RAM disk is important, you'll want to SAVE it to a real floppy disk regularly while you work.)

IBM PC*jr* 128K Memory Expansion Attachment

IBM's official means of expanding Junior requires the use of one 128K Memory Expansion Attachment for each 128 kilobytes of memory you want to add to your enhanced Junior. Although a PC*jr* system unit can supply sufficient electricity to operate only one module, up to five IBM 128K Memory Expansion Attachments can be usefully added to a single computer (to bring it to a full total of 640 kilobytes of memory). To plug in more than one memory module, you'll have to use IBM's PC*jr* Power Expansion Attachment between the added memory and the system unit. IBM recommends using the first memory module plug-in next to Junior, followed by the Power Expansion Attachment followed by a maximum of three Memory Expansion Attachments. Although there's more than enough power for four Memory Expansion Attachments, such a configuration had not been FCC certified at the time IBM announced the memory enhancements.

Included with the added memory is the necessary software to let DOS know what you've done, to reserve extra memory for enhanced graphics, and to add a disk emulator.

Microsoft Junior Booster

The Junior Booster is one of the first hardware products from the software company that wrote Junior's operating system. Besides adding 128 kilobytes of extra memory to Junior, the Booster also includes a clock that minds the hours when your computer is turned off and a socket for attaching a mouse.

The software that is supplied with the Booster is the most comprehen-

sive of any add-on memory product. Not only does it include a software patch that allows Junior to recognize more memory, but it also adds a patch for cartridge BASIC (Microsoft also wrote Junior's BASIC program) that fixes some potential problems that the company believes might afflict programs using the added memory. Microsoft's disk emulator is more limiting than most, however, and only allows one memory disk with a small, 91K capacity. (Microsoft Corporation, 10700 Northup Way, Bellevue, WA 98004; (206) 828–8080 or (800) 426–9400)

Quadram QUADMEM jr.

Quadram's first enhancement product for Junior is purely add-on memory, up to 512K, which will extend an enhanced Junior all the way up to its full memory potential. The QUADMEM jr. uses power frugally so no additional power is required *if it is the only enhancement you add to your Junior*. To keep the power requirements low, the QUADMEM jr. uses the latest 256K random access memory chips for its larger memory versions.

Also included with QUADMEM jr. is Quadram's *QuadMaster jr* program, a menu-driven utility that allows you to consign part of the add-on memory to a disk emulator or a printer buffer. (Quadram Corporation, 4355 International Blvd., Norcross, GA 30093; (404) 923–6666)

Tecmar *jr*Captain

If you want to run a memory-munching program and Junior's native endowment of 64K or even optional 128K of memory is not enough, you can pile in more bytes with this expansion module. Designed to handle integrated circuits with either 64K or 256K bits per chip, the *jr*Captain alone can add either 128K or 512K of random access memory to your computer, putting even the biggest programs into Junior's hands. Included with the *jr*Captain is the program necessary to bring the added memory to life, and it works with both cartridge BASIC and Lotus's *1-2-3*. Its electronic disk program and twenty or so other programs give you a start with Junior.

The *jr*Captain also features a parallel printer adapter that works exactly like the IBM add-on module and allows you to connect any IBM-style parallel printer to Junior. There's also a clock with a battery backup that keeps accurate quartz time even when you have Junior switched off or unplugged. With the appropriate command in your AUTOEXEC.BAT file, the *jr*Captain will automatically set the time that Junior keeps in its memory.

Because its circuits draw a good deal of power—more than Junior can spare—the *jr*Captain comes with its own calculator-style transformer that plugs into and powers the module. (Tecmar Inc., Personal Computer Products Division, 6225 Cochran Rd., Solon, OH 44139; (216) 349–0600)

Tecmar *jr*Cadet

If the *jr*Captain isn't full of bytes enough to please you, you can add the *jr*Cadet to stretch your computer's memory to its full 640K addressing capacity. If you buy a Tecmar *jr*Captain with 64K memory chips, it will only hold a total of 128K of memory. The Cadet will add 384K to that, so your tally will become 128K in Junior, 128K in the *jr*Captain, and 384K in the Cadet for the 640K total. The Cadet works only with the *jr*Captain, from which it gets its power.

Other companies supply memory enhancements for Junior as part of add-on disk systems and expansion chassis.

More Disk Drives

A single disk drive may be sufficient for most computing needs but working with one drive is hardly pleasurable. Copying disks takes a seemingly interminable shuffle between the source disk and the target disk. Do it a few times and you'll know exactly how a yo-yo feels. A second disk drive reduces the five-minute copying chore to less than a minute. And many complex programs *require* two disk drives.

There's more to adding a disk drive than just buying a disk drive. You need a complete disk-drive subsystem, which includes a power supply and software. As with memory, by itself Junior is not apt to recognize an extra disk drive: you'll need the right software to make it aware of the addition. Such programs are almost always included with disk-drive systems, except when you purchase a budget-priced disk drive from a mail-order house.

Legacy II

The Legacy puts exactly the same disk drive that most Juniors use, a 360K double-sided, double-density drive manufactured by Qume, in a metal case that matches Junior almost perfectly. The case is sturdy and offers enough interference shielding that you can place it on top of Junior and then put an IBM Personal Computer Color Display on top of it without upsetting Junior's operation.

Although installing the Legacy is easy enough for you to do without a lot of electronics experience, it can be tricky. First, you replace Junior's standard disk-drive controller with one made by Legacy, then run a cable from Junior into the Legacy unit. Another cable attaches to Junior's expansion slot and leads into the Legacy. A dummy plastic module covers up the cables.

Besides the extra disk drive, the Legacy has a front panel full of indicator LEDs that show what's going on inside Junior's circuits. While they don't really reveal anything, they can be fun to watch. Provision has been made for putting expansion memory and other accessories for Junior inside the Legacy case using that company's own system of plug-in cards. (Legacy Technologies, Ltd., 4817 North 56th St., Lincoln, NE 68504; (402) 466–8108 or (800) 228–7257)

Rapport Drive II Enhancement Package

Perhaps the most aesthetic add-on you can buy for Junior is the Rapport disk system. It will make your Junior look like it was designed with two disks. Snap the top off your system unit, drop the Rapport in its place, and put Junior's lid on the Rapport. An extension drops down from the Rapport to mate with Junior's expansion connector. (You can still add additional modules after the Rapport is connected, but such additions ruin the designed-on look.)

Inside the Rapport is a half-height disk drive much like Junior's own. Rather than requiring that you toss out the IBM disk-drive controller card, as you might have to with some add-on drives, the Rapport neatly takes its signals and converts them for use with the second drive. Optionally, enough memory can be added to the Rapport to take your system up to 512 kilobytes of memory. (Rapport Corporation, 80 South Redwood Rd., Suite 213, North Salt Lake, UT 84054; (801) 292–9454)

Expansion Chassis

An expansion chassis adds even more to Junior's versatility. These products allow Junior to run specialty products designed for other IBM personal computers; they hold expansion cards of the same size and style used by those more powerful machines.

Most expansion chassis include both additional floppy-disk drives and a hard-disk, or Winchester, drive. Hard disks can hold ten to a hundred times more data than floppy disks and can load and save that data twenty times faster. Of course, hard disks are expensive: the cheapest units cost about the same as an enhanced Junior. Furthermore, many hard-disk units

don't let you remove and exchange the disks themselves; these units are called fixed-disk drives.

Some expansion chassis add an important feature that Junior lacks, DMA (direct memory access), which allows disk drives and other devices in the expansion chassis to operate as you're typing at the Freeboard.

Alternate Keyboards

Although almost everyone will agree that IBM made a major improvement in Junior when the Freeboard was redesigned with typewriter-style keys, even the new Freeboard may not be to your liking. If you're used to the pleasures of dedicated function keys, Junior's two-step approach probably will prove bothersome. Or the small, lightweight Freeboard just may not inspire your faith; and to do your best work, you should be happy with your tools. After all, what your fingers like is a matter of taste. Keyboard prejudices are not something new that appeared with Junior. Several companies have long manufactured replacement keyboards for other computers, including the IBM PC. Most will point out that the keyboard shipped with any computer must strive to please everyone, no matter whether it spends all its computing hours word processing or shifting numbers in a spreadsheet. But some keyboards are better for word processing, and others are better for number crunching and programming. By selecting one specifically designed for what you plan to do, you'll likely work faster and more efficiently.

Here are some of your choices to replace the Freeboard:

Colby Key-2

When you sell computers, by necessity you must sell keyboards. Colby Computer started by offering a series of portable computers that were more compact than most, and, of course, packed each one with a matching compact keyboard that was roughly 4 inches narrower than the now industry-standard IBM PC design but was equipped with more than a full quota of keys. Obviously eyeing the opportunity in the Junior marketplace, Colby started offering a very similar product, which can plug directly into Junior.

Patterned after the IBM PC keyboard with full-size, typewriterlike keys and snap-action feel, the Colby keyboard features several "improvements" on the original PC design. The left-hand Shift key and backslash are rearranged, the Enter key is turned sideways, and the function keys are arranged in a horizontal row across the top rather than on the left side.

Although the narrow width of the Colby keyboard makes it a better match for Junior than most replacement keyboards, the raised border on each side of the key area may make you feel like you're typing in the Grand Canyon.

The Colby is not cordless; although you can unplug both ends of the 6½-foot cable, it won't work wirelessly. The preproduction model I fingered was not quite up to production refinement, and its snap-in bottomplate invariably plopped out on its own accord. Presumably the models offered for actual sale will be sturdier. (Colby Computer, 849 Independence Ave., Mountain View, CA 94043; (415) 968–1410)

Key Tronic KB-5151*jr*

If there is a Cadillac of keyboards for Junior, it's the Key Tronic. Not satisfied with merely extending Junior's 62-key quota to an IBM PC-like 83 or 84, the Key Tronic KB-5151jr offers an extraordinary 99 full-size, typewriter-style keys, including a separate cursor keypad (with 11 keys instead of the Freeboard's 4), a full calculator-style numeric keypad, 10 dedicated one-press function keys arranged across the top, special "pause" and "reset" keys, and illuminated indicators on the Caps Lock, Num Lock, and Cursor Pad keys.

The Key Tronic keyboard uses state-of-the-art capacitive switch technology, which gives it a lifetime of about 100 million operations but requires that the board be tethered to your Junior by a cord because it consumes a good deal more electricity than batteries could be reasonably expected to supply. The force needed to operate each key is quite light—lighter than the IBM PC keyboard requires and much lighter than the Freeboard. A keypress is pleasantly quiet and your fingertips are treated to tactile feedback so you can feel with certainty when your every keystroke has been completed. The light-touch keyboard can give experienced typists a speed advantage, or just make the job easier.

Key Tronic also sells the eighty-four-key KB-5150jr, a more modest (but still impressive) upgrade for Junior, which is much like the IBM PC keyboard in size and shape but features a slightly refined key arrangement and Key Tronic's light, sure touch. (Key Tronic Corporation, PO Box 14687, Spokane, WA 99214; (509) 928–8000 or (800) 262–6006)

Mice

A mouse is an alternate input device, a means of getting information into your computer without using the traditional keyboard. In most cases

the mouse is a small plastic box with a number of switches (one, two, or three) on it and a position indicator on its bottom. Using a mouse is simple: you roll the mouse across a surface and a pointer moves correspondingly across the monitor screen. When the pointer reaches the screen position that corresponds to a function you want to perform, you press one of the buttons to tell the computer what to do. For many people, using a mouse is more natural than typing. For others, it's just a bother.

If you're a mouse fancier, you have two products to consider connecting to your Junior, one from Microsoft and one from Mouse Systems.

The Microsoft mouse requires its own adapter module, which may be part of the Junior Booster memory attachment. The mouse also requires physical contact with some surface upon which a large ball on its underside rolls. A detector inside the mouse senses the rotation of the ball and relays the information to your computer.

The Mouse Systems product is an optical mouse, which means that it uses a special metal grid-pad upon which it is moved. A red light and a special sensor are used to detect movement across the gridwork and to advise the computer accordingly. Although you must use the special mouse pad with it, you need no special adapter as the mouse plugs into Junior's serial port.

No matter which mouse you choose to use, the mouse itself is not enough. You need software to instruct your computer how to use the hardware. Some software, like PC*jr ColorPaint*, already knows how to use a mouse. Other programs must be adapted. For instance, Microsoft includes special programs with its mouse to allow you to use it with programs like *WordStar* and *Multiplan*. (Mouse Systems Corporation, 2336 Walsh Ave., Santa Clara, CA 95051)

More Power to Junior

Admittedly, Junior's internal power supply is just enough for all of its own electronics. It's got barely a watt to spare for accessories. At most, the computer can handle a single expansion module without help.

The IBM PC*jr* Power Expansion Attachment adds an extra 20 watts of power for whatever peripherals you might want to add. (Specifically, the unit supplies 3 amps at 5 volts, ⅓ amp at 12 volts, and 3/100 amp at -12 volts.) That's enough for three or four additional expansion modules. You can add another Power Expansion Attachment for even greater capacity.

When installing the Power Expansion Attachment be sure that you install it between the system unit and the accessories that it is to power. Otherwise its power won't get to the peripherals that need it.

Speak to Me

You can make your Junior talk to you with the IBM PC*jr* Speech Attachment, which plugs into Junior's expansion slot like any other add-on module. Not only can the module create speech from simple commands (which you can send it through BASIC or any other language), but it can convert your voice, as picked up by a microphone, into a digital form that can be recorded on a floppy disk. Later, the module can reconstruct your voice and play it back. It uses sophisticated technology—linear predictive coding and variable slope delta modulation—to squeeze the digital representation of speech into the minimum possible amount of memory and expand it back to reality with good intelligibility. Although primarily designed for the "Writing to Read" program, the Speech Attachment is available to consumers and programmers who want to develop their own uses for the novel technology.

Clustering

A great deal of time could be saved in an office if all the individual computers on executives' and secretaries' desks could communicate with each other and share both information and peripherals. When the computers used in a business are linked together, they become a very powerful system called a *network*.

Actually the term network is rather vaguely defined; your Junior becomes part of a network when you plug your modem into the telephone line and dial up a database. A more specific term is *local area network* (*LAN*) which refers to the linking of two or more computers in one location so that they can communicate and use common peripherals such as multimegabyte hard-disk drives and high-speed, high-quality printers.

In mid-1984 IBM offered its first personal computer networking system called the Cluster. The Cluster can link, besides IBM's more sophisticated machines, *up to sixty-three Juniors* and a single PC-XT, which supplies the hard-disk drive for system operation. IBM is quick to point out that the Cluster is not designed to be a true local area network but rather to function as an inexpensive entry into networking for small businesses.

After attaching a Cluster module to an entry model Junior's expansion port, adding a 64K memory enhancement and running the Cluster software, Junior becomes a "diskless work station." It shares the hard-disk drive with the PC-XT and other Juniors in the Cluster and functions as if it has three floppy-disk drives of its own. Of the three disk drives, it can write to

only one—the rest contain files shared with the rest of the system and Junior can only read from them. The Cluster lets each computer in the network send files and short messages back and forth as desired.

As networks go, the Cluster is slow, operating just a little faster than a floppy-disk drive does. It is also limited in function because all of the computers in the system must share one disk drive, and they cannot operate other parts of the system (like printers) by remote control.

Junior fits into the Cluster and other more powerful computer networks quite well. When equipped with the proper networking hardware and software, it operates as a moderately priced and very intelligent monochrome or color computer terminal. And while the one-computer family will have little need for a full-fledged network, it's nice to know that Big Blue hasn't consigned your favorite computer to live forever in the toybox.

Software Thoughts

Logo: Junior's Second Language

As with human tongues, there is no one true computer language. You might be tempted by reports about the miracles that can be wrought programming in languages other than BASIC. Should you look into other languages? In most cases, probably not immediately, unless you're an experienced programmer or the language you're considering is Logo.

Known to the world at large mostly for its "turtle graphics," Logo may just become the biggest threat to BASIC's popularity as a language for small computers. (The name Logo is derived from the Greek word for thought, *logos*.) Actually, Logo and the turtle are not the same thing: turtle graphics were originally designed as a revolutionary educational tool based on a robot that moved like a turtle, and Logo was an easy-to-use language that brought the turtle to life. Other languages, like Pilot and Pascal, have recently grown turtles of their own.

As a language, Logo is a generation newer than BASIC and is designed with a totally different concept in mind. While BASIC was written to work in the smallest of computer memories with little regard to its ease of use, Logo was created to be a powerful and easy-to-use language written with little thought to the amount of memory required. BASIC will run in computers with as little as 4 kilobytes of memory. Logo requires much more; in fact, according to Logo's creator, Dr. Seymour Papert of the Massachusetts Institute of Technology, Junior and other IBM personal

computers are the first machines powerful enough to put Logo to practical use in the business world.

The rewards of using Logo can be great: many mathematical functions can be carried out with precision to a thousand places using official IBM Logo; data is stored as "lists" rather than "arrays" (as in BASIC), so they can be easier to handle without the worries and disasters of dimensioning; and programming graphics and almost any other chore in Logo is quicker, more understandable, and often more satisfying than in BASIC.

Logo does suffer its share of problems, however. The versions for Junior are still relatively new and have not been refined to the extent that BASIC has, and all versions of Logo are interpreted, meaning that its instructions are translated into Junior's machine language at the time a program is run. If you're most familiar with Junior's interpreted BASIC, you won't notice much speed difference, but compared to compiled languages like Pascal and compiled BASIC, Logo (and interpreted BASIC) just poke along. No company yet sells a compiler for Logo, which is particularly odd because Logo's parent language, LISP (used in artificial intelligence research) is often a compiled language.

Currently several versions of Logo are available for Junior.

IBM Logo was written by Logo Computer Systems, Inc. (LCSI) of Lachine, Ontario, Canada, and is one of the most powerful realizations of the language to date. LCSI also wrote the "official" Logo languages used by Apple, Coleco, DEC, Texas Instruments, and other computers, and the various versions have a strong family resemblance—giving it a universality that may be its greatest strength. It also has more powerful mathematic abilities than other Junior-sized Logos. The 1.0 version, available at the time Junior was released, requires 128K of memory and a disk drive. A version specially designed for Junior will probably be available in late 1984, perhaps on cartridge. (Available through IBM)

PC Logo, sold by Harvard Associates, is a more compact Logo version (the 2.0 version requires only 64K of memory in the IBM PC, but still needs the disk drive and memory expansion in Junior), which is aimed primarily at education. A special nonprofit school-licensing agreement can be obtained from the publisher for a nominal fee so that the program can be legally shared in the classroom. PC Logo has more than enough power for schools or most business applications and is a little more forgiving in some respects, but its mathematic precision suffers a severe limit of only a few decimal places, about the same as BASIC's single precision. Its editor is superior to IBM's, featuring a search-and-replace facility, and a disk containing utility and demonstration programs and a good tutorial is included with the language. (Harvard Associates, 260 Beacon St., Somerville, MA 02143)

LadyBug is *not* a complete version of Logo, but it is free from its author, David N. Smith (send a disk to the author to get a copy. If you like the program, a $35 contribution is welcomed). A few years ago, Smith read about the educational value of Logo and found no IBM-compatible version of the language available, so he wrote his own version of turtle graphics in compiled BASIC. The commands and syntax of *LadyBug* are essentially the same as Logo's turtle graphics, though the program runs much more slowly. If you want to use Logo only as an educational tool for your children, *LadyBug* is a good, affordable choice. (David N. Smith, 44 Ole Musket La., Danbury, CT 06810; do not telephone)

The current implementations of *Waterloo Logo*, from Waterloo Microsystems, and *Dr. Logo*, from Digital Research, will not run on Junior. Their nonstandard (or at least non-PC-DOS) operating systems conflict with Junior's allocation of RAM display memory.

Other Tongues

Compiled languages can be difficult to program in, but they run fast. The compiler programs that make these languages work are also great big programs, nearly always taking up more than Junior's allotted 128K. Those characteristics have made most compiled languages the realm of mainframe computers or only the most powerful personal computers.

Although Junior can't do the compiling itself, when programs written in many of these languages are compiled to run on the IBM PC with IBM compiler programs, they will run on Junior if memory size, disk drives, and other factors are compatible. So, if you have an IBM PC, you will probably be able to compile programs on it to run later on your Junior.

Of all the compiled languages currently around, the first one that may be converted to run on Junior is Pascal, which has developed a large following among users of other personal computers. Compilers for Pascal run on computers that are not nearly as powerful as Junior.

Pascal developed as a classroom language. To listen to academicians, the biggest problem with BASIC is the one single command GOTO. Loops and jumps can infest a BASIC program like a nest of snakes, tangling themselves and everything else into such a mess that the program cannot be sorted out. One computer scientist in Switzerland decided to do something about the plague of GOTOs and wrote a new language in which they were noticeably absent. His goal was to develop a language to be used solely in the college classroom to teach the students how to write compact, structured computer code. The Swiss scientist, Niklaus Wirth, named his teaching language Pascal, after Blaise Pascal, the seventeenth-century French philosopher, mathematician, and physicist.

Because Pascal is a compiled language, programs written in it are compiled *before* they are run. Pascal's compiler produces instructions for Junior in machine language that can then be run at any time and much faster than interpreted BASIC.

Although Pascal was never meant to be used in the real world for real problems, it soon escaped the classroom and was adapted to genuine problem solving. Several versions are available for IBM computers, but few of the Pascal compilers will work on a factory-issue Junior. The program code generated by a Pascal compiler on a larger IBM computer stands a very good chance of working on Junior, however.

An inexpensive (about $50) version of a Pascal compiler that will run on a plain enhanced Junior, called Turbo Pascal, is available from Borland International, 4807 Scotts Valley Dr., Scotts Valley, CA 95066; orders (800) 227–2400, or in California, (800) 772–2666. (Versions of Turbo Pascal are available for several computers, so be sure to specify that you want one for Junior.)

The chief advantage of Pascal is still that it forces its users to write in a clean, highly structured form. Wirth, noting the wide approval won by Pascal in the working world, has written a new language based very much on Pascal but with functions that make it more useful and easier to work with. The new language, called Modula 2 (yes, there was a Modula 1) has won critical praise, but no version is yet available that will run on Junior.

Applying Application Programs

An *application program* is exactly what the name implies—a program written specifically for one application. That distinguishing mark is perhaps the only thing that application programs have in common. They may be written to suit any purpose, from finding the mass of an electron to playing an eat-'em-up game. They may be written in any language from machine to mumbo jumbo. They may be any length, from a single line to a dozen disks full. And they may be any price, from free to thousands of dollars.

Usually application programs are described by the application that they are to perform. Here are some of the most popular application programs.

Word Processors

A word processor does more than just turn your computer into a typewriter. In the worst case, a primitive word processor is like a correctable typewriter that lets you see your mistakes before you commit them to paper. Better programs allow you to manipulate the words without retyping

them, rearranging sentences and paragraphs just by pressing a few keys.

Word processors are excellent tools for the student whose first draft inevitably becomes the final paper, but they are equally suited to writers who want to go through several layers of revisions without typing each one.

Word processors differ chiefly in three aspects: editing features, which affect the ease with which you can change what you type; formatting features, which include changing margins or adding headings and footnotes; and display features, which depend on the various abilities of the monitor.

Generally, the easier a word processor is to learn, the more tedious it will be in the long run. Many programs are menu-driven and present a list of possible functions from which you choose. The menus make the program easy to learn because you don't have to memorize anything. But menus are a big bother, too. They take up valuable screen space that could be devoted to text, and changing and rearranging menus takes so much time that many menu-driven programs run ponderously slow.

The number of word processors available for Junior is rapidly increasing. Some of the more widely advertised programs include the following. All require a disk drive (unless otherwise noted); most need an 80-column display (monochrome or color).

Bank Street Writer will work with either 40-column or 80-column displays, which immediately marks it as a home/school-oriented product. It will serve either place well, providing you don't write more than about 2,500 words at a time, the maximum file length the program can handle. Although *Bank Street Writer* requires that you switch modes between typing text and editing it, you can use several editing commands in succession and check the results of your work without bouncing between modes. Especially welcome is the way the program handles deletion of large blocks of text. It gives you a second chance to change your mind and lets you move the block to different places to see if it fits somewhere else better. Relatively inexpensive at about $80, *Bank Street Writer* is designed to be easy for children to learn and use and is a good choice for casual letter writing and schoolwork. (Broderbund Software, Inc., 17 Paul Dr., San Rafael, CA 94903)

EasyWriter 1.15 is one of IBM's approved programs for Junior, so you're assured it will work properly. Its manual is exemplary, an IBM PC-style binder filled with an easy-to-follow tutorial. *EasyWriter* makes a few small demands that you'll probably get used to, such as ending lines with musical notes to remind you to press the Enter key and requiring

you to use its favorite file-name extension, EWF, with the files you want to edit. Once you know what you're doing, you can use the whole screen for editing (some programs cover half of it with menus, instructions, and information that you may no longer need or care about), yet still be able to switch to a menu when your memory lapses. Editing, however, is a bit clumsy, and if you need to revise several times to arrive at a presentable draft you may find it quite a chore. The program requires an 80-column display and its blue-on-black scheme in color is particularly dissatisfying. (Available through IBM)

HomeWord is IBM's official 40-column word processor, designed with Junior in mind. It's a more than adequate word processor that is based on the latest computer fad—icons. *HomeWord*'s icons are postage-stamp sized pictures that graphically represent function choices, replacing the menu. Perhaps it's a clever idea, but on Junior it takes a big toll in speed. Using the icons slows down the program so much that a quick typist can get a whole line ahead of the screen display, which will trudge slowly behind, popping one character on the screen at time. Luckily, hypertypists who can remember the list of the program's commands can turn off the icon display and speed the program up, but it still may not prove fast enough. *HomeWord* is not for pros or heavy-duty use—it will handle only about twelve single-spaced pages at a time. (Available through IBM)

IBM Writing Assistant is the word-processing portion of IBM's *Assistant* series of integrated business programs. The word processor itself is primarily based on two other programs, *pfs:WRITE* and *WordProof*, but combines the two to make a relatively easy-to-use word processor with on-line spelling help. You get a blank screen to type on to your heart's content. If you need help because you don't remember a command and don't want to pull out the quick reference card that accompanies the program, just press Fn then 1. If you want to check the spelling of a word, just press Fn then 2; your disk drives will grind and in a few seconds a list of recommended replacements (in which you should find the correct spelling) will appear on the screen. Other functions are just as simple. You can even transfer your work to the other *Assistant* programs to make form letters or to add data to reports.

The chief drawbacks of the program are its copy protection and the speed of the spelling check. Neither of these problems exists when you have a hard disk (like the rest of the *Assistant* series, you can only make one copy of the distribution disk, and that can be to a fast-access hard disk), but when run on Junior, you might find these characteristics

bothersome. With those exceptions, *Writing Assistant* makes a good, business-oriented word processor for letters and reports. (Available through IBM)

PC-Write has a lot going for it. It races through documents faster than most other word-processing programs, has some of the most colorful displays around, and is amazingly adaptable to any keystrokes you want to command it with. What more could you want? Oh sure, you'd want to get it free. Amazingly, you can.

PC-Write calls itself "Shareware." The program boldly announces that you can copy it to your heart's content, but it recommends a contribution of $75 to its author, which also earns you a bound manual and a $25 commission from whoever copies and pays for your copy of the program. *PC-Write* is not flawless, however. It does not give you intimate control of your printer, so even simple commands to underline or boldface text are beyond its comprehension. And you won't know how many pages you've typed until you get ready for printing, because the program is not screen-oriented.

If you're a Pascal programmer, however, you may be in luck. If you register *PC-Write* with the author (that means pay for it), you are treated to the source code and you can then modify the program to your own needs. Few programs, let alone word processors, give you that convenience.

Intrigued? Borrow *PC-Write* from a friend and copy it for yourself. Or, if you're short of friends, send $10 (to pay for the disk and mailing) to Quicksoft, 219 First Ave. N., Suite #224, Seattle, WA 98109; (206) 282–0452.

PIE Writer is one of Apple's favorite word processors translated into IBM form. It's a quirky, powerful program for those who want *complete* control; *PIE Writer* gives you the use of a full 184 commands. Many of these commands take two or three keystrokes to execute on a PC and possibly four on Junior.

If you're not familiar with dot commands, *PIE Writer* will teach you about them very quickly. You need to use them for many formatting functions, such as centering text, making temporary indentations, inserting spaces between paragraphs, and the like.

However, the program gives you little on-screen help to sort through everything that you can do with it. It takes learning. One day will get you started, but you could spend much more time exploring all the possibilities. If you're a perfectionist, however, this program will help you get the most from your mania. (Hayden Software Company, 600 Suffolk St., Lowell, MA 01833; (617) 953–0200)

Select Write is a command-oriented word processor, so you must switch between modes to first enter text and then edit it. That's good in one way, because commands often use logical easily remembered keys. But that's bad, too, as simple functions, like deleting a single letter, might take half a dozen separate keystrokes. Nevertheless, *Select Write* is a full-featured word processor with an almost all-inclusive help facility that can make using it easy initially. It's probably a good choice for occasional letters, but a poor selection for a professional writer. (Select Information Systems, 919 Sir Francis Drake Blvd., Kentfield, CA 94904)

WordStar is the standard by which all other word processors seem to be measured. It gives a good combination of power and ease of use (it has help menus for beginners which can be progressively switched off as you become more proficient at using it). Although many people decry it as hard to learn, a few hours toying with it has proven sufficient to get most people started. The program is quick but slowed down at times by its use of *overlays*, chunks of program too lengthy and little-used to keep in memory all the time that are read from disk as needed. The pauses required to read from disk may prove bothersome. (Remember, you can't type—or do anything else—while the program reads from disk. Too, you must keep a copy of the *WordStar* program files in the disk drive the whole time you use the program.)

A new version of *WordStar*, designed specifically for single-drive Junior (it won't run on any other computer) is to be available around November 1984. The entirety of the new program will be disk resident, meaning no overlay files and no disk swapping. Further, the new version will make better use of DOS's file structure.

Using any version of *WordStar* at first seems formidable, but after a couple of hours of pecking, you can be well on your way to word processing. The program does not have modes—you enter text and edit it with no switches; you just press different keys in combination. That makes it particularly quick to use, although one finger might tend to lodge itself permanently on the Ctrl key. *WordStar* forces no limit to the length of documents you type (except caused by your patience—longer means slower because parts of the file are kept on disk). *WordStar* can also be used to edit programs and to produce standard ASCII files. Its tremendous versatility makes it the top choice of most professional writers. (MicroPro International, 33 San Pablo Ave., San Rafael, CA 94903)

An entirely different program, *NewWord*, copies much of *WordStar*'s style and technique (it was written by some of the same people who

worked on *WordStar*). *NewWord* offers a few refinements over *WordStar*. Chiefly, it's easier to customize and makes better use of the display. For some functions, it's quicker than the older program, but sometimes it's slower. Otherwise, it's a very close clone. (Rocky Mountain Software Systems, 2150 John Glenn Dr., Suite 100, Concorn, CA 94520; (800) 832–2244 or in California, (800) 732–2311)

Databases

A database program is most often compared to a filing system. Like a stack of index cards, a database is a way of organizing information. The electronic advantage is speed in sorting and classifying. In a properly designed database, you can locate all of your lefthanded friends, all of your clients located in certain zip code areas, or all of the blue-eyed blonds in your roster of friends.

Most database programs require that you design a template, essentially the set of labels you'd otherwise place on an index card. The labels are followed by blank spaces of fixed lengths that you specify. To store the information, you then type in whatever names, addresses, or other data you want to fill in the blanks. Each completed electronic card becomes a *record,* and the entire collection of records is a *file.* You can add and remove records in any order at almost any time. To organize the records, you give commands to the program to sort through the file in any appropriate way.

Displaying the individual records properly sorted is sufficient for many purposes; for instance, when you want to look up a client's dental records or the specific gravity of lignum vitae. Often, however, you may want to create a single collation, perhaps as hardcopy, of the sorted information from your files.

The reorganization of your sorted file information, often as a table, is termed a *report.* The generation of reports is probably the most valuable ability of an electronic database. The reorganization and printing of the report requires no added typing except for the few commands necessary to select the information you want to include in the report and how you want it arranged. Computerized databases are a great saving in keystrokes, time, and sanity.

Database management programs that will run on Junior include the following:

dBase II is perhaps the best known database program and the standard to which all are compared. Its abilities are endless but that means

that you must familiarize yourself with a complex set of commands. In fact, *dBase II* can be used much as a record-oriented computer *language* in which you can write lengthy programs. It comes with two manuals in one: a tutorial that helps you get a grasp on the program's features and a reference guide that you can use to explore on your own. Software companies often compare their products to *dBase II*, citing some awe-inspiring improvement. Don't let such comparisons make you think *dBase II* is second-rate. Few applications where Junior is likely to be found actually require more than the 65,000-odd records *dBase II* can handle. Rather, *dBase II*'s chief disadvantage may be its price of several hundred dollars. (Ashton-Tate, 10150 West Jefferson Blvd., Culver City, CA 90230)

PC-File may be the best free program for IBM personal computers, bar none. It gives you a complete database system with more power than you'll ever need. It operates fast and, generally, painlessly. The author has not just created a good program, but he continually improves it, making updates so valuable that it's genuinely worth registering as a user by paying the small price, $35. It has full filing, sorting, reporting, and printing facilities. Its only drawback may be its black-and-white display and 80-column monitor requirement. (Jim Button, PO Box 5786, Bellevue, WA 98006)

IBM Filing Assistant, Reporting Assistant, and *Graphing Assistant* are an integrated database package brought to you by IBM, individually packaged in official IBM binders with official IBM-quality documentation. This trio should be considered as one program, no matter how it's packaged: the *Filing Assistant* section helps you organize and, of course, file, your data, but you need *Reporting Assistant* to generate compilations and, yes, reports of your organizations and reorganizations of data. The *Graphing Assistant* will give you a graphic look at your data. When combined, these programs will handle nearly any job you give them. Add *Writing Assistant*, and you might not need any more software for your small business.

All of these programs are based on the *pfs*: series (*WRITE, GRAPH, FILE,* and *REPORT*) but have been refined by IBM. Although the *pfs*: and *Assistant* programs will perform similarly and give you the same functions, the *Assistant* versions are a better match to the computers (what else do you expect?). (Available through IBM)

Spreadsheets

A spreadsheet is the electronic equivalent of a ledger, and more. Like a ledger, it has rows and columns in which to enter numbers such as daily

receipts and disbursements. But rather than being merely a means of listing numbers, a spreadsheet allows you to manipulate numbers and show their relationships and dependencies.

A spreadsheet can be used to organize data so that you can see the effect that changing one value has on related values.

The classic use for spreadsheets is answering *what if?* questions to make business predictions. Often changing one number in a business situation—like profit margin—can have far-reaching effects on dividends or working capital. By almost simultaneously displaying the effects of the change of any single number in the spreadsheet, it can quickly give insights into related effects.

Today the trend is to combine spreadsheets with other business-oriented software into a single integrated package. The preeminent example of this kind of program is *1-2-3* from Lotus, at the time of this writing the most popular software package for the IBM personal computers.

Lotus's *1-2-3* is a powerful business spreadsheet that combines graphics and database management into a single package. While formerly only available for computers with a minimum of 192 kilobytes of random access memory, Junior's ROM slots allow the program to run in its normal 128 kilobytes of memory. The program comes on two cartridges and takes full advantage of Junior's advanced graphics abilities without losing the power of the original program. In fact, the ROM-based version of *1-2-3* loads faster and seems to execute more quickly than versions running on the bigger IBM personal computers. (Lotus Development Corporation, 161 First St., Cambridge, MA 02142; (617) 492–7171)

Multiplan is the spreadsheet from the folks who brought you DOS. It runs without a hitch on Junior, exactly as you might expect. It gives the expected spreadsheet power and functions, with a few quirks of its own that are designed to make it faster and easier. Although it uses a slightly different number system for its cells than do other spreadsheets, you'll quickly get used to it. Alternately, you can name, in English if you prefer, individual cells or blocks of them and refer to them by those names. Extra functions abound: you can alphabetize columns and open up (and close) up to eight windows in the program. (Microsoft Corporation, 10700 Northrup Way, Bellevue, WA 98004; (206) 828–8080)

ScratchPad is a spreadsheet with a simple on-screen display that does not belie the power behind the program. Its chief asset is its virtual memory feature; even with Junior's modest 128K memory, spreadsheets that are truly huge can be easily created and handled. Once the available

memory in your Junior fills up, *ScratchPad* starts using disk space to hold the excess. However, disk memory is much slower than RAM and you'll end up waiting awhile, particularly when you want to recalculate a huge spreadsheet that's partly on disk. (*ScratchPad* allows you to turn off the calculating function so that the program does not pause to think every time you enter a new number.) Of course, the keyboard is unusable whenever Junior runs its disk drive. *ScratchPad* is command-oriented, and the most help you'll get is the reference card that accompanies its manual. Like most professional spreadsheets, you need an 80-column monitor and you must be willing to settle for black-and-white. (SuperSoft, PO Box 1628, Champaign, IL 61820; (217) 359–2112).

VisiCalc is the father, grandfather, and prototype of all spreadsheets. It has been credited with turning the Apple computer from a hobbyist's toy into a businessman's tool, and it can do the same for Junior. Its business orientation shows; it displays data in black-and-white, 80 columns wide. Although no harder to use than any other spreadsheet, learning *VisiCalc* has proved to be such a challenge for so many people that training software and entire courses of study have been developed to teach its use. Being the first does not necessarily make *VisiCalc* the best, and other spreadsheets have more functions and claim more power. Be that as it may, a program does not last five years or longer without being a powerful and valuable tool. (VisiCorp, 2895 Zanker Rd., San Jose, CA 95134; (408) 946–9000)

Home Financial Software

A computer—even one as powerful as Junior—is not the machine to solve your home financial problems, but it can help you straighten them out. It can also monitor your investments and keep neat tax records. But remember that the computer won't help if you don't cooperate.

Some of the best investment advice you can get now fits into Junior's ROM slot. Andrew Tobias's *Managing Your Money* is an integrated financial-planning package that keeps track of your current income and expenses, insurance, and retirement needs, and investment records (once you enter them in, remember!). The *Managing Your Money* package consists of eight programs that share information: a reminder pad, a budget and check-book tracker, a tax estimator, an insurance planner, a financial calculator, a portfolio manager, and a net worth indicator. Together the programs record, code, and analyze investments, evaluate insurance needs, calculate tax shelter and rental property rates of return, help with tax planning and

suggest optimal tax strategies, remind you of investments going long-term, tally realized (and unrealized) gains and losses, and even print schedule D for your taxes! (Available through IBM)

Don't pin much hope on the other home financial packages, the so-called checkbook balancers. Most of them exist solely because people often need some excuse to justify their spending $1,000 on something that they really wanted. Buying a computer just because you want one is materialism at its worst, but if you buy one to help with your finances, you're demonstrating cleverness, realism, astute business acumen, and a bunch of other things that will make your head swell like you were sucking on a helium tap.

In most cases, should you approach home bookkeeping software with the intent of paying for your computer system with no effort on your part, you'd be sorely disappointed. A computer bookkeeping program can show you how to change your spending habits, but to take full advantage of the computer's help, you've really got to want to change. If you're willing to give the effort, however, your computer *can* help you keep track of your finances and may just improve your budgetary habits enough that you'll treasure Junior more than just for the sake of having it.

Enough with the disclaimer. Following are a number of programs that will help show you what you're really doing with your home finances.

The Checkbook System will help you balance your checkbook, something that your bank does for you free. With Junior, however, you may find mistakes, if you make any. The program will give you a report of your account activity, but remember, you have to enter every financial transaction to make the reports worthwhile. It's quick and easy to use, even if it doesn't do much. You'll need an 80-column monitor to make sense of the on-screen displays. (Creative Research Systems, 1864 Larkin St., San Francisco, CA 94109; (415) 771–0912)

The Electronic Checkbook does the easy part of keeping a home budget—the math. The program will help organize your finances, but warns that it's not an easy job and that the hard decisions remain yours. It won't solve spending problems, but it will help you keep track of them. It will give you reports on your spending and saving but won't track your assets.

The program is easy to use. It's completely menu-driven and the documentation, written in a light vein, does a reasonable job of explaining what it's all about. The program makes quick work of reconciling your checkbook, but it's not a complete, comprehensive money manager. (Cortland Data Systems, PO Box 14414, Chicago, IL 68614; (312) 549–2029)

Home Budget, jr. does exactly what its name implies, helps you plot your home budget on Junior. It will track up to forty-eight separate accounts, probably more than enough for most homes. The program shows its IBM heritage in good documentation and easy-to-use, menu-oriented design with a help screen. The "jr." in its name also means that it is designed for 40-column operation, perhaps on a television set rather than a computer monitor. As a first program to familiarize yourself with both your finances and your new computer, you might not do much better. (Available through IBM)

Home Accountant Plus is a program for people whose home finances may be on the edge of being too much to handle by hand. The program will track up to five different checkbooks and an almost unlimited number of credit cards simultaneously. Besides normal report-generating capacity, the program can also give simple graphic displays of your personal finances. It can handle more complex financial manipulations, but you may find the procedure to be complicated. (Continental Software, 11223 Hindry, Los Angeles, CA 90045; (213) 417–8031)

Education

Education is both the most promising field for software development and one of the most disappointing fields. Most "educational" computer software consists of adaptations of traditional learning methods to the revolutionary computer system, an adaptation that shortchanges both the computer and the kids.

Perhaps the brightest spot in computerized education is the Logo language's turtle graphics, which successfully teach complex concepts of physics to young children in a totally unprecedented manner. Similarly, IBM's *Karel the Robot* teaches the Pascal programming language, and the "Writing to Read" program (used in the classroom, not available to individuals) teaches reading with a wide vocabulary to very young children.

However, too much educational software turns the computer into a drill instructor. The initial novelty may intrigue children into learning, but in the long run it may turn the child's perception of the computer into an unsympathetic task master, making the child the slave of the machine instead of the machine the tool of the child.

The following sampling of educational programs, all distributed by IBM, illustrates the initially available software—some to be enjoyed, some to be avoided:

Adventures in Math (no ages specified) is probably the most creative of the IBM educational game offerings. The contestant(s)—one of two players—search through a castle for treasure, answering mathematical questions to gain treasure or to open doors. Okay, so it's flash cards with imagination, but the graphics are a step above the common IBM educational norm, and the game does take some ingenuity. True adventure game lovers may find the geometry of the castles being explored rather odd; somewhat four-dimensional with rooms changing position occasionally. Also, no time limit is imposed on answering the questions, so there's no incentive to encourage faster thinking.

BASIC Primer 2.0 uses graphics and easy-to-follow examples to teach the BASIC programming language. It follows in the footsteps and philosophy of Logo, using the language that Logo is most often pitted against. Nevertheless, it's an alternate way of learning what must be the most popular personal computer programming language, and it's probably more fun than staring at "syntax error" by the hour.

Bumble Games (ages 4–10) is a set of six educational games that together teach the concept of locating points in a matrix framework by giving coordinates. The games all resemble each other—the concept is the same for all, only the second-rate graphics are changed to separate the different sections of the program. These games are not too exciting, and odds are *Dragon's Lair* and the rest of the crowd from the arcade have made your kids demand more than they will get here. An atlas or roadmap will teach the same concept as is presented in this program, and if you and your kids use your imaginations, that would probably be more fun.

Bumble Plot (ages 8–13) is similar to *Bumble Games*—same concept, different rough drawings. The software seems to follow a trend in modern education in which a subject is taught because it is easy to teach. The matrix/coordinate concept is easy to render in a computerized framework, so that is what this software teaches.

Earth Science Series (teens) is a set of four programs that teach about the processes at work in the environment. Each program uses two interactive tutorials with computer-animated color diagrams. Included in the series are *The Hydrologic Cycle*, which shows how water circulates from land to sea to air and back, including a look at the global water supply; *Ground Water*, which shows the underground life of water in wells, how it gets there and why it stays; *Surface Water*, which explains the movement of water from

stream to sea and the impact of people on the flow; and *Moisture in the Atmosphere*, which shows evaporation and condensation resulting in clouds and weather. Although the programs can be used for individual instruction, they're best as part of classroom instruction, particularly at about $50 each.

Juggles' Butterfly (ages 3–6) teaches the concepts of left and right and up and down, or tries to. The up and down shown on the screen correspond to near and far on the keyboard. And instructions are typed on the screen (for three-year-olds?). A piece of cardboard that you are to cut out is provided for dividing up your keyboard; a genuine IBM-style keyboard overlay is not provided with this software! The graphics are rough and primitive. *The Electric Company* and *Sesame Street* teach the same concepts on television, probably better. Forgo the software and contribute to a local station that carries these programs.

Monster Math is flash cards gone electronic. See how quickly you (or the children) can add, subtract, multiply, or divide. See if you can solve enough problems in a minute to do in a monster so ominous you're tempted to yawn at any minute. The computer handles the drill okay, but flash cards are cheaper—and more portable.

Rocky's Boots (9–adult) teaches elementary computer logic and electronic circuitry by allowing a computer simulation of building machines on the display screen. The object is to create a machine to solve a particular puzzle. Thirty-nine different puzzles are provided, or the players can design their own games.

Miscellaneous Programs

Norton Utilities. A utility program is one that helps you out by making routine chores easier. Usually short, but not always, each utility specializes in one function that helps you or your computer run better. The *Norton Utilities* are programs that help you do things you'd expect to be able to do on Junior but IBM has not given you the software to handle. This valuable set of programs allows you the following functions, among others, under simple menu control: "unerasing" files you accidentally deleted; repairing damaged files; hiding files from the DIR command, and unhiding them; sorting disk directories; changing volume labels; and just seeing what sectors of your disk are used by which files. (Peter Norton, 2210 Wilshire Blvd., #186D, Santa Monica, CA 90403)

Pcjr ColorPaint. If your aesthetic creativity needs an outlet, but you've always been troubled by the fear of making mistakes when you draw,

ColorPaint is for you. Using a mouse to move an arrow across Junior's monitor screen, you draw just as you would on paper. You can choose to use a thin pencil line or a dozen different brushes, or an "airbrush" that lets you shade. Make outlines and fill them in with the press of a single button. Type in titles and labels in a variety of colors and character sizes and shapes. Duplicate patterns or move them around. Best of all, you work in color. And you get the painter's dream; striped or plaid paint is no problem for Junior. You can even print out on paper everything you draw—on any IBM printer (Compact, Graphics, or Color) or on Quadram's inexpensive four-color Quadjet printer.

If there is one program you should get for your idle moments with your computer, *ColorPaint* is it. What you do with it—and Junior—is limited only by your imagination. And that's what a computer is for. (Available through IBM)

Learning Even More about Junior

The concept of user groups is simple—if two heads are twice as good as one, a dozen is six times better than two. A user group is a gathering of computer users who share knowledge and experience. Interest in a single brand or model of computer usually unites the group. Sometimes guest speakers are invited to meetings; the larger groups often lure big names.

Joining a user group is probably the best way to learn about computers and to keep abreast of new developments. Likely you'll find people who have suffered through the same problems with their equipment that you have and can offer lots of worthwhile advice and camaraderie.

Although groups dedicated to Junior are just beginning to spring up, PC-oriented organizations can nevertheless help you along. Many large user groups have libraries of *public-domain software*, programs that are not copyrighted and can be duplicated and used without the bother and expense of licensing. Usually public-domain programs are routines that are written for a specific reason and donated to the club, or just spare-time projects developed by aspiring programmers and shared with pride. Public-domain programs range from the trivial to the esoteric; some well done, others full of bugs.

A big group may publish a catalog of the public-domain programs it has available, everything from homebrew language compilers to spreadsheets. And you may be able to transact business by mail, with a nominal charge for duplication and disks.

Listings of PC user groups nationwide, updated monthly, can be found in both *PC Magazine* and *PC World*. Check your newsstand.

Index

ABOUT THE AUTHOR

WINN L. ROSCH is a contributing editor to *PC: The Independent Guide to IBM Personal Computers* magazine. He is an attorney and a member of the instructional staff at the Cleveland Institute of Electronics. He has written regularly on consumer electronics for *The Cleveland Plain Dealer.* Dr. Rosch is also the author of *The Commodore 64 Survival Manual, Introduction to Digital Circuits* and *Introduction to Microprocessors.*

EVERYTHING YOU NEED TO MAKE THE NEW, ENHANCED IBM PC*jr* YOUR MOST PRODUCTIVE PARTNER.

Acknowledged computer expert Winn L. Rosch opens up the riches of the new, enhanced IBM PC*jr* in this complete, up-to-date guide. He goes beyond the basics to give you the information you'll need to get the most out of this microcomputer, a machine with truly remarkable resources of power and adaptability. Whether you're looking for a problem solver or an office organizer, a number cruncher or a word processor, the IBM PC*jr* has the sophisticated technology and user-friendly design to do the job quickly and easily. This invaluable guide gives you all the facts on:

THE FREEBOARD: How to take advantage of *jr*'s new, vastly improved keyboard.
MASS STORAGE: Using the cassette and the disk drive.
DEALING WITH DOS: A complete overview of the *jr*'s operating system.
SOFTWARE: Which software works best with *jr* for your needs.
BASIC'S BASICS: An introduction to BASIC programming language.
PICTURES AND SOUND: *jr*'s superb color graphics and pleasing audio capabilities.
PRINTERS: How to connect and use your computer with all types of printers.
COMMUNICATIONS: What you need to use modems and databases.
HIDDEN TREASURE: Advanced applications for your *jr*.

AND MUCH MORE, ALL IN STRAIGHTFORWARD, EASY-TO-UNDERSTAND LANGUAGE.

WINN L. ROSCH is a contributing editor to *PC: The Independent Guide to IBM Personal Computers* magazine. He is an attorney and a member of the instructional staff at the Cleveland Institute of Electronics. He has written regularly on consumer electronics for *The Cleveland Plain Dealer.* Dr. Rosch is also the author of *The Commodore 64 Survival Manual, Introduction to Digital Circuits* and *Introduction to Microprocessors.*

N 0-553-34161-8>1295

34161-8 * IN U.S. $12.95 (IN CANADA $14.95)