

# THE COMPLETE GUIDE TO SUCCESS

WITH THE

# IBM

# PCjr

Vernon K. Sondak  
Eileen M. Sondak  
Norman E. Sondak

**INDISPENSABLE!**  
Contains everything you need  
to know to save time,  
money and grief!

**The Complete Guide  
To Success With  
The IBM PC*jr***

# **The Complete Guide To Success With The IBM PCjr**

*Vernon K. Sondak, M.D.*

*Eileen M. Sondak*

*Norman E. Sondak, DEng.*



**Times Mirror / Mosby Publishing**

St. Louis • Toronto • London • Santa Clara

Sponsoring editor: S. A. Newman  
Developmental editor: Liz J. Currie  
Editorial assistant: Lacey Wootton-Kraus  
Manuscript editor: Jerilyn Emori  
Book designer: Nancy Benedict  
Cover designer: Robin Gold  
Illustrator: Scientific Illustrators, Inc.  
Production coordinator: Greg Hubit Bookworks  
Compositor: Graphic Typesetting Service, Inc.  
Printer and Binder: Von Hoffmann Press

Copyright © 1984 by Times Mirror / Mosby Publishing  
First Edition 1984

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission from the publisher.

Printed in the United States of America

**Library of Congress Cataloging in Publication Data**

Sondak, Vernon K., 1957—

The complete guide to success with the IBM PCjr.

Bibliography: p.

Includes index.

1. IBM PCjr (Computer) I. Sondak, Eileen M.


II. Sondak, Norman E. III. Title.

QA76.8.I2593S66 1984 001.64 84-43

ISBN 0-8016-4671-5

Grateful acknowledgment is made to the following companies for the photographs used in this book: BASF Systems Corporation; Electronic Protection Devices, Inc.; FTG Data Systems; Innovative Concepts, Inc. (Flip 'n' File is a trademark of Innovative Concepts, Inc.); Intel Corporation; International Business Machines Corporation; Microsoft Corporation; Novation, Inc.; Perma-Power Electronics, Inc.; Qume Corporation (a subsidiary of ITT Corporation); VisiCalc® screen display reproduced by permission of VisiCorp.

# Contents

Why Should You Buy This Book?	xiii
A Few Words From The Publisher	xvi
<b>1 So Now You Own A Computer: The First Step To Success</b>	<b>1</b>
Computers—What Are They?	3
What Makes The Computer Work?	6
What Are The Major Uses Of A Personal Computer?	10
Further Reading	12
<b>2 Anatomy Of The PCjr</b>	<b>15</b>
Input—The Keyboard	15
Processing—The System Unit	20
Output—The Monitor And Printer	25
<i>Serial versus parallel interface,</i>	<i>32</i>
Storage—Cassette, Cartridge, And Diskette	33
Other Parts	37
<i>The keyboard connection,</i>	<i>38</i>
<i>Joystick,</i>	<i>39</i>
<i>Modem,</i>	<i>40</i>
<i>Light pen,</i>	<i>41</i>
Further Reading	42
<i>Magazines and periodicals,</i>	<i>43</i>
<i>PCjr magazines,</i>	<i>44</i>
 <i>Printers Available For The IBM PCjr</i>	<i>31</i>
<i>Output Selection Guide</i>	<i>33</i>
<i>Printer Comparison Chart</i>	<i>34</i>

**3 Setting Up Your PCjr: Do's and Don'ts** **45**

The Computer Workspace 45  
*The proper power, 47 Furniture and layout  
of equipment, 50*

Insurance 55

What's Left? 56

*Diskettes, 56 Printer paper, 56 Ribbons, 57*

*Dust covers, 57 Cleaning supplies, 57*

*Fire protection, 58 Carrying case, 58*

*Odds and ends, 59*

Setting Up The Computer 59

Further Reading 63



*Computer Workspace Checklist 59*

*IBM PCjr Products Available 60*

**4 Getting Results: Running A Program Makes It Happen** **65**

Turning On The PCjr 66

BASIC 67

*Just follow the blinking light, 69 Taking a  
keyboard adventure, 73*

PCjr Cartridges 75

Writing And Running Your Own Programs 77

Program 1: A Simple Arithmetic Problem 78

Program 2: Metric Conversions 81

Program 3: Guessing Game 86

*Typing tips, 86 Using the printer, 89*

Further Reading 91

*BASIC Commands Coded By*

*Function Keys 75*



*BASIC Keywords Coded By Alternate*

*Keys 87*

**5 Operating The Operating System: It's Easier Than You Think** **93**

The Operating System 93

PCjr DOS 2.10 94

DOS's Duties 95

Loading The Operating System 97

Diskette Directories 100

Formatting A Diskette 103

Diskette Handling 106

Write Protection: Safeguarding Information On Disks	107
Copying A Diskette	108
Verifying The Copy Process	111
Buying A Second Disk Drive	113
The "Phantom" Disk Drive	114
CHKDSK Command	115
External And Internal Commands	117
Starting And Stopping DOS Commands	117
Recap	117
Further Reading	118
<i>Summary Of DOS 2.10</i>	
<i>Commands</i>	97
<i>Disk Drive Checklist</i>	104
<i>Diskette Formatting Procedure</i>	105
<i>Diskette Do's And Don'ts</i>	
<i>Checklist</i>	106



## 6 Let's Get Organized: Setting Up And Using Computer Files 119

Filename And Extensions	121
Preparing A Work Diskette	125
Copy Command	128
Wild Card Filename Conventions: The "*" and "?" Have Their Uses	129
Verify Option	131
COMP Command: Comparing Files	131
ERASE Command	132
Renaming A File	134
Displaying The Contents Of A File	135
Printing What You See And Type	136
Editors	136
EDLIN Commands	137
<i>The Q command: Undoing what you've</i>	
<i>done, 138</i>	
<i>Starting an EDLIN session,</i>	
<i>138</i>	
<i>Line-editing commands, 144</i>	
Recap	150
Further Reading	150
<i>DOS Filename Checklist</i>	122
<i>Examples of Valid And Invalid</i>	
<i>Filenames</i>	123
<i>DOS Filename Extension Checklist</i>	124



*Valid And Invalid Filenames With  
Extensions 125  
Summary Of EDLIN  
Commands 145  
Summary Of Editing Key  
Functions 146*

**7 How Much Is That Software In The Window: Selecting  
And Buying What's Right For You 151**

To Buy Or Not To Buy? 152  
Computer Games People Play 153  
The Big Three Software Packages: Word  
Processing, Spreadsheets, And Data Base  
Management 155  
*Word processing by computer, 156 What  
on earth is a spreadsheet! 158 Data base  
management, 158*  
Where To Buy Software 162  
"Borrowed" Software 165  
Public Domain Software 166  
Picking Your Software Packages 166  
*Documentation, 167 Software user  
friendliness, 168 Software efficiency, 168*  
Information About Software 169  
Further Reading 169  
*Checklist For Computer Games  
Software 154  
Buying Software: Advantages And  
Disadvantages Of Alternative  
Sources 164*




**8 Word Processing, Or How To Make The Eraser Obsolete 171**

Word Processing—What's Out There 173  
*HomeWord, 174 EasyWriter, 175*  
Other Word Processing Options 200  
Other Word Processing Programs 202  
Further Reading 203  
*The Seven Basic Rules For EasyWriter  
Imbedded Commands 196  
EasyWriter Imbedded Commands 196  
Printer Checklist 198*






<b>9</b>	<b>Spreadsheets, Or How To Play The Numbers Game</b>	<b>205</b>
	What "Personal Productivity" Means To You 205	
	Is Using A Computer Always A Good Idea? 206	
	A Closer Look At Spreadsheets 206	
	What Can Spreadsheets Do For You? 208	
	Spreadsheets—What's Out There? 209	
	Getting Started With VisiCalc 211	
	<i>Introducing the VisiCalc spreadsheet,</i>	
	212 <i>Let's see what VisiCalc can do—a</i>	
	<i>sample problem, 217 More work for</i>	
	<i>VisiCalc, 222</i>	
	Parting Thoughts 227	
	Further Reading 228	
		
	<i>VisiCalc Commands</i> 223	
<b>10</b>	<b>Data Bases, Or How To Have Your Own Electronic File Cabinet</b>	<b>231</b>
	What Is A Data Base? 231	
	pfs:FILE And pfs:REPORT 232	
	Designing A Data File 233	
	Entering Data On A File 239	
	Retrieving Data From A File 244	
	Printing Selected Forms 250	
	Removing Forms 253	
	Generating Reports 254	
	Numbers And Calculations With	
	pfs:REPORT 261	
	Further Reading 263	
<b>11</b>	<b>Talking To Your PCjr: BASIC And LOGO</b>	<b>265</b>
	The Art Of Programming 265	
	Programming Languages For The PCjr 266	
	Six Steps To Successful Programs 268	
	Variables In BASIC 269	
	Arithmetic In BASIC 270	
	Logic In BASIC 272	
	Displaying Output In BASIC 274	

	<i>The PRINT statement, 274</i>	<i>The PRINT statement with TAB and SPC, 276</i>	<i>The LOCATE statement, 277</i>	<i>The PRINT USING statement, 278</i>	<i>Hardcopy output, 281</i>
Loops In BASIC	281				
Subroutines In BASIC	284				
Placing Comments And Remarks In The Program	286				
Storing And Using Information With The DATA And READ Statements	286				
Arrays In BASIC	289				
Files In BASIC	291				
Opening And Closing Files	292				
Writing And Reading Data On Sequential Files	293				
Adding Data To Sequential Files	296				
Error Handling	297				
Writing Random File Programs	299				
Games And Graphics In BASIC	304				
	<i>SCREEN statement, 304</i>	<i>COLOR statement, 305</i>	<i>Drawing lines, 306</i>	<i>Random numbers, 306</i>	<i>String and character functions, 307</i>
	<i>An additional note—sound and play statements, 311</i>	<i>More fun and games—The DOS 2.10 supplementary programs, 311</i>			
BASIC System Commands—A Summary	312				
The LOGO Language	318				
	<i>Training the turtle, 319</i>	<i>PCjr draws PCjr, 323</i>	<i>Recursive programs, 327</i>		
LOGO—Not Just Another Pretty Picture	328				
Pascal Revisited	333				
Further Reading	334				
	<i>BASIC Relational Operators</i>	273			
	<i>Screen Statement Summary</i>	305			
	<i>Available Colors And Codes</i>	306			
	<i>Summary Of BASIC Statements</i>	313			
	<i>Summary Of BASIC Commands</i>	316			
	<i>Summary Of BASIC Functions</i>	317			
	<i>Some Important LOGO Primitives</i>	321			



<b>12 Telecommunications—The PCjr's Window On The World</b>	<b>337</b>
The Future Is Now	337
Telecommunications Fundamentals	340
Setting Up Your Equipment—The First Step	346
Communications Software—Your Window On The World	346
<i>Personal communications manager,</i>	
349 <i>CROSSTALK,</i> 355 <i>COMM.BAS,</i> 355	
Information Services	359
<i>The Source and CompuServe—information utilities,</i> 359	
<i>Dow Jones,</i> 361	
Other Encyclopedic Services	362
Public Access Systems	362
Further Reading	363
<i>Telecommunications Settings Checklist</i>	346
<i>Personal Communications Manager Checklist</i>	350
	
<b>13 Success Beyond The PCjr: Where Do You Go From Here?</b>	<b>365</b>
The Future And Your PCjr	365
Hardware Enhancements For The PCjr	367
Software Breakthroughs	368
Beyond The PCjr	370
The Last Word	373
<b>Appendix A Data For pfs:FILE And pfs:REPORT Data Base Example</b>	<b>375</b>
<b>Appendix B PCjr Technical Specifications</b>	<b>379</b>
<b>Glossary</b>	<b>387</b>
<b>Index</b>	<b>395</b>

## ***Why Should You Buy This Book?***

Welcome to the world of personal computers! This book, much like the *PCjr*, is designed for people who have never used a computer before. So, the odds are that if you're reading this, you might be a little nervous about sitting down and working with your *PCjr*. But don't worry: we have written this book with you—and your needs—in mind. We start slowly with very basic concepts of computing and gradually build up your computer confidence (and vocabulary) as we go along. Even if you *do* have prior experience with computers, you'll find our practical approach to the *PCjr* may teach you some things you never knew before. There's an old adage, "Experience is the best teacher." Well, in the area of computers, it ought to be—it's the most expensive. We hope that some of our own costly experiences will save you time, money, and grief as you learn to succeed with your *PCjr*.

We've made this book as practical as possible, developing very special features to make learning about the *PCjr* simple:

- **Helpful checklists** to save you time and money. (To make it easy to find these handy hints, we've marked each one of them with a "time- and money-saving figure." Look for the logo shown at the left as you read the chapters.) These suggestions for using your system and purchasing supplies, components, and programs guide you through the sometimes-confusing world of personal computers.



- **Hands-on exercises** to guide you through your first experiences with the *PCjr*. These exercises cover everything from turning on the computer for the first time to simple programming tasks. Fully illustrated, the exercises let you compare your own computer screen with ours and quickly spot any errors.

- **A complete glossary of computer terms** at the end of the book.

- **Original programs** we have created just for you—programs

## Why Should You Buy This Book?

that not only explain the fundamentals of programming, but provide you with hours of fun.

- **Helpful photographs and drawings** that illustrate exactly what we're talking about in the book. A picture really is worth a thousand words, and we've combined text and art to guide you through learning about the *PCjr*.

- **An introduction to BASIC and LOGO**, two of the most popular programming languages available for personal computers. For those of you interested in writing your own programs, we've explained programming fundamentals to get you started.

- **Complete chapters on word processing, spreadsheets, and data base management programs**—the "big three" software applications you'll most likely use.

One more thing you should know is that this book was written by an author team. We all have the same last name, and, in fact, we're all related (son, mother, and father). But we have quite varied backgrounds—one medical doctor, one writer, and one computer scientist/teacher. This diversity of experience helps us provide you with *all* the information you need to master the *PCjr* without becoming bogged down in material that is too technical. We'll be there every step of the way to make sure your introduction into the world of computers is rewarding and fun.

No book is the result of just the authors' effort. Many people play a vital part in the finished product, and this book is no exception.

We would like to thank Rosemary Morrissey, Jill Liscom, Rick Scott, and other IBM'ers for their help with the technical material on the *PCjr*. We thank Chuck Ericson for his contribution to the programming portion of the book. Thanks are also due to Fred Kellenberger for his astute comments and encouragement. A collective thanks to the numerous vendors and equipment manufacturers who cooperated with us in the preparation of this text and allowed us to use photographs of their products. A special thanks to Ed Hutshing, book editor of *The San Diego Union*, who has always been a great source of wisdom and advice. Thanks again, Ed.

Finally, we owe a deep debt of gratitude to the wonderful and talented people at Times Mirror, particularly Susan Newman for managing the project, Liz Currie for tremendous support in editing and production, and Lacey Wootton-Kraus for editorial assistance.

And now, if you're all set to go, let's get acquainted with your new personal computer. Get ready for one of the great adventures of your life!

Vernon Sondak  
Eileen Sondak  
Norman Sondak

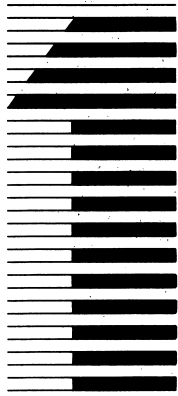
## ***A Few Words From The Publisher***

We are pleased to publish *The Complete Guide To Success With The IBM PCjr*. Owning a PCjr provides you with a marvelous opportunity to discover the pleasures of a personal computer. Whether you plan on using the PCjr for business, home, or educational purposes, you'll soon wonder how you ever got along without it!

We have made every effort to present you with a quality book that will be of value to you for a long time. We assembled a team of authors who have a combined experience of over 35 years of computer education. The authors carefully tested every program in the book on the PCjr. And, as a successful publisher of educational books for over 75 years, Times Mirror/Mosby brings a demonstrated history of publishing experience to the project.

If you have any comments about this book to share with us or with the authors, please contact us at:

4633 Old Ironsides Drive  
Suite 410  
Santa Clara, California 95050  
(408) 748-8588



# So Now You Own A Computer: The First Step To Success

Welcome to the age of computers. If those words fill you with dread, you're probably not alone. But they really shouldn't. A computer is no more threatening a tool than a pocket calculator, because computers are really just bigger, faster, smarter, and more versatile calculators. Remember, the computer is only a machine. It's a powerful machine—the most powerful ever invented—but it is still only a machine, and it can help you in a multitude of different ways.

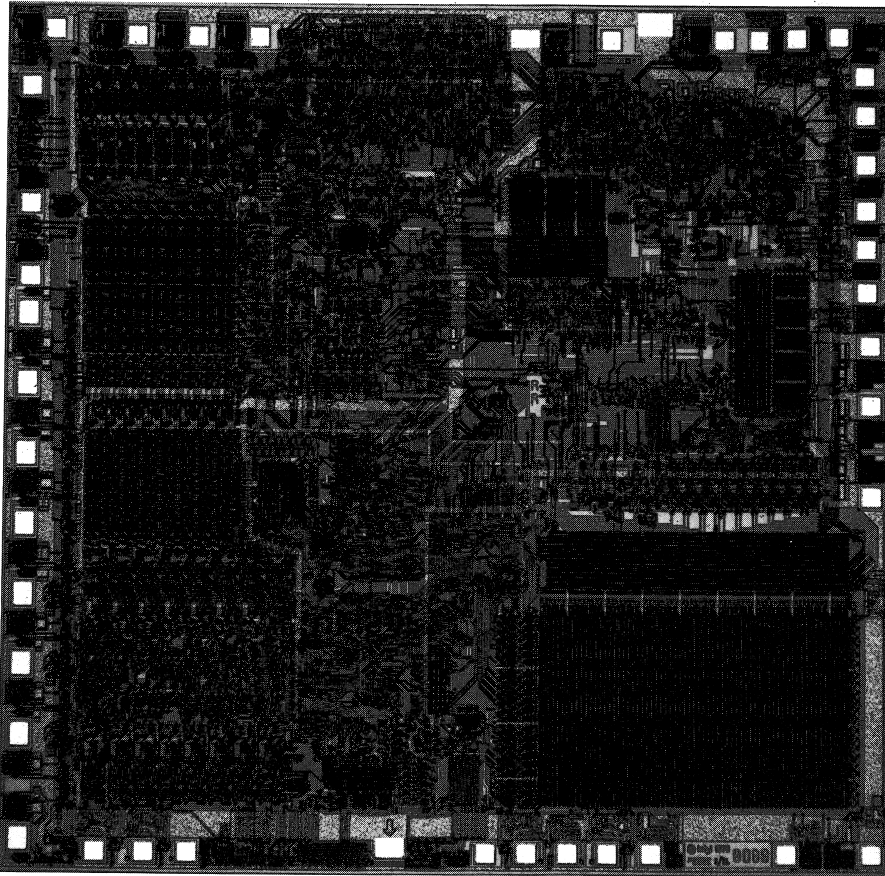
Computers are not all that new. The first computer was built over thirty-five years ago, but your little *PCjr* would look about as out of place next to that first massive computer as the space shuttle would look next to the Wright brothers' airplane. Those early computers were nearly the size of basketball courts, and they consumed so much energy that when the first one was turned on in Philadelphia, the entire city's lights dimmed because of the power drain.

Computers have come an enormous way since those early days. In fact, your IBM *PCjr* is considerably more powerful than that first gargantuan machine. Computers are truly universal machines. Theoretically, their only limit is that the tasks they perform must be stated as a series of steps. And with the advanced state of today's technology, there is practically nothing computers cannot do. The only catch is a human one—people have to understand the problems well enough to transfer them to computer solutions. And that, as you'll find out, often is easier said than done.

Computers monitor the condition of seriously ill hospital patients, do the lightning-quick calculations necessary to keep



spacecraft on course, and even control and design other computers. Today's business computers are small and compact. Their speed and reliability have made them wonders of modern engineering. And even with the phenomenal increase in calculation speeds and data processing capacity, the price of computers has plummeted over the years. Computer scientists and engineers proudly state that if this same progress had been made in the automotive industry, a Rolls Royce capable of 140 miles per gallon would be available at the price of forty cents.



The Intel 8088 microprocessor chip, the "brain" of the PCjr.

Home computers similar to your PCjr have been around for about eight years. There are now more than 4,000,000 home computers, and that number is expected to triple in the next few years. So, as you can see, the age of computers is here—and it is here to stay.

You're going to hear a lot of words like *home computers*, *personal computers*, *microcomputers*. Most of them mean about the same thing. Personal and home computers, for example, are essentially the same in terms of their technological aspects. If your computer is at the office, it's generally called a personal computer. It changes to a home computer when you take it home. Both of these small computers are microcomputers, which means that the entire processing unit is contained on a single "chip" (integrated circuit). But we'll get technical about things a little later on.

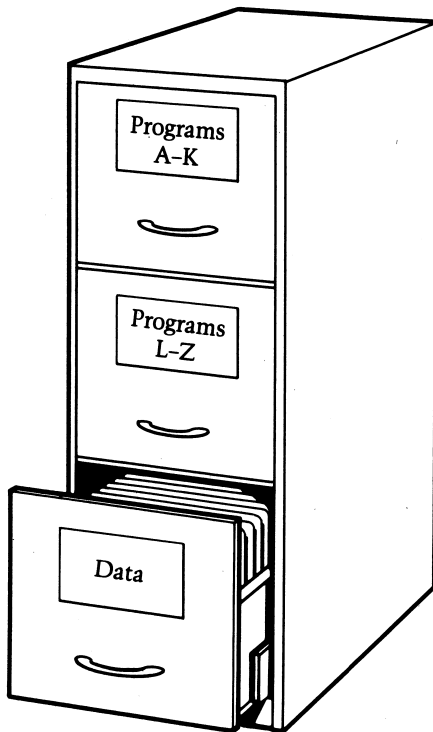
## Computers—What Are They?

A computer is a lot like a big, fast calculator but has one thing most calculators don't—memory. You may say, "Wait a minute, my calculator has a memory." What happens, though, when you turn it off? Most calculators lose the results once the power is turned off. Computers, on the other hand, have the capacity to store information and data for future use.

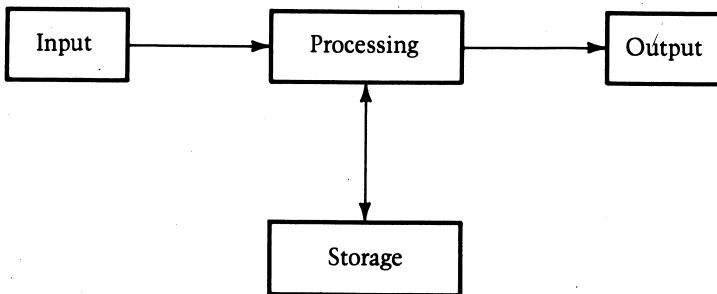
If you understand the concept of memory, you've gone a long way toward mastering your computer. The notion of memory is difficult for some of us to grasp, because we tend to associate it with human faculties, not with machines. The computer's memory is just a place to put facts and figures until you need them. A file cabinet full of papers is a form of memory, and so is this book (Figure 1-1).

Computers have their own internal file cabinets that allow them to store all sorts of facts and figures ("data," in computer talk) until they are needed. Computers, unlike many other devices that hold information, allow rapid and easy access to all the data contained in their memory.

Every computer has four main functions, which define the way the computer performs its operations. Simply stated, these functions are: input, processing, output, and storage (Figure 1-2). *Input* means entering raw facts into the computer. Those facts can



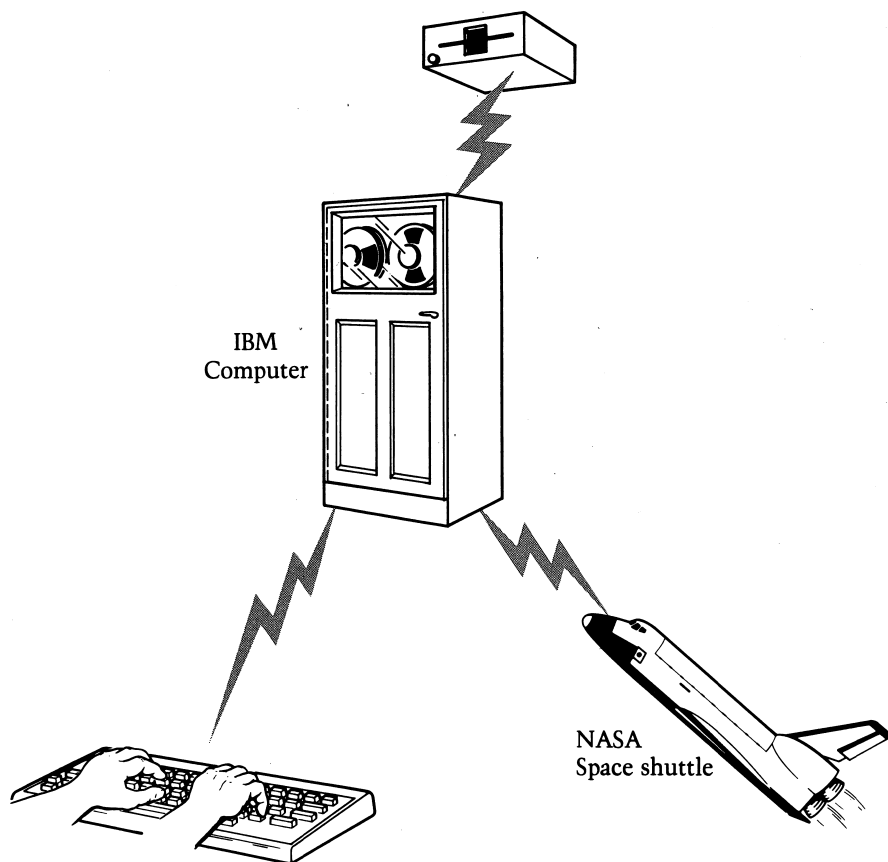
**Figure 1-1** Memory comes in many forms, but to the computer it is just a convenient place to store and retrieve data.



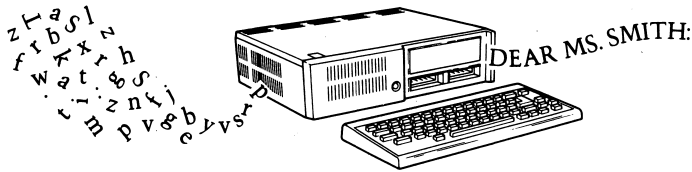
**Figure 1-2** The four main functions of any computer are input, processing, storage, and output. The central processing unit is responsible for coordinating these interrelated functions.

come from a person typing numbers at a keyboard, from a file on a "floppy disk," or from an instrument on a spaceship's exterior. Figure 1-3 illustrates this variety of input sources.

*Processing* is converting the input data into a more usable form (Figure 1-4). For example, the instrument on the outside of that spaceship may be taking measurements (a form of data) of the sun's position, and the computer may be processing that information into the spaceship's position and—by comparing the current



**Figure 1-3** Input is the acquisition of raw facts for the computer to process. These raw facts can come from numerous sources.



**Figure 1-4** In electronic data processing the computer turns raw data into usable information.

position to the position one minute earlier—the ship’s speed and direction as well.

*Output*, quite logically, is the tangible result of the processing operation, whether that data is shown on the spaceship’s display screen, on display screens at Mission Control, or even on your own home video monitor.

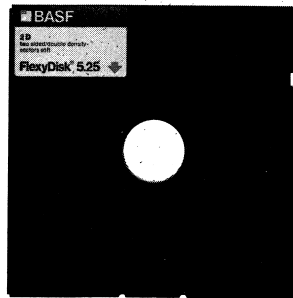
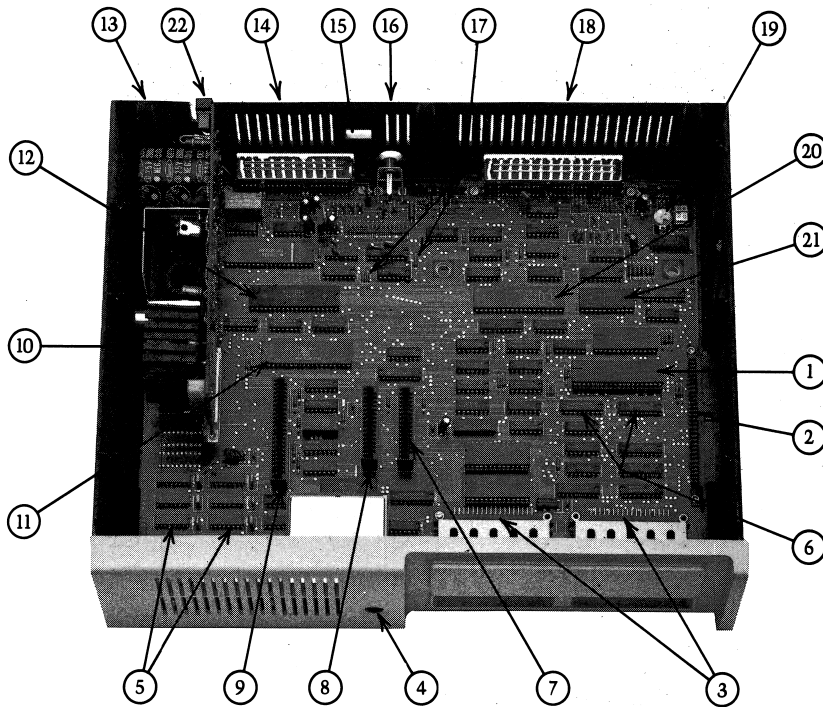
Last is *storage*, the part of the computer that holds the results for later use—for instance, for recalculating the spaceship’s speed and position again. Storage is another, better, way of saying memory. Simple, isn’t it?

Of the four functions the most crucial, of course, is processing. That’s why computers are called data processing machines. Inside the computer all data is stored in the form of electronic pulses. To the computer, processing is nothing more than adding, subtracting, sorting, storing, and retrieving data in the form of these electronic pulses, and reporting the results. With these four basic operations the computer can perform any required task quickly and accurately.

## What Makes The Computer Work?

How does a computer do all these things? To answer that question, we have to understand just what a computer is. In a nutshell, a computer is a system. A system can be defined as a collection of parts or “components” arranged together to perform a specified task. The task of the computer, as we have seen, is to process data.

A computer system’s components are the hardware and the software. The actual physical pieces of equipment—such as the keyboard, disk drive, printer, and so forth—are known as the *hardware*. Hardware is general-purpose equipment. What this means is that anyone with an IBM PCjr can do nearly any type of data



Components are: (1) Intel 8088 microprocessor—the System CPU, (2) I/O expansion connector, (3) Cartridge connector, (4) Infrared receiver, (5) RAM memory chips, 64Kb total of 8, (6) ROM memory chips, 32Kb each. Contain start-up and diagnostic programs, I/O routines, cassette BASIC, keyboard adventure, and other functional programs, (7) Diskette drive adaptor connector, (8) Internal modem connector, (9) 64Kb memory expansion connector, (10) Power board, (11) Video graphics chip, (12) Asynchronous communications chip for serial port, (13) On/Off switch, (14) Connector strip—cassette, serial port, direct drive video, ports (left to right), (15) Hold for modular telephone connector to internal modem, (16) A composite video connector, (17) Grand pins for internal modem, (18) Connector strip—television light pen, keyboard spare connector, joystick one, joystick two (left to right), (19) The system cycle clock, (20) Input/Output interface controller chip for input/output expansion, (21) Real time clock chip for time of day, and sound generation control, (22) A connector for transformer power cord.

Internal memory for the IBM PCjr is provided by a series of specialized chips that plug into the main or “mother board” of the computer. External storage is provided by diskettes (shown here), cartridges, or cassette tapes.

processing, whether it's word processing (such as writing this book), balancing the budget, keeping a recipe file, or calculating income taxes.

In contrast, *software*—the programs or instructions used by the computer—is task-specific. WordStar, for example, is the name of the word-processing program with which this book was written, and it is an excellent program for that purpose. But you would not want to use WordStar to balance your budget or do your tax returns. Instead, you would use programs designed to make financial data processing simple.



"Hardware" is the actual equipment that makes up your IBM PCjr personal computer, as opposed to the "software," or programs.

Software comes in two parts: (1) the program, or the part that the computer can understand, and (2) the documentation, the associated literature that helps people understand how to use the program.

A *program* is nothing more than a step-by-step series of instructions that tells the computer how to do what you want it to do. If a computer could cook (not so farfetched an idea, mind you), a recipe would be a perfect example of a program. The simple, clear-cut steps you follow in a recipe to bake a cake are similar to the steps the computer follows in doing complicated calculations. It wouldn't make much sense to have a recipe that said: "After baking the cake for one hour, go back and add a pinch of salt to the batter before baking." Similarly, the sequence of events is crucial to the success of a program.

Also, being specific and unambiguous is just as important in a program as in a recipe. Can you expect the computer to say, "How much salt is a pinch?" It's up to the programmer, the person who prepares the instructions, to specify exactly what the computer needs to know.

Computer programs, like recipes, can be simple or complex—and can be written by anyone. You probably wouldn't want to try a recipe for baked Alaska, though, from someone who had never even made an ice cream sundae. The same is true for computer programs. Writing programs for tasks as complex as word processing, data base management, or sophisticated accounting is best left to the experts.

Fortunately, expert "chefs" abound in the computer world, and they provide excellent programs for performing specified tasks. These programs are for sale at reasonable (not cheap) prices. And one of the convenient things about these professional software packages, like a recipe from a world-famous restaurant, is that in many cases they can be modified to fit your particular tastes.

The computer user (that's you, if you haven't figured it out) is barraged by a seemingly overwhelming number of programs to choose from nowadays. You must be a careful and wary shopper. The reason for all the programs is all those users—each with slightly different needs, desires, and computer budgets. But software is not cheap, and the costs can mount for programs you rarely use or don't really understand.



## What Are The Major Uses Of A Personal Computer?

We've hinted at, but haven't directly addressed, a personal computer's major uses. A personal computer can be useful in four basic areas: business data processing, home and personal use, education, and entertainment.

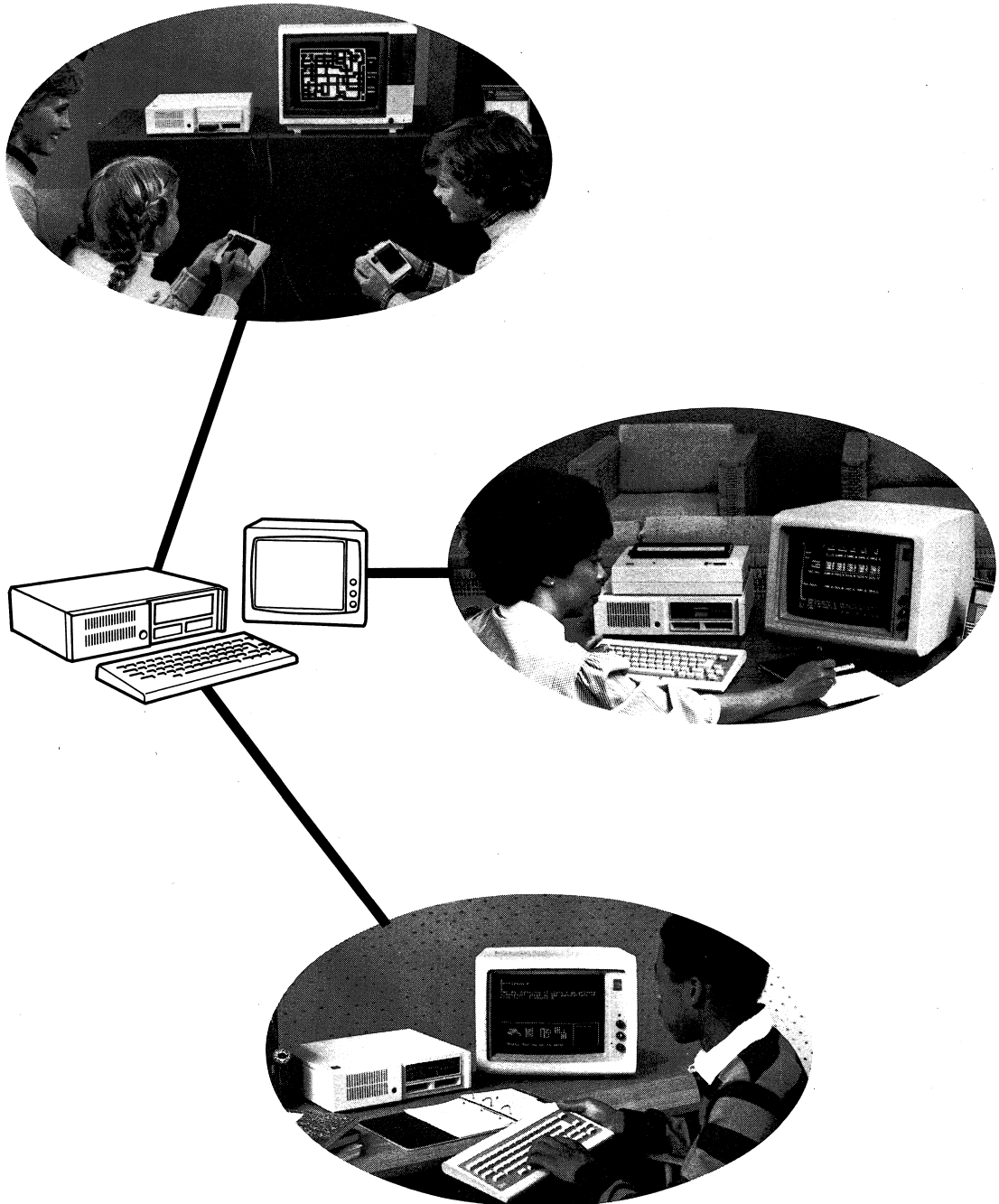
Business applications are the most frequent uses for personal computers. In fact, more than half of all personal computers are sold to businesses. Business uses include accounting, inventory, record keeping, and financial management. Excellent software is available for these purposes.

Home and personal activities, such as writing letters, balancing the budget and checkbook, and keeping recipe files, are an ever-growing area of personal computer use. Although programs are available for such uses, you may find that your personal needs are such that you must write some of your own programs or modify slightly the commercially available programs.

Perhaps the fastest growing field of personal computer use is in education. A superb teacher, the computer can tailor its speed to the pace of the student, without ever losing its patience. More and more new programs for learning foreign languages, spelling, arithmetic, and even programming are becoming available.

Entertainment or amusement is the application for which the personal computer originally was designed. Now computers literally have become "supertoys." When you play games on your computer, you can set the level of competition to your liking. The computer is always a good loser (and an even better winner). It never gets tired and is always ready for a rematch. As you will find, some of the graphics for computer games are magnificent. A number of outstanding games are available, prompting some to dub the home computer the "coinless arcade." And, if truth be told, the personal computer is really the secret toy of many computer owners, including some professionals!

The computer is a patient, accurate, versatile, and absolutely obedient helper—nothing more and nothing less. You have nothing to fear from a computer and everything to gain. To realize the computer's full potential, however, you must learn to use it. The rest of this book will help you learn how to make the most of this incredible machine.



Major uses of personal computers are for business, education, personal effectiveness, and entertainment.



Computers literally have become supertoys, providing countless hours of enjoyment for people of all ages.

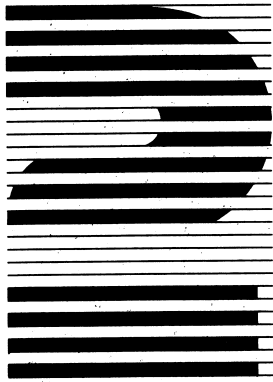
## Further Reading

In those chapters where it is appropriate, suggestions for further reading are included.

*Are Computers Alive?*, by Geoff Simons (Birkhauser, 1983). The author conjectures that computers may be the next step in evolution.

*Artificial Reality*, by Myron W. Krueger (Addison-Wesley, 1983). A view of a world in which computers create an environment of sights, sounds, and sensations of unusual aesthetic beauty. The style of the book is pedantic, but its ideas are fascinating.

- The Computer and the Brain*, by John von Neumann (Yale University Press, 1958). The classic analysis by the genius who conceived the modern computer.
- The Electronic Cottage*, by Joseph Deken (Morrow, 1981). A computer scientist's view of the super home computer of the '80s. Offers some interesting insights into what people can, and will, be doing with home computers.
- Electronic Life*, by Michael Crichton (Knopf, 1983). A well-known author's discussion of how to think positively about computers. Contains excellent advice, but the book's poor organization makes it difficult to use.
- The History of Computing*, by Marguerite Zientara (Computerworld, 1981). A chronicle of the important events in the development of computing equipment, together with a snapshot of the people that made them happen.
- The Intimate Machine*, by Neil Frude (New American Library, 1983). A British psychologist looks at the love/hate relationship between people and their computers.
- The Micro Millennium*, by Christopher Evans (Viking Press, 1979). A critical examination of the future of computers and the computer industry.
- Microman*, by Gordon Pask (Macmillan, 1982). An in-depth analysis of the man/computer relationship, with some predictions for the future of the social impact of computing.
- The Rise of the Computer State*, by David Burnham (Random House, 1983). A "big brother, 1984," view of computers in government and industry. This book will give you added reasons for understanding and mastering your own computer.
- Travels in Computerland*, by Ben Ross Schneider, Jr. (Addison-Wesley, 1974) A novice's comical journey through the early frustrations of large-scale computing systems.



# Anatomy Of The PCjr

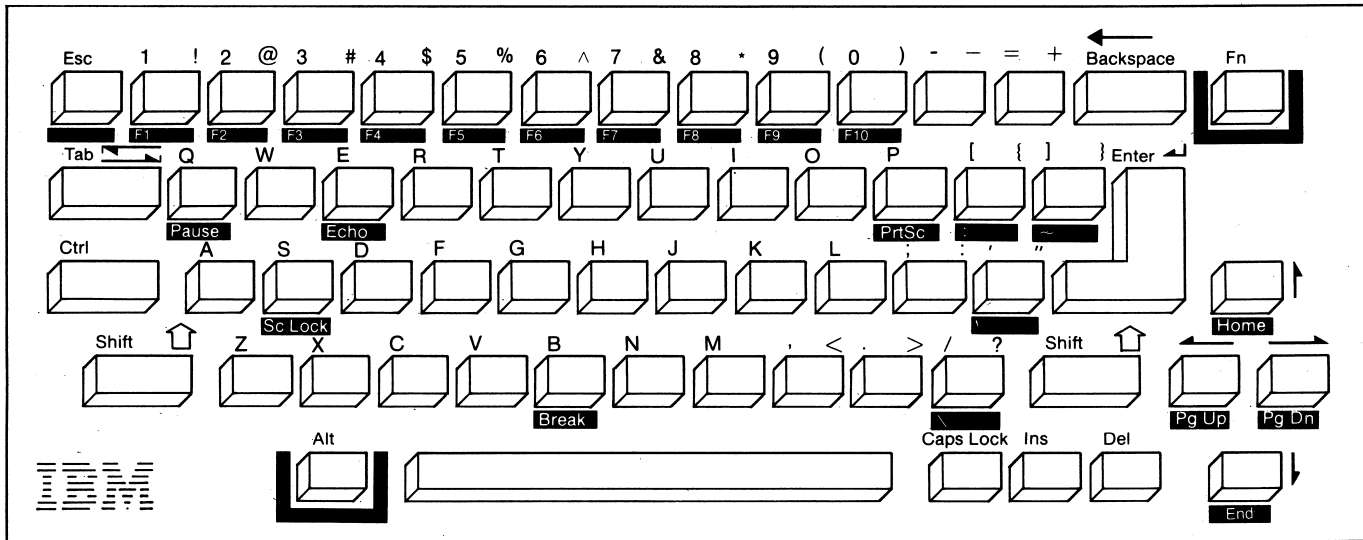
This chapter looks specifically at the hardware that makes up the IBM PCjr computer. In discussing each individual device, we will refer back to the computer's four basic functions: input, processing, output and storage. This will give us a framework in which to work and will allow you to see how each individual device contributes to the system. An understanding of how the parts contribute to the whole is particularly important if you wish to purchase additional hardware for your system.

## Input—The Keyboard

The *keyboard* is the part of your computer that looks like a typewriter. But you might as well know right from the start, it is *not* a typewriter. Perhaps the most obvious difference is your keyboard's special infrared linkup with the rest of the system. We'll talk in more detail about this and other technical aspects of your system later. For now, we're going to look specifically at the "input" part of the keyboard, which entails an explanation of the keys themselves.

As you look closely, you will see that there are a number of keys not found on a typewriter. Also (and we'll talk more about this later), the keys on the keyboard are not always what they appear to be. But enough mystery, let's examine the keyboard (Figure 2-1).

The first thing to notice about the keys on your PCjr keyboard is that the characters are identified on the keyboard surface rather than on the keys themselves as they are on a typewriter. (In this



**Figure 2-1** The major input device for the IBM PCjr is the keyboard. The keyboard uses an infrared cordless connector (powered by 4 AA alkaline batteries) to relay input to the systems unit. The PCjr keyboard has 62 programmable keys. Some are identical to standard typewriter keys; others have different functions that may vary depending on the programs being used.

book we will use boldface brackets [ ] to indicate keys. This will make it easy for you to follow our discussion.) In addition to the character, some of the keys (like [B], [8], and [/]) also have a color code as well. For now, we will concentrate only on the standard keys.

The *alphanumeric keys* (letters, numbers, and punctuation marks) basically are identical in arrangement and function to those on a standard typewriter. For instance, for upper-case letters or for the punctuation marks above the numbers—such as the dollar sign (\$) above the 4 or the asterisk (\*) above the 8—you depress the [SHIFT] key simultaneously with the desired key, as you would on a regular typewriter. Watch out for the [CAPS LOCK] key, though, because it behaves differently from what you might expect.

When the [CAPS LOCK] key is depressed, all letters will appear as capitals,

LIKE THIS.

Numbers and punctuation marks are unaffected, however, so to print a number sign (#) with [CAPS LOCK] activated, you still must depress the [SHIFT] key. The [CAPS LOCK] key needs to be depressed only once and all subsequent letters will be capitalized. If you press the [SHIFT] key while [CAPS LOCK] is activated, the letters will appear as lower case. To deactivate [CAPS LOCK], you must depress [CAPS LOCK] a second time. Pressing the [SHIFT] key does *not* release the [CAPS LOCK] key.

The [CAPS LOCK] and [SHIFT] keys are examples of control keys. The other control keys are the carriage return (also referred to as [RETURN] or [ENTER]), the [BACKSPACE] key, the [TAB] key, the CONTROL key [CTRL], the ESCAPE key [ESC], the FUNCTION key [FN], and the ALTERNATE key [ALT]. Notice that the last four keys are not standard typewriter keys at all. These keys have special purposes, which may vary from program to program.

Three of these keys — [ESC], [ALT], and [FN] — are colorcoded as well. The [ESC] key is colorcoded red, as a warning to the unwary. The [ESC] key can do exactly that—let you escape from a part of the program you mistakenly got into—at least in some programs. However, if you press the key accidentally, you may lose valuable data — thus the red warning color.

The other two keys—[ALT] and [FN], which are blue and green, respectively—operate in conjunction with the other keys of the same color on the keyboard. For example, when the [ALT] key is depressed simultaneously with the [/] (slash) key, a backslash

is produced.

Notice that the backslash is depicted on the surface of the keyboard in a blue area just below the key. The [ALT] and [FN] keys are just different kinds of “shift” keys. That means they shift the meaning of the key being pressed to a new meaning. On the keyboard, the new meaning is designated by the color coding. For example, the backslash is color coded blue and is obtained by striking the [ALT] and [/] keys together. We represent the combination of these two keys in “shorthand” as [ALT/] or [\]—either way is okay. Pressing the [ALT] or [FN] keys along with a key that either is not color coded or is color coded differently will not produce any character and will be ignored by the computer. Figure 2-2 shows all the combinations used by the PCjr for the [ALT] and [FN] keys.

The *function keys* [F1] through [F10], color coded green, are special-purpose, time-saver keys. They are produced by the combination of the [FN] key and the number keys [1] through [0] (for 10) depressed together. Different programs can assign different meanings to these keys. For example, typing [F1] may take the place of typing a string of other characters. These keys speed you up by saving keystrokes and minimizing errors, and they increase the power of the keyboard and the usefulness of the program. Proper use of the function keys requires a knowledge of how they are assigned in a particular program.

Notice the arrows by the four keys in the lower-right corner of the keyboard along with the words [HOME], [END], [PG UP], [PG DN]. These keys are *cursor control keys* and can be used to move the *cursor* (the flashing bar of light that indicates where the next typed letter will appear) quickly around the monitor screen (more on this later). The keys marked [INS] and [DEL], which stand for *insert* and *delete*, are just to the left of the cursor control keys and are also used in typing and word processing.

If all of this is starting to sound too complex, now is the time to take heart. It's easy to use the IBM PCjr keyboard. In fact, in



Alt		=	:
Alt		=	~
Alt	/	=	\
Alt	'	=	'
Fn	B	=	Break
Fn	E	=	Echo
Fn	P	=	Print screen
Fn	Q	=	Pause
Fn	Up	=	Home
Fn	Left	=	Page up
Fn	Right	=	Page down
Fn	Down	=	End
Fn	S	=	Screen lock

**Figure 2-2** Combinations used by the PCjr for the [ALT] and [FN] keys.

many ways it is easier than using a standard typewriter. And probably the best news of all is that the PCjr is extremely understanding and forgiving of typing mistakes. It will usually (but not always) give you a chance to correct those typographical errors before they become a permanent part of the memory.

To demonstrate this feature, you are going to do some simple typing exercises as soon as your system is set up. These exercises will give you a feel for the keyboard and will help you get acquainted with that funny little flashing bar of light called the cursor (soon

to become one of your closest friends). These exercises will convince you how easy it is to correct your mistakes with the IBM PCjr.

But any further discussion of using the keyboard must wait until we have met and assembled the rest of your PCjr computer.

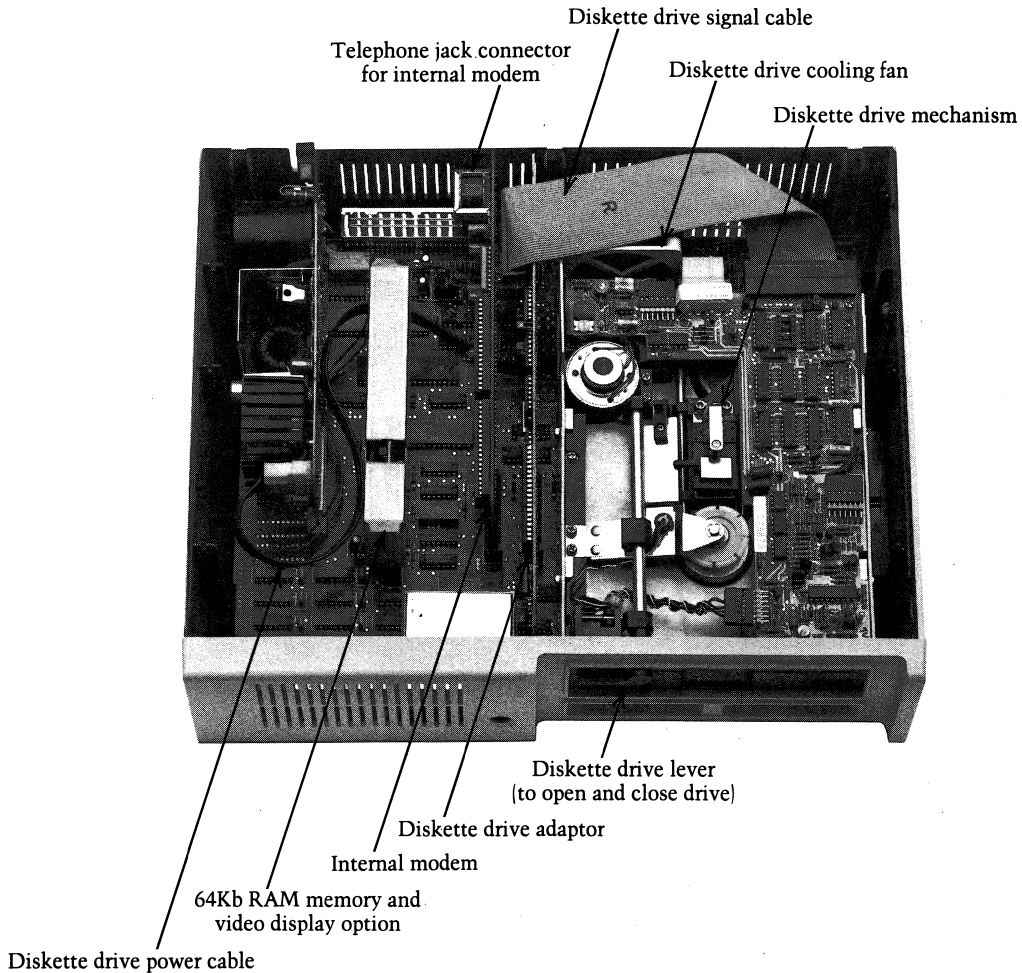
## Processing—The System Unit

Processing, as we've already stated, is the most crucial function of your IBM PCjr. It is this function that really sets your computer apart from a calculator or similar device. Processing is done in the *system unit* (often abbreviated SU). Your PCjr system unit is made up of an outer case, an enclosed power supply, a printed circuit "mother board" (a base board to which everything is attached), and the various "cards" or "boards" that also are made of printed circuits. A *card* or *board* in computer terminology refers to a laminated plastic unit on which dozens of circuits, chips, and wires are embedded. These boards are self-contained modules that plug into slots inside the system unit. Each serves a special function or functions.

Personal computers are designed in separate pieces for an important reason—expandability. If you want to use a device or function that your computer is not capable of handling, you don't have to change your entire system. You just add the appropriate module—for example, a memory expansion module—that allows your computer to adapt.

A good example of a function that many computers cannot perform right out of the box is *graphics* (which is nothing more than the ability to draw pictures and graphs on your computer). All you have to do to upgrade your system for this new task is to open the back of your system unit and plug in a graphics card. Simple, right?

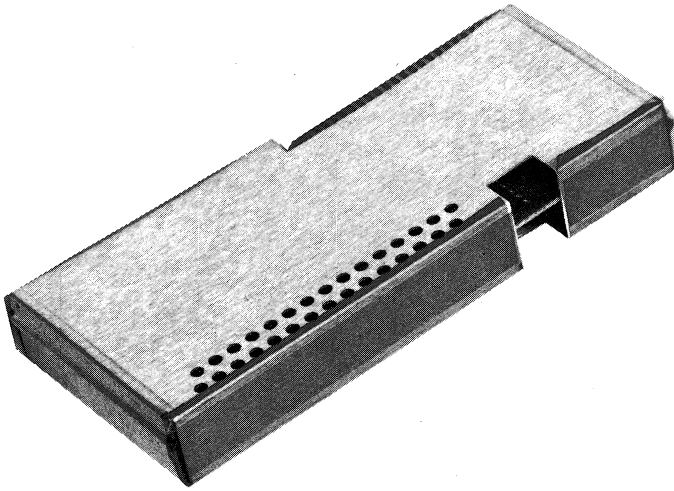
The heart of the computer's processing ability is the *micro-processor chip*, that tiny marvel of electronics that gives your home computer its power. If we looked inside the IBM PCjr, we would find the Intel 8088, 16-bit microprocessor. That's a mouthful to say, but it conveys some important information. To understand what this means, we have to be familiar with "bits" and "bytes".



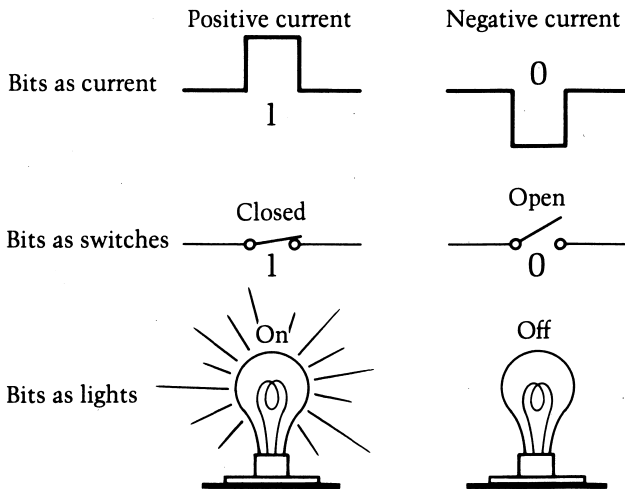
The inside of the IBM PCjr, showing the Intel 8088 microprocessor and the dozens of other chips that make up the systems unit.

The *bit* is an important computer concept, because it is the basic unit the computer uses to process data. There are two different bits (which, by the way, is an acronym for **binary digits**), the two binary numbers 0 and 1. If you are unfamiliar with the binary or base-2 number system, don't worry. Just come along for the ride for a moment.

Inside the computer the bits 0 and 1 are represented as electronic pulses or currents, which are either "off" (0) or "on" (1), or absent (0) or present (1). Figure 2-3 illustrates this idea.



This memory and display option upgrades the 64K model of the PCjr to the 128K enhanced version. Note that the module is encased in metal shielding to reduce interference from other electronic devices.



**Figure 2-3** Bits, or binary digits, can be represented in the computer in a number of ways—as a switch, current, light, or any other two-condition (for example, On/Off) state.

Every piece of data stored in the computer is stored as a group of bits. These bits (groups of 1's and 0's) can mean letters, numbers, or symbols to the computer. Usually, eight bits are required to store one character. In computer lingo eight bits is called (no, not \$1.00) a *byte*. The letter A, for example, usually is written inside the computer as 01000000. The computer interprets the pulses that correspond to these bits and recognizes them as A. Larger combinations of bytes become words, sentences, paragraphs, and so forth. So when the computer "sees" a string of binary digits like

0100100101000010010011010010000001010000010000110110101001110010

it can interpret these pulses as

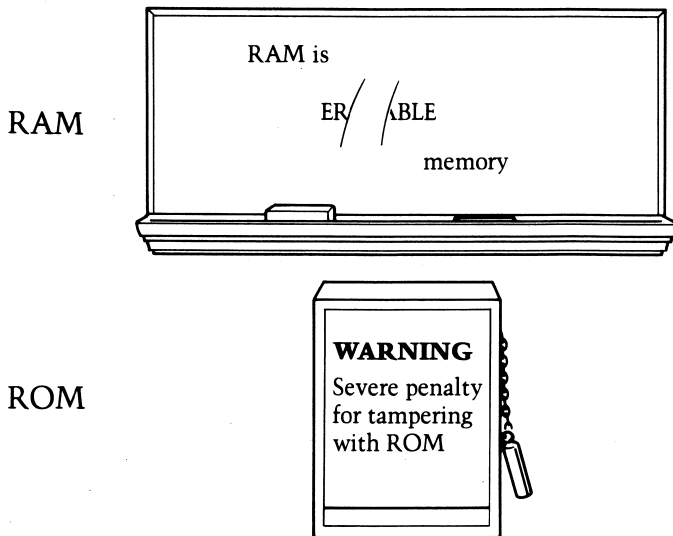
IBM PCjr

Fortunately, all this binary number business is built into the system and you don't have to know a thing about it to use the PCjr.

Until the IBM PC, older brother of the IBM PCjr, came along, most personal computers used eight-bit microprocessors. The IBM PC was the first major system to use the sixteen-bit microprocessor. The extra eight bits make the IBM PCjr microprocessor faster and more powerful, because it can store more data and perform more functions in each operation.

The microprocessor works hand in hand with the computer's *memory*, or storage. The function of memory is to store data and programs and permit their ready retrieval by the microprocessor. There are two kinds of memory: RAM and ROM. RAM stands for **r**andom **a**ccess **m**emory. With this type of memory, information can be stored and retrieved at any time. ROM stands for **r**ead **o**nly **m**emory, or, actually, **r**etrieve **o**nly **m**emory. The original data in ROM is entered at the factory or by special equipment, and the user can retrieve information from it but cannot alter the data in any way (Figure 2-4). ROM extends the storage capacity, but, as you can see, it has its limitations.

ROM, however, does have one important advantage over RAM memory. The information stored in RAM memory is lost as soon as you power down (turn off) the computer, but ROM retains its information when the computer is turned off. This is the reason



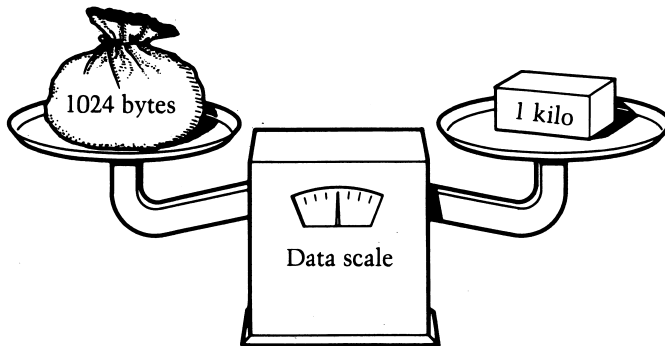
**Figure 2-4** The two types of memory have different characteristics. RAM (random access memory) is somewhat like a chalkboard on which information can be written that is erased at the end of the lesson. ROM (retrieve only memory) is more like a glassed-in bulletin board, containing a special message that cannot be erased or changed.

ROM is used to hold programs and other data that must be available over time.

Regardless of the type of memory in use, all programs must be stored inside the computer in order for you to use them. Therefore the size, speed, and type of memory are key factors in computer performance.

Memory is measured in terms of the number of bytes of information that can be stored. Most memory devices are capable of storing many thousands of bytes, which we call kilobytes (Figure 2-5). The basic IBM PCjr has 64 kilobytes of memory (abbreviated as 64K) as RAM. The expanded version has 128K of RAM. An additional 32K can be added by using plug-in cartridges of RAM—such as the game cartridges.

Also located inside your system unit are other empty slots for additional function cards—for instance, a communications modem, or for extra memory. These extra slots are important because without them you cannot expand your system further without an expensive add-on chassis.



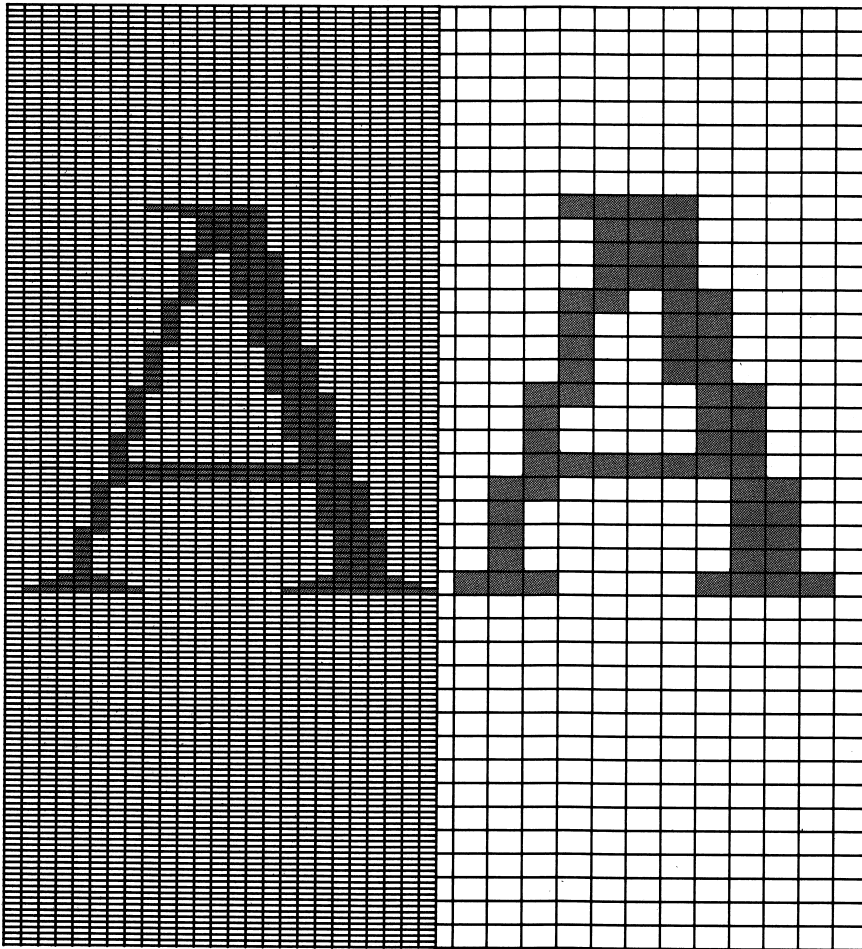
**Figure 2-5** One kilobyte equals 1024 bytes.

That's a quick trip around the inside of the IBM PCjr. The good news is that you really don't have to think much about the innards of the computer while you use it. Understanding the machinery is valuable, though, when the time comes to expand your system.

## Output—The Monitor And Printer

The main output device for the IBM PCjr is the *video monitor*, the televisionlike screen on which the computer displays what's going on. Like the image on your TV set, the image you see on the monitor is formed by combinations of dots. *Resolution* refers to the number of dots (or *pixels* as they are called) that fill a given area. The smaller each dot, the more dots there are and the higher the resolution. You can see the difference between high and low resolution in Figure 2-6. The *refresh rate* refers to the frequency with which the picture on the screen is renewed. To eliminate screen flicker, you need a high refresh rate. For clarity and readability, high resolution is essential.

Monitors display two distinct types of information: text (words and numbers) and graphics (pictures and graphs). Inside the computer different cards are required to handle graphics, and in some cases the same is true of the monitor. A monitor that is suitable only for text display uses *serifs*, or tiny horizontal lines of dots, to form letters and numbers. This results in a clear, attractive, easy-to-read display for word-processing applications (Figure 2-7).



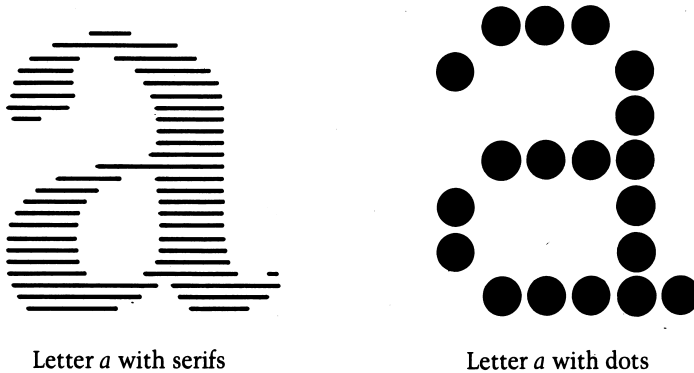
High resolution:  
640 horizontal by 200 vertical,  
or 128,000 pixels

Low resolution:  
160 horizontal by 100 vertical,  
or 16,000 pixels

**Figure 2-6** The resolution of any video display is determined by how many dots, or picture elements (pixels), there are in a given area of screen. The more dots, the higher the resolution.

A graphics monitor uses tiny dots for forming letters and numbers as well as for drawing straight or curving lines, which in turn form pictures. Unfortunately, the dots produce somewhat poorer quality letters and numbers as the price for their extra capability of also being able to draw lines and curves. As monitors continue





**Figure 2-7** Text-only monitors form images of characters using short lines called “serifs” to create a sharp, easy-to-read display. Graphics-capable monitors form character images from groups of dots. The resulting character is not as clear as that made by text-only monitors. However, graphics monitors make up for their relative lack of sharpness by allowing complex patterns and graphs to be displayed. Any pattern, no matter how complicated, ultimately is composed of the same tiny dots that letters are made of.

to improve, the discrepancy between the text legibility of graphics and nongraphics monitors may decrease.

In addition to this difference in the presentation of graphics information, monitors differ in their color patterns. Most monitors are *monochrome* (one color). Three major color schemes are available for monochrome display monitors: green on black, amber on black, and white on black. The amber on black scheme is now generally considered to be the easiest to view for long periods of time, but many people prefer the green-on-black combination.

Monochromatic video monitors are fine for looking at words and numbers. But for really exciting games and graphics, color is the way to go. Of course, color monitors are also more expensive, just as color TVs are more costly than black-and-white ones.

For word processing the color monitor is usually set to white on black. Color is used when graphs or games are displayed. Color is definitely a worthwhile investment, especially if games or graphics are your major interest. Besides IBM there are a number of other vendors who offer fine color monitors at reasonable prices. Two that we have used with success are the Princeton Graphic Systems and the Taxan.

Because display monitors are not exactly disposable, you should spend an hour or so using any monitors you are considering (whether it be color or one of the various monochrome schemes) before you make your final purchase. All differ in color and intensity, and it's important to test the equipment yourself, because the "best" is strictly a matter of personal preference.

The PCjr can also use your home TV set as an output display device, but this definitely is not recommended for anything other than game playing. Your color TV is an extremely low-resolution device compared with a specially designed and engineered computer video monitor. The colors will not be as sharp nor the images as well defined, and letters and numbers will appear ragged. Lines can be no more than forty characters in length, compared with the eighty-character lines of computer monitors. This is a serious drawback if you are using a home TV as a monitor for word processing, because it means you will be limited to seeing only half of what you are writing.

Using a TV set for an output display with the PCjr can be compared to trying to power a yacht with a five-horsepower outboard motor—all right for emergencies but inadequate for any serious sailing.

There is no firm evidence that sitting at a computer terminal for extended periods is harmful to your health. An improperly designed and organized work area, however, can lead to unnecessary strain and fatigue. In the next chapter we will discuss the design of the computer workplace in detail. There is one point, though, that should be made now. The placement of the monitor is extremely important for viewing comfort. The background against which the monitor is viewed should be a dull and nonreflective surface. Do not place the computer monitor against a window, as this will cause your eyes to tire rapidly. Also, be careful of reflections on the screen from lights or lamps. These, too, can be distracting.

Monitors are the output devices we spend the most time interfacing with (looking at). The output of a monitor is temporary, or "softcopy" as it is called in trade talk. When you turn off the computer, the monitor screen goes blank; when you turn it back on, whatever previously was there is gone.

This temporariness is fine for some uses, such as game playing. However, in a wide variety of applications, we need permanent or

“hard” copies of the results. This is where printers come in. In general, there are two types of printers: dot matrix and letter quality. Figure 2-8 shows characters made by dot-matrix and letter-quality printers.

The *dot-matrix printer* forms characters as a pattern of dots. Each character is produced from a nine-by-seven matrix of potential dot positions. With the *letter-quality printer*, the characters are fully formed, like those of a high-quality typewriter. Also, the characters are produced in the same way as with a typewriter — by a wheel, ball, or thimble’s striking against an inked ribbon and paper.

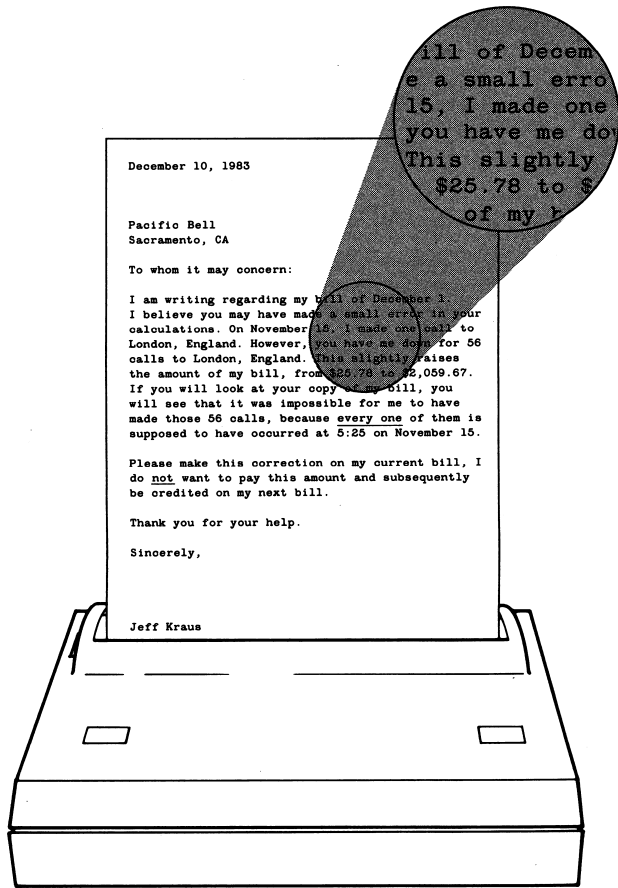
A dot-matrix printer can print graphics as well as text. As with graphics monitors, some of the quality of the letters and numbers is sacrificed to gain the ability to print graphs and pictures. On the other hand, a letter-quality printer that produces fully formed characters comparable to a typewriter cannot produce high-quality graphics. As you can see, each type of printer has its advantages and disadvantages.

Choosing a printer is one of the most difficult hardware decisions you will have to make when you purchase your PCjr. For most people the higher quality printing the letter-quality printer provides is not really necessary, and justifying the greater expense is difficult. In addition, the greater versatility and speed of dot-matrix printers usually lead the average user to start off with one of these.

The factors you must consider in printer selection are: the volume of printing you will do, the nature and quality of the print-out you will need, and the environment in which the printer will be located.

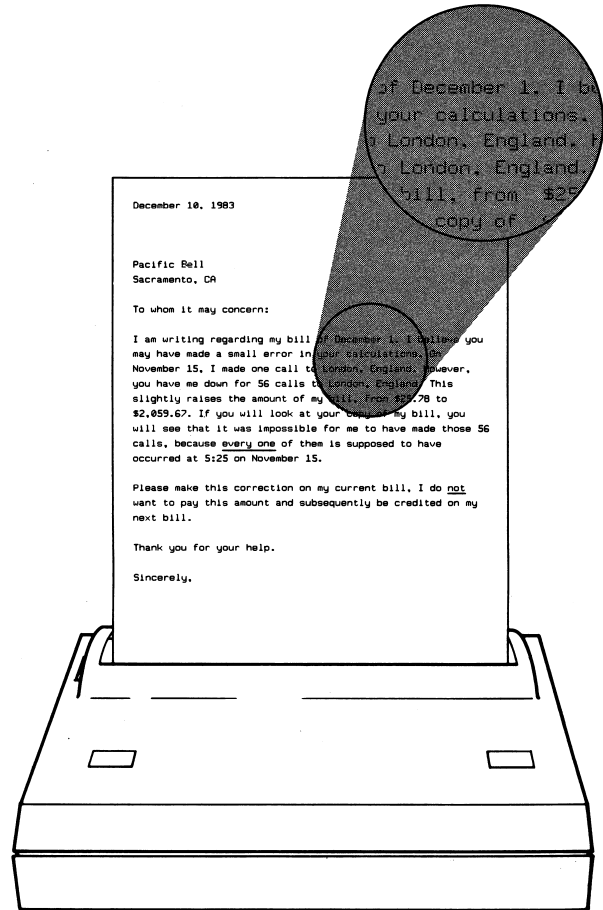
Your volume needs will dictate the requirement for speed in your printer. The manufacturer’s suggested printer companion to the IBM PCjr is the IBM PC Compact Printer. This printer is rated at fifty characters per second and can print eighty characters per line, which means that it will produce a printed page in about a minute and a half. This should be sufficiently fast for most home requirements. The print quality of the Compact Printer, though, is only fair.

The Compact Printer is a thermal printer — that is, it forms characters and images on special heat-sensitive paper. This type of printer is inexpensive to manufacture but has a high cost of operation, because thermal paper is expensive (about five cents per sheet).



Letter-quality output

(a)



Dot-matrix output

(b)

**Figure 2-8** For printing the highest quality text, you need a letter- (or typewriter-) quality printer. The dot-matrix printer, although not producing as sharp a printout, has the advantages of speed, graphics capability, and lower cost. The decision of what type of printer to purchase is one of the most difficult ones facing the prospective computer buyer.

Thermal printers are quiet in operation, which is a strong advantage if the printer is in a multipurpose room. Standard printers, using the impact of an inked ribbon on paper to form characters, are nearly as noisy as an office typewriter. And we all know that typing noise makes a poor companion, especially if you are doing work that requires concentration.

The *PCjr* product line also features a Color Printer, a remarkable piece of printer technology. It can produce color graphics in as many as eight colors, as well as text in a high-speed (200 characters per second) graph mode, and fully formed character-quality text at thirty five characters per second. As you can imagine, the cost of this printer is more than the price of the entire extended *PCjr* system. Most households don't require a printer of this cost and sophistication.

The dot-matrix printer used for the *PCjr's* big brother, the PC, is another printer that should be discussed. This printer, manufactured by Epson, is sold under the IBM label. It also can be purchased directly as an Epson MX-80 printer, usually at a significantly lower price than the same unit offered by IBM.



### ***Printers Available For The IBM PCjr***

<b><i>Printer Model</i></b>	<b><i>Character Formation</i></b>
IBM Graphics	Dot matrix
IBM PC Compact	Dot matrix (requires the use of thermal paper)
Epson FX-80	Dot matrix
Epson FX-100	Dot matrix (allows the use of up to 15-inch wide paper)
Epson MX-80	Dot matrix
Epson MX-100	Dot matrix (allows the use of up to 15-inch wide paper)
NEC Spinwriter 3550	Fully formed, letter quality

Several other members of the Epson family of printers are now available, such as the FX-80, RX-80, and the FX-100. All are excellent devices. They differ mainly in print speed. All of this book's preliminary drafts were produced on an Epson MX-80 or FX-80 dot-matrix printer, and that represented some heavy-duty printing. The price of paper for these printers is usually in the penny-a-sheet range, and the printer ribbons are long lasting.

These Epson printers range in price from about \$300 for the MX-80 to \$600 for the FX-80. All should be satisfactory for even heavy printing demands in the average household. For those applications requiring reports on wider paper, the MX-100 and FX-100 are available. The Epson printer is considered the industry standard for a personal computer.

You may be thinking, "Do I really need a printer at all?" We believe the answer is absolutely yes. There will be many times when you will need hardcopies of your data, such as when you finish a word-processing session and wish to proofread the material at your convenience. Besides, hardcopy data provides extra security against a total loss of hours of work—in the event your diskette or cassette is damaged. The printer purchase is not the place to cut corners to save a few dollars on the cost of your computer system.

### Serial Versus Parallel Interface

When you look at the variety of printers available for the PCjr, you'll discover that besides the dot-matrix and fully formed character printers, there's another wrinkle. Printers also come with two different interfaces—serial or parallel. To allow many manufacturers to produce inexpensive I/O (input/output) devices for computer systems, the industry has standardized these two general ways to connect the devices. For relatively slow devices (in terms of character transfer), a byte or character is moved as a stream of single bits, in serial fashion. Hence the name *serial interface*. For faster devices, the entire eight-bit byte is transferred at once, or in *parallel interface* (which means there are actually eight separate wires making the connection).

One of the standard features on your PCjr is a serial interface for the IBM PC Compact Printer. However, the sales representative may have suggested a different printer for your PCjr and may have



## ***Output Selection Guide***

---

<b><i>Main Use of Computer</i></b>	<b><i>Suggested Monitor</i></b>	<b><i>Suggested Printer</i></b>
Word processing	Monochrome (direct drive or composite)	Dot matrix (Epson), letter quality
Business/financial	Monochrome (composite) or color graphics	Dot matrix (Epson)
Education	Color graphics	Dot matrix (Epson) or compact (thermal)
Games	TV or color graphics	Thermal

told you about the need for a parallel interface, or, in IBM parlance, a Parallel Printer Adapter for that printer.

This arrangement would speed up printing somewhat, as you can see, but the cost factor may make it difficult to justify the increased speed. To make the matter still more complicated, many of the letter-quality printers function perfectly well with the PCjr's standard serial interface. The printer you select depends entirely on your individual needs, so evaluate all possibilities thoroughly before you buy.

## **Storage—Cassette, Cartridge, And Diskette**

In addition to the internal memory we discussed earlier, the computer has external storage devices, wherein data can be stored apart from the computer or can be transferred between programs and other computers.

The IBM PCjr personal computers provide three types of external storage: plug-in cartridges, floppy disks, and cassette tapes (identical to those used for recorded music). The last method actually is not practical for a computer like the PCjr, but for the sake of completeness, we'll discuss it also.



## Printer Comparison Chart

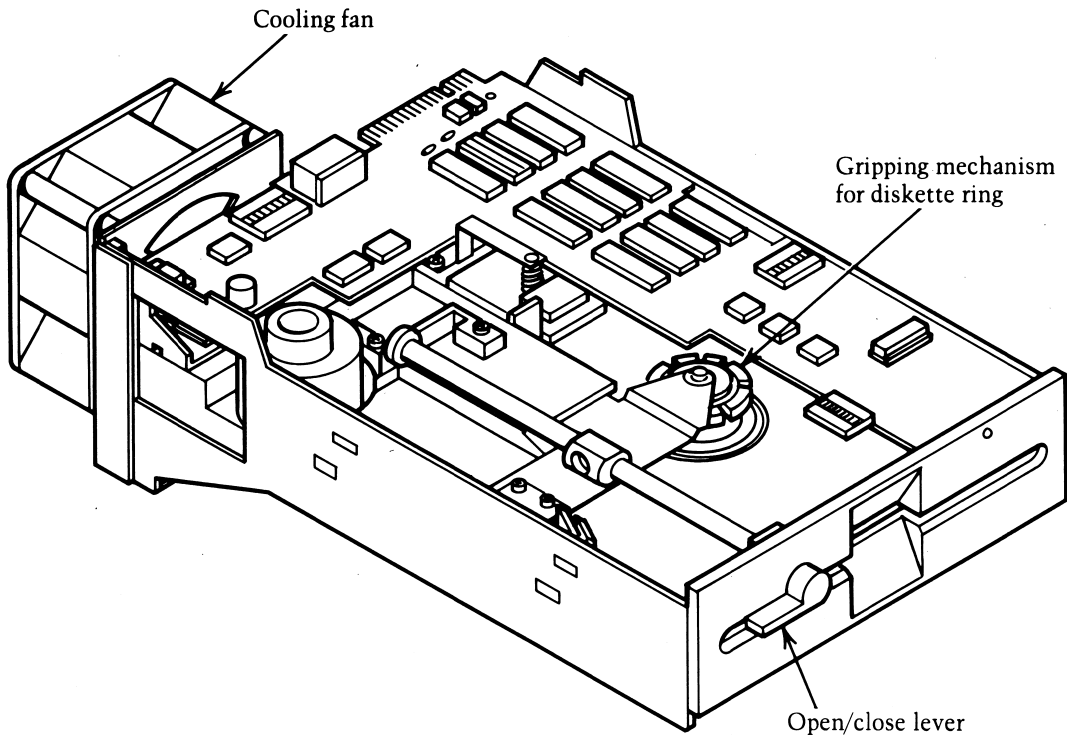
<i>Printer type</i>	<i>Advantages</i>	<i>Disadvantages</i>
Dot matrix (inked ribbon)	Speed, graphics, permanent copy, moderate price	Poor text quality
Letter quality	Excellent text quality	Slow, poor or no graphics, noisy, expensive to purchase
Thermal	Moderate speed, quiet operation, low price	Requires expensive paper, requires that printouts be protected from heat, moderate quality graphics, poor quality text

The first type, *plug-in cartridges*, is found on both IBM PCjr models. These cartridges are made up of thirty two kilobytes of ROM memory and contain commercially prepared programs and games. In actuality, a cartridge is nothing more than a printed circuit board encased in plastic for durability and ease of handling. The PCjr's system unit provides two slots for cartridges.

The second type of storage is *floppy disk* or *diskette* storage, available for the enhanced version of the PCjr. This type of storage requires a device called a *disk drive* to store and retrieve information. Diskettes (the formal name for these little disks) are much like erasable phonograph records, in that they record information on tracks concentrically around the diskette. Figure 2-9 illustrates the interior of a disk drive.

The details of how the data is recorded on diskettes can be fascinating reading, especially if you are an audio or electrical engineer, but the normal or even the advanced programmer, not to mention the personal computer user, need know little more than the fact that the diskette tracks are divided into sectors for data storage. The PCjr diskette stores 512 bytes per sector, uses nine



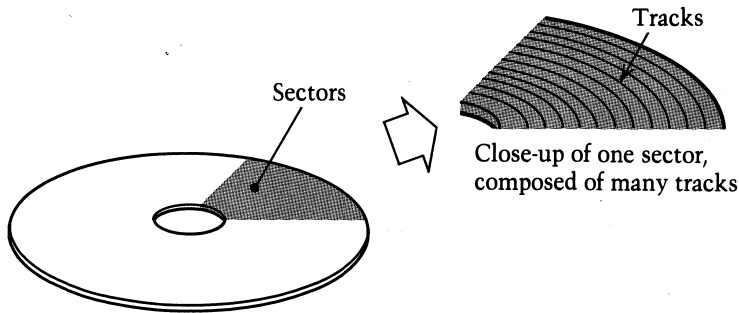


**Figure 2-9** Cutaway of disk drive: Notice the fan at the rear of the drive, which is used to ensure proper operation of the diskette over wide temperature variations.

sectors per track, and has forty tracks per side, with each sector and track numbered so stored material can be accessed rapidly (Figure 2-10). The accessing of data on the diskette is handled by the master control program supplied with the computer. Neither the programmer nor the user of a program has to worry about where data and programs are stored on a diskette.

Diskettes are cheaper and more versatile than cartridges and much more widely used in serious computing systems. In fact, cartridges are not even used in the business setting. The main advantage of cartridges (and the reason for their popularity for computer games) is that they are sturdier than the diskettes, a big plus when youngsters are likely to be handling the equipment.

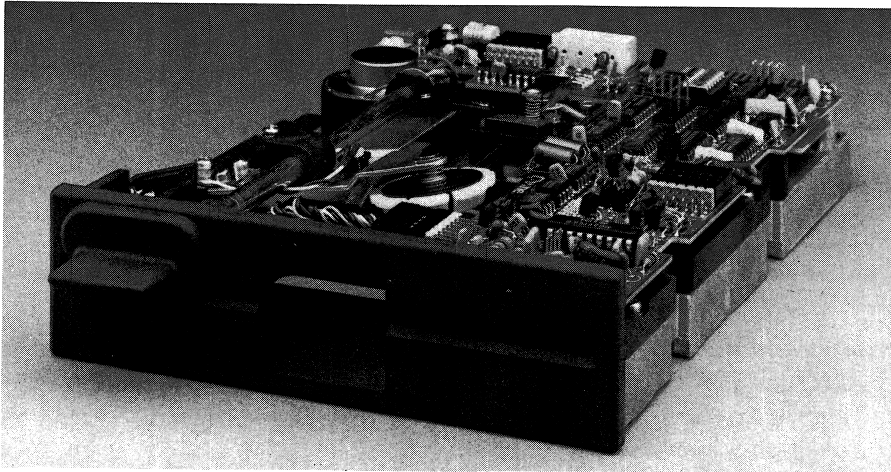
Actually, cartridges provide faster access to information than do diskettes, but given the astronomical speeds of modern computers, this is not much of a benefit. The overwhelming advantage



**Figure 2-10** Once it is formatted, a diskette is partitioned into tracks and sectors. Data is stored on a diskette much like an erasable phonograph record.

of diskettes is that they can be easily reproduced. You can't make a copy of a cartridge, so you can't back up (duplicate for safekeeping) any of the information stored within it. That's a considerable disadvantage in computing, as you will come to realize.

A standard cassette tape recorder can be linked to your PCjr through the use of an adapter cable. Programs can be read into the



The Qume 142 is a high-capacity, half-height, 5.25-inch disk drive. It can store unformatted 500 KiloBytes and is intended for use in a broad range of word processing and small computer applications.

computer through the cassette recorder, and data can be stored on the *cassette tape* in the form of audio tones. Although this type of storage is the cheapest to install initially (especially if you already own a suitable tape recorder), it is the slowest and least practical storage form available. Also, in the long run cassettes are expensive, unreliable, easily damaged, and they tend to lose their quality fairly quickly with repeated use. Using cassette tapes is not recommended with a machine of the power and sophistication of the PCjr.

## Other Parts

Cables connect the various devices to each other. Detailed instructions as to "what goes where" will come later. For now, suffice it to say that the cables are an integral part of the system and need to be handled gently and safely, the same way you would handle any electrical cable. The connectors on the ends of a cable are easily damaged. The key to success here is simply using a little patience and reading the instructions. Don't force a fit with any parts of your computer, especially cables. Most electronic equipment cables are purposely designed to fit only one way, to prevent current and voltage from being impressed on the wrong parts of a circuit. If the cable you have doesn't seem to fit, it's probably the wrong one for the job. We learned this the hard way.

We purchased a diskette drive and the associated cable required to connect it to our computer system. Although the cable did not slip into place easily, with a little extra force we managed to connect it. As soon as we turned the power on, though, we heard a squeaky sound and smelled burning circuitry. When we returned the cable to the vendor, he acknowledged that it was the wrong type for our system, but in the meantime the damage was done. The drive had to be shipped to the manufacturer, and we were out of business for weeks. Had we just returned the misfit cable, we would have received the proper replacement and avoided damaging the system.

Damaged cables are responsible for more computer service calls than any other reason, so be sure to think twice before trying to "help it along" when it doesn't fit easily.

### The Keyboard Connection

The keyboard is the only device that is not physically connected to some other part of the system. This is because of its unique, infrared cordless link, powered by four AA batteries. The rumor circulating in the industry says that the keyboard was deliberately separated from the system because people are afraid of keeping electrical appliances on their laps. If you want to purchase a permanent connection, however, you can hook up the keyboard to the rest of the computer anytime you like by using a cable similar to a modular telephone cord.

There are several things you need to know about the special infrared linkup between your keyboard and the system unit. A successful connection requires a direct, line-of-sight pathway for the infrared beam, free of any interfering people or objects. Imagine that there is an eye inside the back of the keyboard and that it is staring straight ahead. If this eye cannot "see" the computer, no data will be transmitted. The keyboard will allow some degree of tilt (a total of about sixty degrees), but basically it must be facing directly toward the main computer, with no solid objects in the way.

Using the cable connection between the keyboard and the computer is necessary if you have more than one PCjr (or even remote TV controls) in the same room. Otherwise, the infrared signals from each keyboard will interfere with each other and render everything you type meaningless.

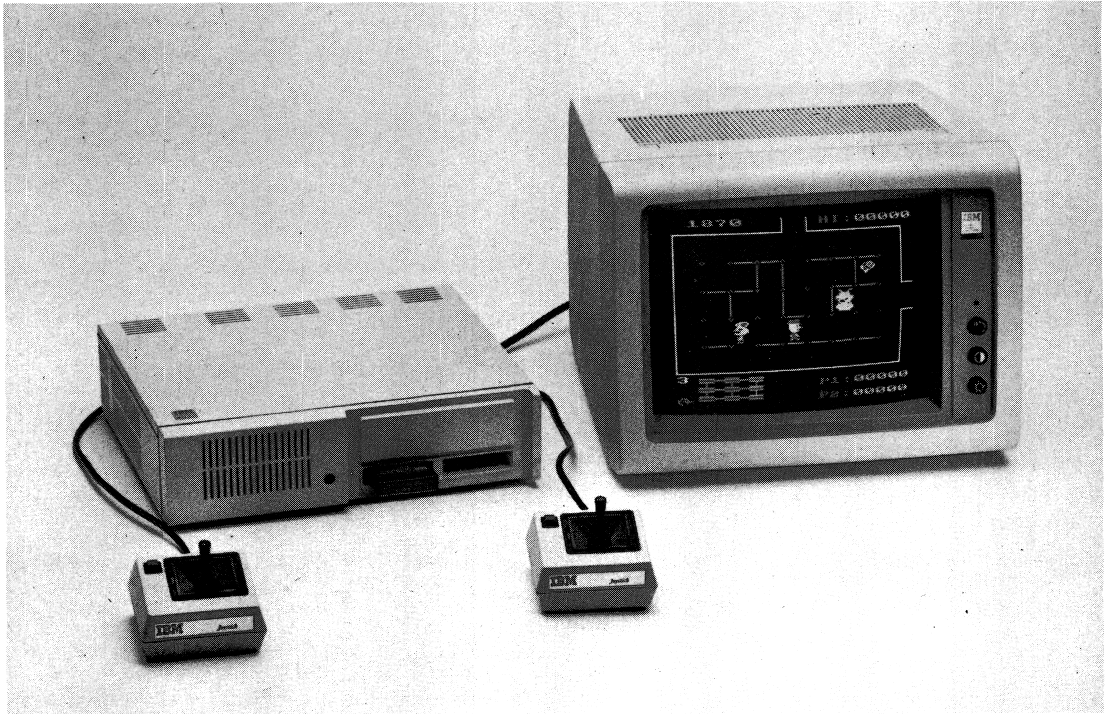
Another cable option is the television connector that allows you to connect your PCjr directly to your home big-screen television set. Although this may not be advantageous when you use the computer for word-processing applications or data processing, it can make a big difference when you play computer games.

A word of caution here: Strong and persistent computer video signals can damage the phosphor coating of your TV screen, especially if the same image is left on in the same location for a long time. This can occur quite easily if you start a game, then leave and forget to turn off the computer. Be sure that standard operating procedure at your house is to turn off the TV or monitor—or at least turn down the brightness and contrast—if no one will be using the system for an extended period of time.

## Joystick

Also available for game-playing applications is the attachable *joystick*, a movable control handle that allows faster, more direct control for computer game playing. The extra control and sensitivity of the joystick should more than justify its added cost, particularly if you have known the joy (and frustration) of playing computer games in the video arcades. The PCjr joystick is exceptionally good, at least as reported by some of our game-playing friends.

There are two modes of operation for the joystick, the "spring return" and the "free-floating" modes, which can be selected by a switch setting. In the spring-return mode, the joystick will return to the center position automatically when released. The free-float-



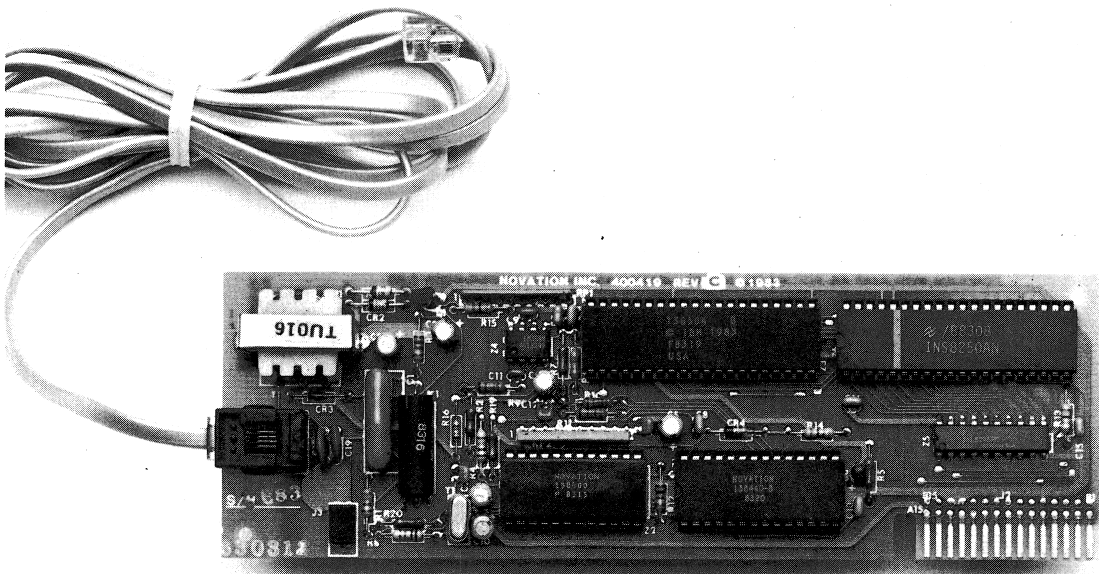
IBM joysticks for the PCjr.

ing mode permits you to maintain the joystick in any position without additional effort. Different game situations and personal preferences will dictate which mode you will want to use.

### Modem

The internal *modem* available for the PCjr is, literally, a window on the world. The modem lets you send and receive data, messages, and programs to and from anywhere that can be reached by a telephone. The technique, called *telecommunications*, is so exciting and important we have a separate chapter devoted entirely to that topic.

A modem is a device that converts the digital pulse of the computer into audio tones that can be transmitted over a telephone line. The modem also can receive audio tones and change them back to the digital pulses the computer recognizes.



A modem is used to connect one computer to another via telephone. A variety of information can be conveyed to your computer over the telephone, like the latest stock market prices.

What the modem really does is make the telephone channel just a long cord between two or more computers. As a result, computers can carry on conversations over distances of thousands of miles, play games with opponents continents away, and search data files in libraries anywhere in the world. A number of services are now available that let you check the weather around the globe, make travel reservations from your own computer, or even make purchases. These services use local telephone lines to keep communications costs low.

We recommend purchasing the PCjr internal modem, and as we point out in Chapter 10, you will soon find many opportunities to use it for business and pleasure. Using the modem is simple. In the back of the PCjr system unit is a modular telephone jack. Your telephone cord plugs directly into this, connecting your computer to the world.

By the way, this "internal" modem (so called because the connection is made between the telephone line and the computer directly) is a big advance over the old, "external" modems. The old, external modems sat next to the computer and required you to insert your telephone receiver into receptacles in the modem. The modem then listened to the telephone signal and translated it to the computer via a connecting cable. The new internal modems eliminate any need to use the telephone itself. It even eliminates the need for dialing with a rotary dial telephone.

### Light Pen

PCjr offers a *light pen* interface as part of its standard features. This device looks like an ordinary ball-point pen, but it is light years away from such a simple writing device. Instead of a point, it has a light-sensing tip that can be pressed against the monitor to enter data directly into the computer memory. Because the light pen eliminates the need to keyboard responses, it is an easy method of communicating with the computer. It allows slow and inexperienced computer users to respond to questions or to select various options.

Programs supporting light pen input are still relatively rare. Be aware, though, that this input device exists. It can and certainly will be used in conjunction with computer graphics displays as an excellent teaching tool for children.



The light pen is a sophisticated input device that conveys information to the computer by the location of an illuminated area on the monitor screen.

## Further Reading

*The Basics of Digital Computers* (volumes 1 through 3), by John Murphy (John F. Rider Publisher, 1958). This early book on computers was based on a course in computer basics for computer repairmen. Because all the ideas and concepts were new at that time, the author used simple analogies, which makes the book excellent for beginners. Almost all of the technical information is hopelessly out of date, so don't purchase the book; look at it in the library instead. Although the details may have changed, the fundamentals have remained constant. In many areas of computing, the old adage "The more things change, the more they remain the same" is certainly true.

*Guide to Personal Computing*, by Digital Equipment Corporation (129 Parker Street, Maynard, MA 01754, 1982). This little booklet formerly was a giveaway by the number-two computer manufacturer in the United States, and it is now being sold in computer stores. Although it is heavily oriented toward DEC equipment, it has some good computer graphics as well as a



sound discussion of the various components of a computer system.

*An Introduction to Microcomputers* (volumes 0 and 1) by Adam Osborne (Osborne/McGraw-Hill, 1980). A technical overview of how microcomputers perform their various functions. Because the discussion goes into the bits-and-bytes level of detail, the book is not what might be considered light reading.

### Magazines And Periodicals

The array of magazines and periodicals available on personal computers is bewildering, but such publications can be classified roughly into two categories: general purpose and equipment specific.

The general-purpose magazines contain articles about the whole spectrum of personal computers, whereas the equipment-specific ones are directed at a particular computer or product line. Computer magazines and periodicals, like some computer manufacturers, have a way of disappearing without a trace. Of the magazines and periodicals that have been published over the years, four distinguish themselves and are recommended for further reading. There is no need to subscribe to all four. One from each category—general purpose and equipment specific—is a good mix. The four are:

*Byte* (70 Main Street, Peterborough, NH 03458, published monthly).

An oldie but goodie in the microcomputer magazine area. The focus of this magazine often is quite technical, with issues devoted to in-depth studies of particular topics or pieces of equipment.

*Creative Computing* (P.O. Box 5214, Boulder, CO 80321, published monthly). This magazine, one of the granddaddies in a young business, covers the gamut of the personal computer industry. Its editorial thrust is toward people, personalities, and personal applications.

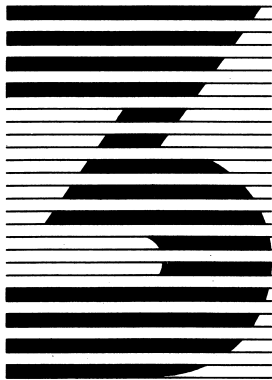
*PC Magazine* (One Park Avenue, New York, NY 10016, published twice a month). This one concentrates on the IBM PC line of computers. It treats business and home usage and often has features on add-ons for the PCs.

*PC World*. (555 De Haro Street, San Francisco, CA 94017, published monthly). A newcomer that looks good. Aimed at both PCs and

PC look-alikes, this applications-oriented magazine is of high quality, both technically and from a publications point of view.

### **PCjr Magazines**

A batch of new magazines aimed directly at the PCjr has sprouted. Some caution is in order here. Often, starting with a general magazine is better, waiting to see which of the new kids on the block turns out to be the best.



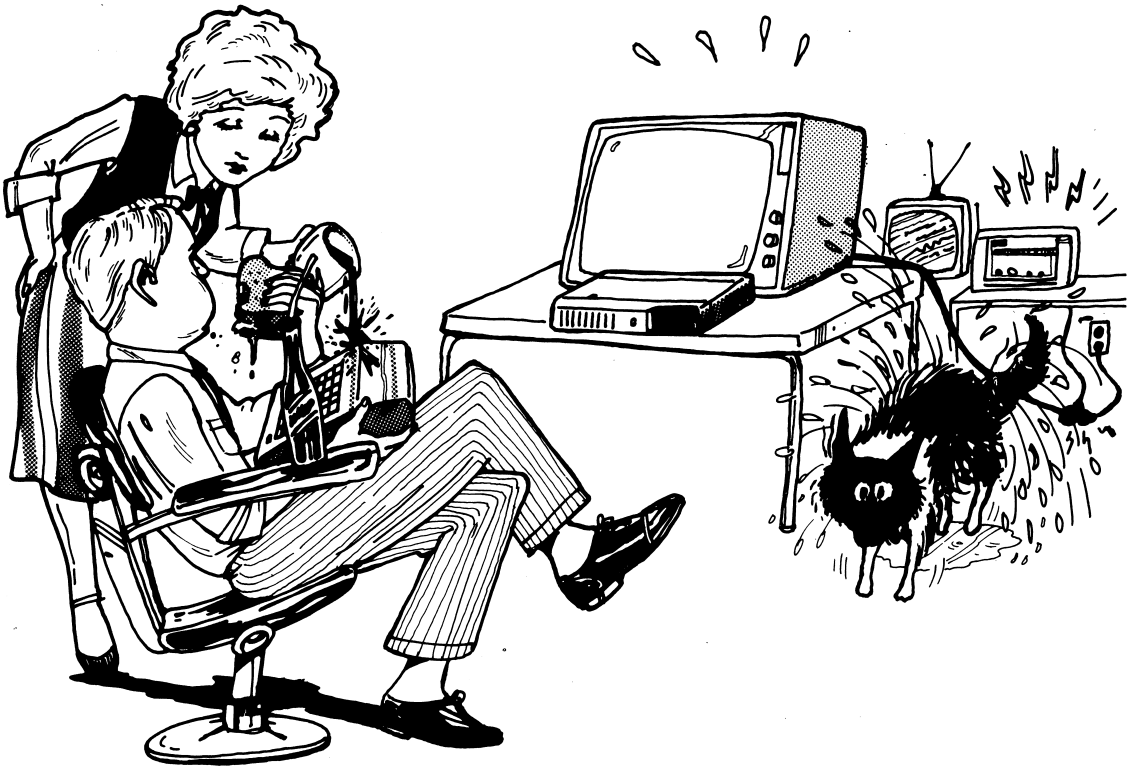
## **Setting Up Your PCjr: Do's And Don'ts**

The grandfather of your IBM PCjr, the large IBM computer of the 1960s and '70s, was kept in a secure, windowless, air-conditioned room—and for good reason. Computers are extremely expensive and powerful pieces of equipment, and they are also fragile electronic devices. Dust, dirt, excessive sunlight, smoke, extreme temperatures and humidity, and rough handling can impair performance or cause errors in delicate storage devices like diskettes or the larger hard disks (similar to floppy disks, but even more susceptible to dust and dirt). The PCjr, although a sturdy machine, is still a precision instrument and needs protection from the same conditions as did its larger ancestor.

In addition, household hazards that the big IBM machines never dreamed of — pets, spilled coffee, curious children, and competing electrical devices like TVs, microwave ovens, refrigerators, and vacuum cleaners (which can cause power surges when turned on) — threaten your IBM PCjr on a daily basis (Figure 3-1).

### **The Computer Workspace**

You probably don't have a special windowless room in your home just waiting for your computer, so you'll need to choose carefully when you decide on a spot for your machine. The work area should be secure, quiet, well ventilated, comfortable (not too cramped for space), and should have plenty of light, several grounded electrical outlets, and some storage areas for books and papers.



**Figure 3-1** Some of the household hazards that can damage the PCjr.

Sunlight may add a lot to an environment for the computer user, but you should avoid the temptation of putting your computer in an area exposed to direct sunlight. Sunlight can generate a great deal of heat, which in turn can damage your diskettes and the delicate electronic equipment you have just purchased. Several hours of direct sunlight can generate extremely high temperatures inside the computer, and you would be surprised how quickly the equipment can be damaged. And even if you avoid permanent damage, you can expect some intermittent problems. To make matters worse, it will take you a long time to trace those read errors or system malfunctions back to the "sunburn" your computer suffered.

This advice is based on hard-earned experience. We went through a period of recurring problems that none of the computer repair shops could diagnose. After weeks of frustration, we finally moved

the computer a few feet away from its original position (which was catching the afternoon sun), and the system miraculously cured itself.

A nearby telephone or telephone outlet is desirable for several reasons. First, if you are the only one home and the phone rings while you are hard at work with your PCjr, the mad dash to the telephone can be quite distracting—and often ends up in the frustration of hearing a dial tone when the calling party has given you up for dead. A nearby telephone (or outlet into which you can plug your phone) will eliminate this unnecessary distraction.

If you do have a phone nearby, *resist* the temptation to keep right on computing while you chat with your friends. From our years of experience we have learned that communicating with computers is a full-time job, and so is communicating with people. Your friends will be quick to recognize the random series of “uh-huhs” that emanate from your end of the line as a sign that you aren’t paying attention, and your phone calls—as well as friends—will get fewer and fewer.

The second reason for having a phone outlet in your workspace is to facilitate telephone communication between your computer and other computers. A variety of devices, such as the modems that connect your computer to others via the phone, are readily available. The services accessible to your computer by telephone include stock market reports, airline reservations, and shopping, to name a few, and the list keeps growing.

### **The Proper Power**

If your home was built within the last thirty years, you already have three-wire, grounded outlets. Most building codes and regulations have called for this type of electrical wiring for at least that long. If you do not have this type of outlet, the best advice we can give for your own safety is that you call a licensed electrician about rewiring your home. Not only will your electrical appliances be less likely to malfunction with grounded outlets, but your home will be much safer.

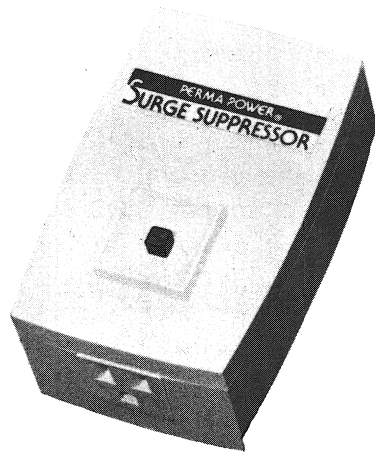
With most electronic equipment you already own, turning the machine on and off is as simple as flipping a switch. IBM PCjr also has an on/off switch, but turning your personal computer on or off

can be more complicated than just that. This is because turning on electrical appliances can lead to power surges throughout the circuit. Toasters or other appliances usually are not bothered by such current surges, but your computer can be very susceptible, and both hardware and software can be damaged. If a large appliance such as a refrigerator, dishwasher, washer, or dryer is on the same circuit as your computer, the chance of a problem is magnified, as the current surges are greatest with these larger appliances.

Electronics experts advise two methods to minimize the chance of damage caused by power surges. You should turn on appliances so that the biggest power consumer is the first one to go on (and so forth down the line). Usually this precaution is all that is necessary to prevent problems. However, if your system consists of a printer, monitor, and systems unit, they should be activated in that order and deactivated in the reverse order.

The other, more satisfactory, alternative is to turn on all the components of your system at exactly the same time. This is easier than it sounds if you use a *power bar* that allows all the devices to be controlled by a single switch. To turn the system on or off, you simply flip the switch on your power bar.

A power bar alone, however, cannot guarantee your system against power surges, particularly ones originating outside the home,

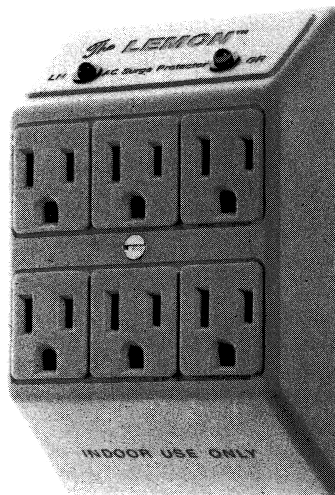


A power bar will minimize the chance of power-surge damage to any of your system's components.

such as with a power failure or electrical storm. And it doesn't take much of a surge to destroy all the data in your computer, as well as the backup file on the disk. In addition, you can burn out the delicate circuitry in a flash, as we learned the hard way when lightning struck a power line in our neighborhood causing a massive burnout of several thousand dollars' worth of computing equipment two days after the computer's maintenance agreement expired.

Another calamity that occurred during the preparation of this book did no damage to the equipment but did cost us several days of hard work. A light airplane crashed into a nearby power line, and the resulting power surge and brief outage played havoc with the computer memory and all the backup data stored on a diskette that was in the computer at the time.

A *surge protector* could have prevented or at least minimized both losses. These two episodes convinced us that lightning and small planes can strike twice in the same place. You can bet that the second lesson convinced us to invest in a surge protector for *all* our computer power lines. This relatively inexpensive electronic device can pay for itself in one incident, saving you a considerable amount of grief.



A surge protector will prevent damage caused by sudden changes in electrical currents.

Also beware of carpeting in your workroom. Carpets can be a source of both static electricity and dirt. A single static electricity discharge can ruin the circuitry of a computer and destroy the data on a disk or tape. Most professional computer work areas are not carpeted or else use special antistatic carpet or pad in front of the computer (which can be quite expensive). If you do have a carpet in your computer area, take care to keep it clean, and if static electricity is a problem, get some antistatic carpet spray from your local computer store or office supply house. The spray is not too expensive but must be used frequently.

If static is particularly troublesome, as it is in some of the drier areas of the country, or in heated homes in the winter, a humidifier can help. (You can tell that static is building up if the hair on your arms starts to rise or if you have a tingling feeling in your fingertips.) Static discharges were playing havoc with both the micro-computer and the data on magnetic media at a bank in New England. A simple home humidifier was installed in the computer room to maintain the relative humidity at slightly over 50 percent, and the troubles were over.

### **Furniture And Layout Of Equipment**

The layout of furniture in your computer workplace is extremely important for comfortable computing. Many people don't realize how many hours they will spend in front of their computer when they first buy it. But after you've been sitting in one place for four or five hours, you will appreciate the time you took to plan your work area, or you will regret *not* taking the time to do so.

The first piece of furniture you need is a table to hold your IBM PCjr. This table should be sturdy and secure, and no more than twenty-seven inches high. The typical dining room or kitchen table is thirty inches high—too tall for comfortable typing over any length of time—whereas the living room coffee table is too low. Placement of the keyboard at a level between twenty-five and twenty-seven inches is optimal for most people.

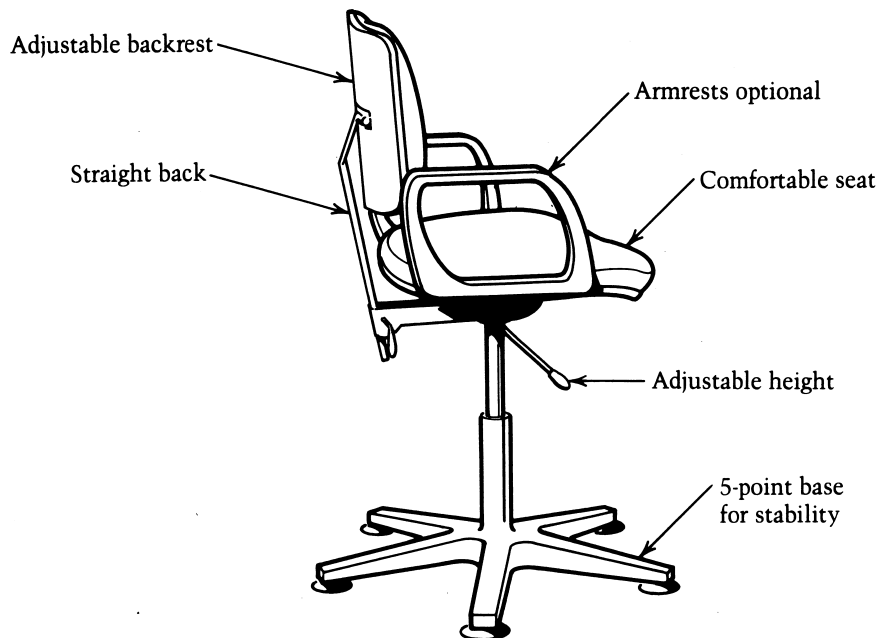
Pay special attention to children's needs. When they are using the computer, they may need either a lower desk top or an adjustable chair that can be raised to the proper height. Children have a tendency to sit too close to the screen, so be sure they have plenty of room and a comfortable arrangement.



The computer table should be at least forty inches wide to provide workspace on either side of the machine for books, papers, or other equipment. Avoid typewriter-size tables, even if they say "computer work table," as they are too tiny for comfortable working. Don't stack flimsy shelves immediately above your computer, which can lead to books or other objects' falling onto the computer.

Another key point about the table: It must also support the monitor at a comfortable height. If you find yourself bending over or getting a crick in the neck, adjust the monitor to a more comfortable viewing height.

After the table, not surprisingly, comes the chair (Figure 3-2). Again, several hours of hard work at your computer keyboard will be all you need to make you appreciate the value of a comfortable chair. A fully adjustable, secretary's chair is ideal — and worth the expense. The chair should be adjustable for height, backrest position, and backrest angle. Adjustability is also a must if different-size individuals will be using the computer. These chairs normally



**Figure 3-2** A good-quality, comfortable chair is one of the best investments for successful computing.

come with rollers and a swivel base so that they can be easily positioned at a comfortable distance from the monitor. Because the usual tendency is to sit too close to the monitor screen, be sure there is room in front of your computer to move the chair back. Also, be sure your chair has a five-pronged base, rather than the old-style four-pronged one, because the five prongs make the chair more stable when you lean backward.

Arm rests are a matter of choice. We have used chairs with and without them. Armrests are valuable if you spend a good deal of time contemplating the results displayed on the screen. For sustained keyboarding, though, an armless chair is better—and less expensive.

It's worthwhile keeping in mind that although the initial investment you have made in your IBM PCjr may be considerable, if you try to get away cheaply from there on, you may find yourself getting less and less use out of your machine. An uncomfortable computer operator usually becomes an *ex-computer operator* before long (Figure 3-3). If the initial expense of a new chair is too steep, consider purchasing a good-quality used one. As we said before, a secretary's chair is ideal. Anything with the label "computer" seems to be priced considerably higher than the identical item designated "secretary," so don't be talked into spending *more* than necessary either.

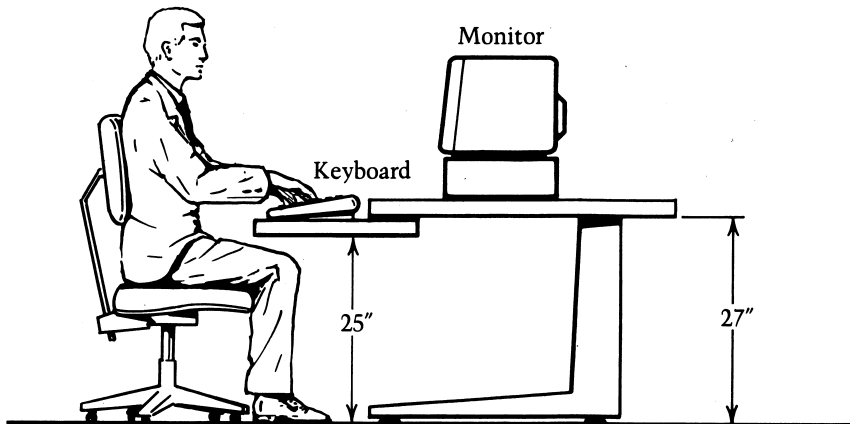
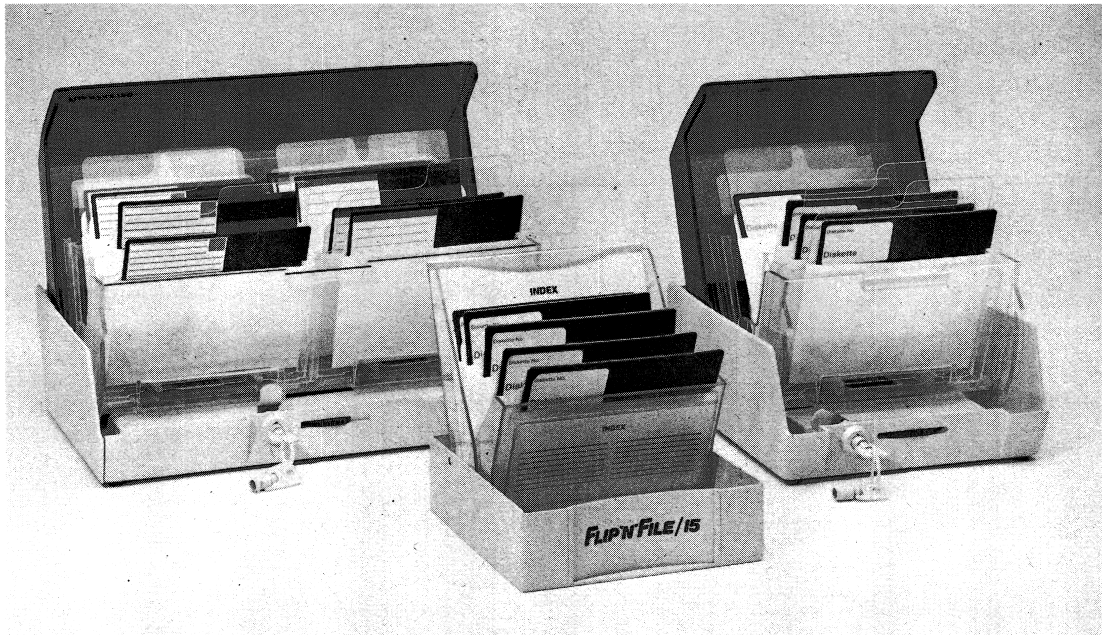


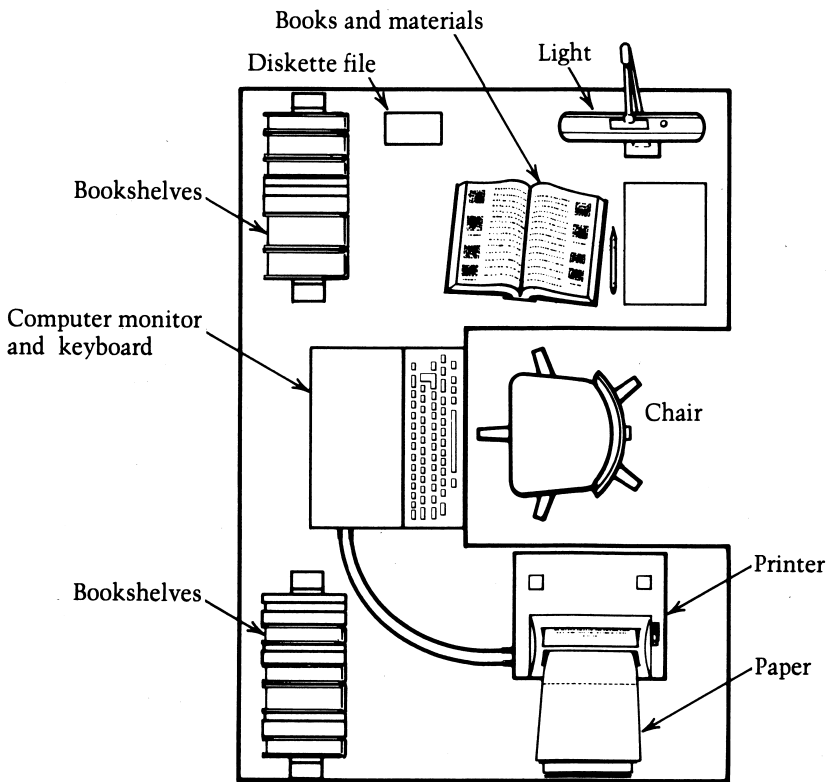
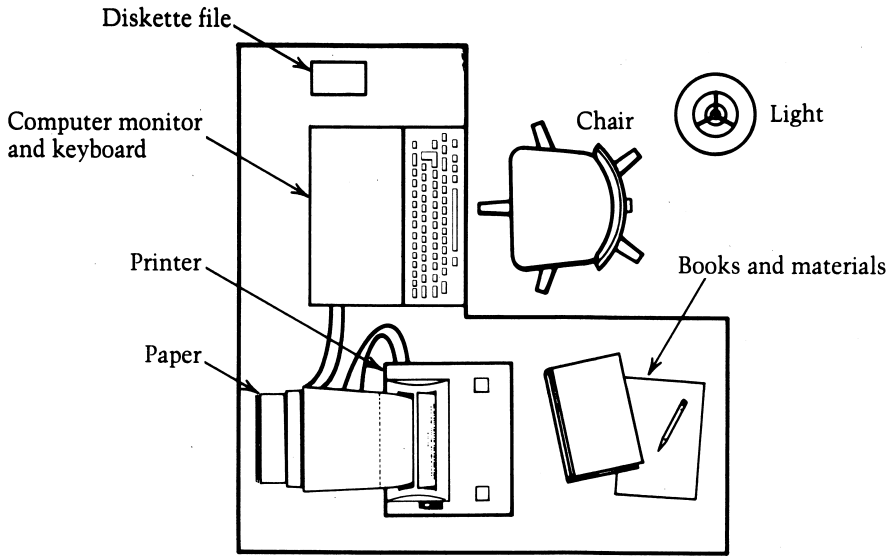
Figure 3-3 Blueprint for comfortable computing.

On your table, or within easy reach, should be a typing stand, a diskette file, a bookshelf for your most frequently used manuals and books, and an open area for writing. A U-shaped or L-shaped layout (Figure 3-4) composed of the computer desk and side table or tables, with back lighting (you don't want a bright light shining in your eyes while you look into the monitor screen), is the most energy-efficient layout for your computer workplace. A locking file cabinet or similar storage area is an excellent addition as well, allowing secure storage of your disks and printouts when you are not working at the computer.

When you design your computer workplace, it will be important for you to decide the main purpose of your computer—games or business. The two often don't mix, for a number of reasons. Games frequently involve more than one player, usually seated side by side. This arrangement requires a larger area for seating and, of course, another chair. But more importantly, the players of the game



Valuable accessories for the work station. Notice the absence of metal parts on these accessories.



**Figure 3-4** Recommended layout of the computer work area is L- or U-shaped. To minimize glare, the light source should be behind and to the side of the user.

may not share your concern for maintaining the environment around the computer. If you plan to share your IBM PCjr with a host of game players, make sure you have storage space to put away all your programs, printouts, and other valuable, fragile, and confidential materials.

## Insurance

Your IBM PCjr was a big investment, about which you are no doubt justifiably proud. As with other major investments, having insurance on your computer is an excellent idea. If you have homeowner's or renter's insurance, ask your insurance agent if the standard policy covers personal computers or if such coverage requires a separate declaration. Remember, though, that whereas standard home insurance may protect against fire, theft, or property damage, damage to valuable software by power surges or spilled coffee generally is not covered.

Policies vary in the amount paid for a loss. Most homeowner's policies base reimbursement on a depreciated value for the computer. Normally, this means you lose at least 20 percent of the value of your system the first day you take it out of the box. For computing equipment, replacement cost coverage is a better approach.

Insurance specifically designed for computer owners is now available from several firms, with premiums that are reasonable for low deductible rates. Most of these policies include damage to programs as well as damage to components of the computer. A good rule of thumb is that you should cover about 80 percent of the current replacement cost of your total system, including software. If your insurance is limited to the hardware components, it's a hollow reed at best. Get a software rider to protect this valuable part of your computing investment.

Coverage and pricing vary significantly among carriers, so shop around and be sure you know what you're buying. Although prices vary, for approximately \$80 per year you should be able to get about \$5,000 in replacement value coverage, with the first \$100 of loss deductible. It's sound business practice to keep all receipts for your computer hardware and software, in the event you have to make a claim.

If you use your home computer for any business-related purpose, it may not be covered by your normal homeowner policy. We had to take out a special rider to cover the computing equipment for our writing projects.

## What's Left?

Now that you have your new IBM PCjr safe, secure, comfortable, and insured, you are ready to get started. But you will need a few additional items as a "starter kit."

### Diskettes

If your PCjr is the enhanced version with the diskette capability, you'll need a supply of floppy disks. A box of ten diskettes costs between \$20 and \$50 for a quality brand—such as Maxell, IBM, 3M, Dysan, Fuji, Sentinel, or BASF—and one or two boxes will get you started in good shape. We have used all of these brands and have found them to be reliable. Technically speaking, the PCjr drive requires double-sided, double-density, soft-sectored, 5 $\frac{1}{4}$ -inch diskettes, such as the Maxell MD2 or the IBM 2D. Most of the PC users we know have been using the less expensive single-sided, double-density versions, with excellent results. However, the PCjr drive may be more temperamental than its full-sized cousin on the IBM PC. Therefore we recommend that you stick to the double-sided, double-density diskettes, until user evidence shows that the single-sided ones work as well.

For a small cost you can buy diskettes in a plastic case. This is a good idea for the first few boxes, but as you start to accumulate your own library, a file case that has a 25- or 50-diskette capacity (and can be locked) is the only practical way to store these fragile items.

### Printer Paper

If you are using the IBM Compact Printer, you will need thermal-sensitive paper. If you are using an Epson or similar dot-matrix printer, you will need 9 $\frac{1}{2}$ -by-11-inch fanfold continuous sheet paper. But even the paper for these printers varies. Paper comes in a variety

of grades and weights. Just as for typing, the heavier the weight of the paper, the stiffer and more opaque it will be. We recommend eighteen- to twenty-pound paper. If you have a letter-quality printer, which takes only single sheets, any good-quality typing paper will do.

### **Ribbons**

A variety of ribbons are available for computer printers. Inked ribbons are the standard type for multiple use. Carbon ribbons afford greater clarity and bolder print, but they cannot be reused. Also, they are more expensive and generally not worth the added expense, unless you are preparing important documents. We recommend inked ribbons, but keep only one or two extras on hand, because they have a relatively short shelf life before they begin to dry out.

### **Dust Covers**

Dust and smoke are not just annoying nuisances in the computer environment, they can be deadly enemies to your equipment. Dust and smoke particles can penetrate the delicate mechanisms in the disk drives and keyboard and can gum up the works. The worst part about them is you won't even suspect that a problem is brewing until you have a system failure. And, of course, failures only happen at the worst possible times.

The solution is easy. Invest in a set of dust covers for the keyboard and the systems unit—and then remember to use them whenever your machine is not in use. They are relatively inexpensive, but they can more than pay for themselves in the long run. We have used the CompuCover keyboard protector and found it to be extremely valuable. Not only is it dustproof, but it protects against water spills as well. And, as you know, water is another hazard for electronic equipment.

Smoke is also a culprit that can wreak havoc with your system, so keep the computer room off limits to smokers.

### **Cleaning Supplies**

One of the preventive maintenance duties you should assume (if you want to avoid costly repairs and downtime on your computer)

is regular cleaning of the equipment. A *gentle* vacuuming of the keyboard and printer about once a month should do the trick for those two items.

The disk drive heads, however, should be cleaned every ten to fifty hours of use. The easiest way to do this is to use a disk drive head-cleaning kit. Such kits are simple to use and only take a few seconds of your time. We use the kit sold by 3M Corporation, but there are several cleaning kits on the market.

Resist the urge to scrub your computer. Water or solvent can ruin a machine in short order, so keep any liquids away from the system. That's why we caution you against leaving drinks around the computer area. We use nothing more than a damp cloth to wipe off our monitor screens and a dry cloth to dust off the systems unit.

### **Fire Protection**

Computers, like any piece of electrical equipment, can cause fires if abused. The most important preventive measure is to have sufficient electrical outlets so you don't overburden one particular line. Be careful about bare wires or cords that can be tripped on and yanked from their sockets.

If you use common sense, your computer should pose no fire danger. In businesses, though, having fire extinguishers on hand is standard practice in computer rooms. If you want to purchase one for your home, get a carbon dioxide (CO<sub>2</sub>) or halon extinguisher. The last thing you want to do is use water on an electrical fire.

### **Carrying Case**

The PCjr is light and compact enough to make a good traveling companion, if you want to take the machine to work with you. But you just can't toss it in the trunk of the car and expect it to survive. A carrying case—offered by IBM as an option for PCjr—will serve double duty. The case will make it easy for you to move the system, and it will protect the equipment in transit. A carrying case is not a shipping container, however, so don't expect it to withstand rough handling.



### Odds And Ends

It's a good idea to keep on hand a couple of felt-tipped pens or markers for labeling your floppy disks. Pencils or ball-point pens definitely are out on diskettes, as they will damage the magnetic surface within.

A notebook to keep a log of what you're doing and to jot down questions or reminders to yourself is always welcome in a computer room. And don't forget to equip the room with a king-size wastepaper basket for all those printouts that weren't quite right. If there's one thing a computer does well, that's generating waste paper.



### *Computer Workspace Checklist*

---

1. Well-ventilated, well-lit, quiet room (with computer out of direct sunlight)
2. Adequate electrical outlets
3. Telephone or telephone jack
4. Table—twenty-five to twenty-seven inches high
5. Adjustable straight-backed chair
6. Locking file cabinet
7. Bookshelf
8. Typing stand
9. Power bar
10. Surge protector
11. Cleaning supplies
12. Fire extinguisher
13. Dust covers
14. Diskette file

### Setting Up The Computer

Now, at last, we have everything set up in the computer work area, and we are ready to install the object of all this attention. Here are some preliminary notes on getting started.

Save all the packing materials (box, styrofoam protective packaging, and so forth) in case you have to return a defective device or move the computer somewhere else at a later date. Immediately record the serial number and unit number of each device, both for insurance purposes and for communicating with dealers over the phone.

Have a clean, uncluttered place to unpack and set up the computer (out of the main stream of traffic in your home), and have all your tools ready. All that should be necessary are several sizes of flat-bladed screwdrivers and perhaps a knife to cut the packing tape that secures the shipping boxes.

Be particularly careful to keep track of every piece when you are assembling your PCjr. One of the easiest and most frustrating things to do during any type of computer repair or assembly is to lose a 39¢ screw, without which your \$2000 system is totally paralyzed.

Now, let's have a look at what you've bought. The basic IBM PCjr line is given in the accompanying checklist. Use this checklist to compare what you ordered with what you have actually received.



### ***IBM PCjr Products Available***

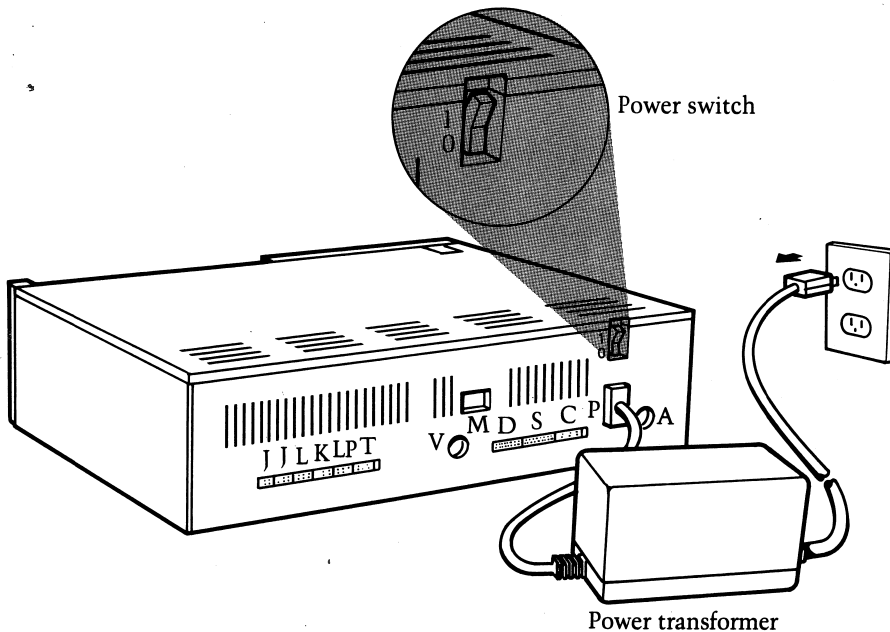
<b><i>Name</i></b>	<b><i>Ordered</i></b>	<b><i>Received</i></b>	<b><i>Serial Number</i></b>
System unit: a. 64 kb or b. 128 kb with 320 kb			
Disk drive (Note: includes separate transformer)			
Monitor: a. monochrome or b. color			
Printer: a. dot matrix or b. letter quality			

(continued)

<b>Name</b>	<b>Ordered</b>	<b>Received</b>	<b>Serial Number</b>
Modem			
Joystick			
Cables: a. color monitor adapter b. keyboard connector c. cassette adapter d. serial devices adapter			
Carrying case			
Cartridges (record name): a. IBM PCjr BASIC b. _____ c. _____ d. _____			
Software (record name): a. _____ b. _____ c. _____ d. _____ e. _____			
Blank diskettes			
Printer paper			
Manuals and other publications: a. <i>Guide to Operations</i> b. <i>Technical Reference Manual</i> c. <i>Service Manual</i> d. <i>Disk Operating System Reference Manual</i> e. <i>BASIC Made Easy</i> f. <i>The Complete Guide to Success with the PCjr</i> g. _____ h. _____			

Note that not everything on this list will be included automatically with your computer order.

On to business. Start with the large box labeled "system unit." Open it and remove the packing material to get out the enclosed computer and transformer. Check to be sure that you have received the correct version you ordered. Does it have a disk drive? Find the *Guide to Operations* manual inside the box, and look it over before you do anything else. Remove the warranty cards and set them aside in a safe place. Papers tend to get lost in the excitement of unwrapping your new toy. Fill out the warranty as soon as you are sure that you have received all the parts you have ordered. Set up the transformer on the floor below your computer desk. Figure 3-5 illustrates the rear of the PCjr. Run the cables connecting the



**Figure 3-5** Rear view of the PCjr showing power transformer and cabling locations.

J	Left joystick	M	Telephone for modem
J	Right joystick	D	Direct drive video
L	Spare	S	Serial port for compact printer or other serial device
K	Keyboard	C	???
LP	Light pen	A	Audio
T	Television	P	Power
V	Composite video	1/0	On/Off for power switch

transformer to the system unit up behind the desk and leave them accessible on the desktop. Open your *Guide to Operations* to the section on setting up the system unit, and follow the instructions closely. Everything in the assembly and installation of your PCjr should go smoothly and easily. Parts, if properly aligned and undamaged, will fit readily and snugly into place. *Never force any part against resistance!* If you are having difficulty getting two parts to fit together, *stop!* Check that you have the correct parts and that you have lined them up properly.

Remember, when all else fails, *read the instructions*. Setting up your PCjr should be quick and easy. If it isn't, don't press on blindly—stop and find out what is wrong. If necessary call the dealer who sold you the computer and ask for help. A \$2000 system is too expensive to ruin trying to jam a \$20 cable into the wrong hole.

Once your system is completely assembled, you will want to run the test programs. These are programs that check the computer's functions to be certain that all parts are working properly. Follow the *PCjr Guide to Operations* in running the test programs. After you have completed the test programs, you are ready to get down to some serious computing. Congratulations! Turn to Chapter 4 and let's start using your PCjr.

## Further Reading

*Don't* by Rodney Zaks (Sybex, 1981). As the title suggests, this is an extensive compilation of things *not* to do with your computer. Because it is oriented toward industrial computers, not all information is relevant to home computers.

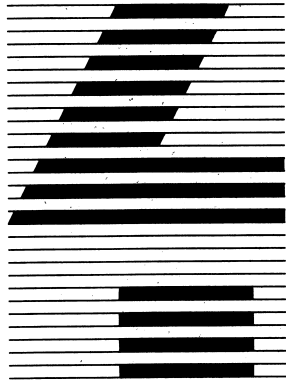
*The IBM PCjr Documentation* supplied with your PCjr purchase. Yes, read those manuals. They will answer a lot of questions about your system. Furthermore, IBM documentation is the envy of the industry for its attention to detail and excellence in presentation. The three manuals most helpful at this point are:

*IBM PCjr Guide to Operations* (IBM part number 1502292)

*IBM PCjr Hardware, Maintenance and Service Manual* (IBM part number 1502294)

*IBM PCjr Technical Reference Manual* (IBM part number 1502293)

*National Institute of Occupational Safety and Health* (Publication No. 81-129). This NIOSH study examined the problems of professional computer operators and came to the conclusion that the design of the computer work station is one of the most important aspects of safe and successful computer operations. If you're interested in such technical details, this study can make interesting reading.



## Getting Results: Running A Program Makes It Happen

Now that your *PCjr* is set up and checked out, it's time to get results. Up to now we've talked about the various "hardware" devices of the computer as separate units. Now all these individual pieces will be working together as a single, coordinated machine—a computer system—to make it all happen. Obviously, that is why you bought all these individual parts in the first place—to have a personal computer system for your own use.

In this chapter we will learn how to use the computer to run programs, some of which you will be writing yourself. In addition, we will use the computer as an educational tool to teach us about itself as we use it.

All of the programs and games we will discuss either will be a built-in part of the ROM (read-only memory, remember?) supplied with the *PCjr* or will be available on cartridges. We will not discuss programs on floppy disks until the next chapter.

Some of the programs and games we will talk about are not "standard equipment" for the *PCjr*, so you may not own everything described in this chapter. We suggest that you read the appropriate section anyway. Purchasing software is an ongoing process for every computer owner, and one of the main ways to learn about what you want to buy (short of using someone else's program) is by reading the available computer literature, such as this book, or the many magazines available, with their numerous advertisements. Being able to understand written material about computer programs and to decide if such programs are something you want to purchase are valuable skills to acquire.

To be able to run the programs in this chapter, you must have your *PCjr* completely assembled and ready to activate. A copy of IBM *PCjr* Cartridge BASIC is not necessary. The BASIC we will use for now is called Cassette BASIC, and it is built into the *PCjr*. If you do have other cartridge programs (as opposed to floppy disk programs), have them handy, however. A printer is also useful, whether it is the IBM Compact Printer or another type, but you can get along without one too. If you have joysticks, you will want to be sure they are connected.

## Turning On The *PCjr*

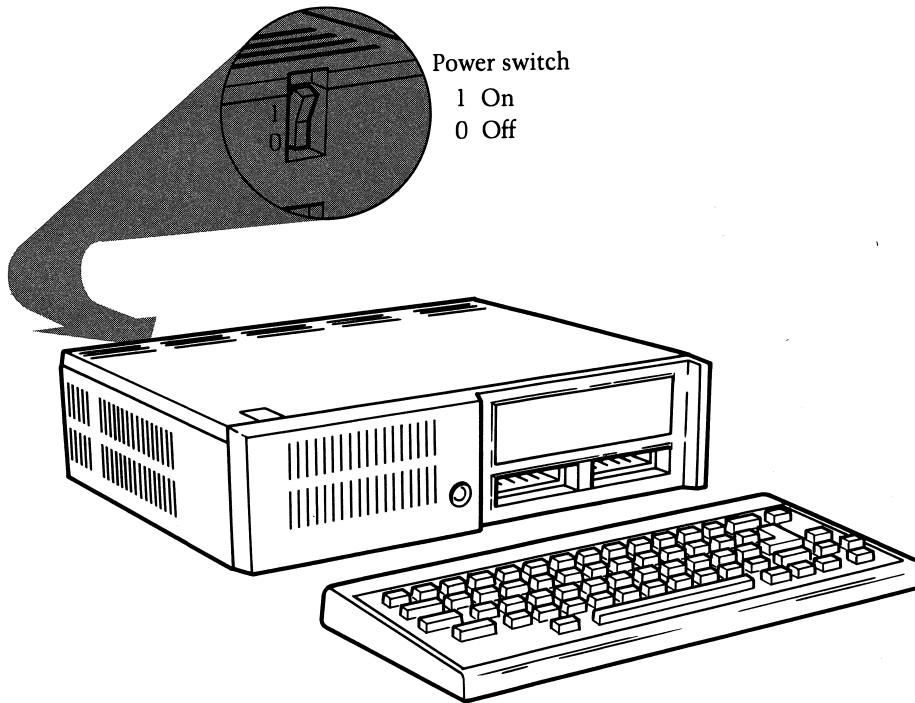
Once you have connected all of the components of your computer, the first step is turning it on. In Chapter 2 we talked about the importance of turning on all your components at the same time (using a power bar) or turning them on in a specific sequence (printer, monitor or TV, and system unit). Doing this will minimize the hazard of dangerous power surges and will add to the life span of your computer system.

You will find the power switch for the system unit on the left-hand side of the rear face of the unit (Figure 4-1). It is a black switch labeled with the code ("0" for "Off" and "1" for "On") that indicates which switch position is which. This 0/1 code is now being used on all types of electronic equipment.

When you turn on any computer, the first thing it does is look for a program to tell it what to do. Remember, a computer can't do anything by itself—it can only follow a program. Your IBM *PCjr* is no exception. Where the *PCjr* begins this search depends on the type of system you have. If you own the disk drive version (the enhanced model), the first place the machine will search is on a diskette inside the drive. So you can expect to see the disk drive power light flash and to hear the whir of the motor as this search takes place.

If a disk drive is not part of your system, or if the drive is empty, the *PCjr* will look next to either of the two cartridge slots built into the system unit. If a cartridge is plugged into one of those slots when the power is turned on, the computer will begin using that program. It's best, however, not to plug in your cartridges until after the power is turned on. The *PCjr* is different from the video





**Figure 4-1** The PCjr power switch is located on the left rear panel of the unit. "On" is marked "1," "Off" is marked "0."

game devices in that respect. It need not be switched off before inserting a cartridge, so the normal practice is to turn on the power, let the PCjr go through its start-up routine, and then insert the cartridge.

When the computer has looked everywhere else and still not found a set of instructions (program) telling it what to do, it assumes that you want to write a program of your own and it offers you some assistance. Built into the PCjr's ROM (read-only memory) is a complete, easy-to-use computer language. With this language the PCjr readily can be converted into a useful problem-solving tool.

## BASIC

This starter language is called BASIC (an acronym for **B**eginner's **A**ll-Purpose **S**ymbolic **I**nstruction **C**ode), and it lives up to its name.

BASIC is designed to be a language people can use right away to write programs, even if they don't have previous experience with computers.

BASIC is one of the closest computer languages to plain old English. By the way, if you ever wondered why computers aren't designed to understand English instead of all these specialized computer languages, the answer is that English and other human languages, although colorful and descriptive, can also be ambiguous. Imagine trying to explain to your very literal PCjr the difference between horseshoes, tennis shoes, and alligator shoes—or even something as simple as the difference between light hair, a light bulb, and a light package!

Like most languages, including computer languages, BASIC has several dialects. *Cassette BASIC* got its name from the fact that it was designed for the early versions of the IBM PC computer to be used with a cassette tape recorder as a storage device. You can use Cassette BASIC this way, but because diskette storage is so superior for personal computer applications—in terms of speed, reliability, capacity, and ease of use—cassette tape recorders are not recommended for use with the PCjr. We will use Cassette BASIC only to illustrate some of its fundamentals as a computer language and in the running of programs. In Chapter 11 you will write programs using the *Cartridge BASIC* available for the PCjr. This version of the BASIC language offers a number of features that are not included in Cassette BASIC.

You will know that you have reached the built-in Cassette BASIC language when you see the following message on your screen:

```
The IBM Personal Computer Basic  
Version C1.20 Copyright IBM Corp 1981  
62940 Bytes free  
Ok
```

At the bottom of the screen, you will find a strange set of numbers and backlighted words:

```
1LIST 1RUN← 3LOAD“ 4SAVE” 5CONT←
```

This list serves as a reminder of the special function keys (color coded Green) that are available for use in BASIC and what they can

do for you. For our first exercise, however, you don't have to worry about function keys.

In fact, the IBM PCjr computer keyboard—with its color codes, special function keys, and so forth—seems at first to be one of the most difficult parts of the computer system to master. So your first task will be to get to know the keyboard. In the process you will learn a little about the PCjr as a computer.

### **Just Follow The Blinking Light**

As you look at the screen that displays your BASIC message, notice the small flashing bar of light positioned below the word "Ok", on the fifth line. This bar of light is called a *cursor*. It functions a lot like the famous "bouncing ball" you followed on those sing-along TV programs. The word on which the bouncing ball landed was the word you were supposed to be singing. Well, the cursor on your display does the same thing as the ball on your television: It indicates where the next letter you type will be placed.

Let's experiment with the cursor by typing:

**This is where the cursor is.**

Next, press the [ENTER] key, marked [↵]. As you type, notice how the cursor keeps pace with you. Watch what happens to it when you press the [ENTER] key. And, of course, watch what fun the PCjr has with your sentence after you finish.

Because the sentence—This is where the cursor is—is written in English (not BASIC), the PCjr doesn't understand the "syntax." That is just another way of saying that, at least in BASIC, the sentence does not compute! When BASIC receives a word or sentence it can't understand, it tells you so by saying:

Syntax error

As you can see, there are no hard feelings. The computer tells you it's "Ok" and gives you another chance to enter more information. ("Ok" is known as the BASIC prompt. It means that BASIC is ready for action.)

Let's ignore the syntax error for a moment so we can learn a little more about the cursor. Type the following sentence—mistakes and all—but *don't* press the [ENTER] key just yet:

### What if i make a mistake?

The cursor should be in the position just after the question mark—if you haven't pressed [ENTER]. If you did press [ENTER], then you got the syntax error message again, so go ahead and retype the sentence *without* pressing [ENTER]. If you didn't get the sentence typed exactly the way it appears in this book, please press [ENTER], and after the next "Ok" is displayed, retype the sentence exactly as it is shown.

Now you will learn more about the cursor. Look at the lower right-hand corner of your PCjr keyboard. You will see a diamond-shaped arrangement of four keys, each with an arrow pointing either north, east, south, or west printed on the keyboard next to it (Figure 4-2). These are the cursor control keys. They enable you to move the cursor anywhere you want on the display screen (or even off it at times). Also notice, above the reversed-L-shaped [ENTER] key, the [BACKSPACE] key, marked with a nearby large arrow [←]. With these keys you are going to correct the mistakes—I mean mistakes—in the sentence you have just typed.

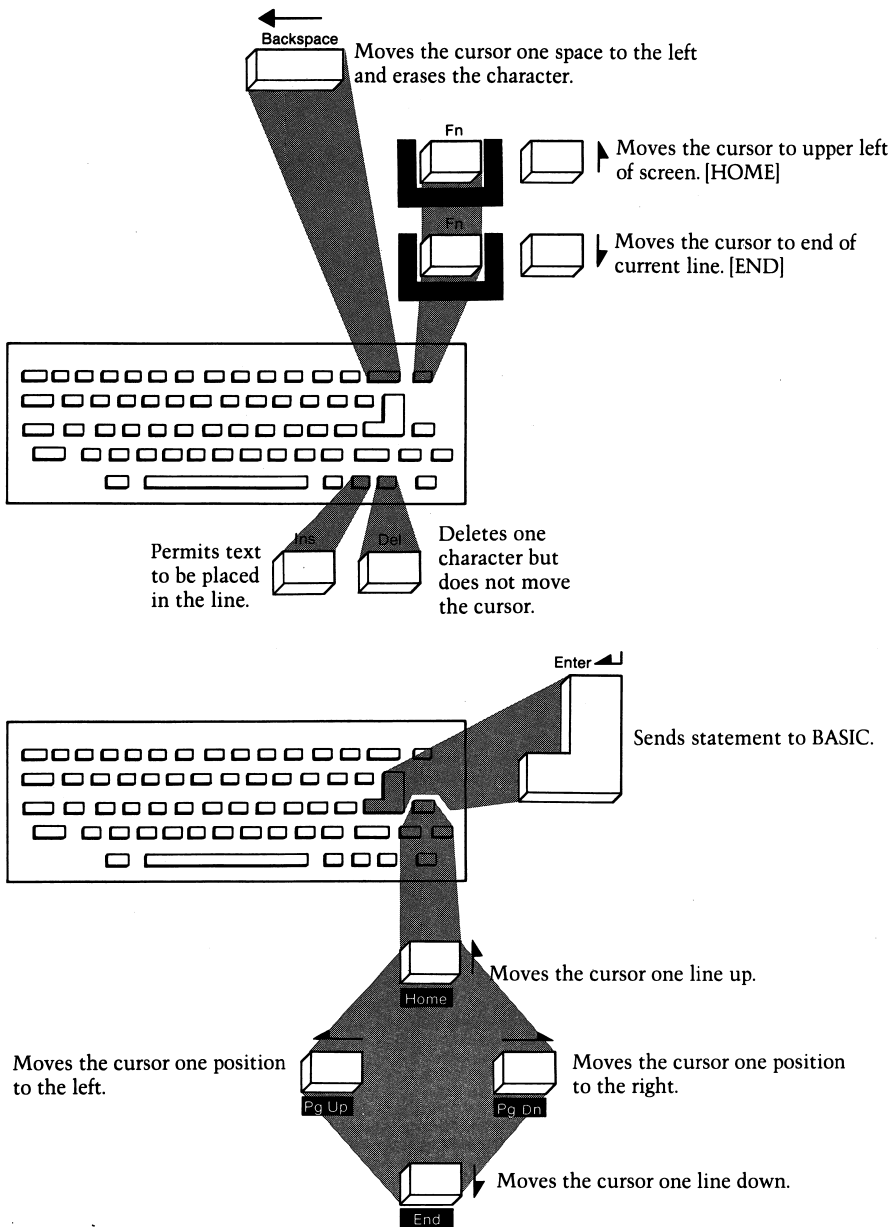
First, use the [←] key of the cursor control diamond to move the cursor to the left until it is underneath the lone "i" that needs to be capitalized. When the cursor is properly positioned, type:

I

If you go too far to the left, don't worry. Use the [→] (*not* the [SPACEBAR]) to get back. Now use the [→] to move the cursor back to beneath the "a" in "mistake?" Use the [BACKSPACE] key to back the cursor one space over to the "e." What happened?

The [BACKSPACE] key, at least in BASIC, does more than just back up the cursor one space; it actually erases anything displayed one space in back of the cursor. Now move the cursor over to the right to the "?" and type:

e?



**Figure 4-2** Cursor control keys.

## Getting Results: Running A Program Makes It Happen

You should now have a sentence grammatically correct enough to please any English teacher. Press [ENTER]. Unfortunately, even though our sentence was written in perfect English, it still doesn't work as BASIC, so we get another syntax error message.

Let's do one more sentence to see just how far the cursor will go. Type:

### How far will the cursor go?

*without* pressing the [ENTER] key. Use the [←] to move the cursor to the left-hand side of the screen. Now move it two spaces farther to the left. What happened? The cursor seems to have vanished, only to reappear at the right-hand edge of the line above!

The cursor is not limited to motion only in the direction of the arrows. When the cursor reaches the end of the line, it moves to the next space—even though that means moving up or down.

Now use the [↑] and [↓] arrows, as well as the [←] and [→] ones, until you are comfortable using the cursor control keys to move the cursor to any spot on the screen.

When you are, return the cursor to the end of the line you have just typed and press [ENTER]. You will get the same "Syntax error" and "Ok" response from BASIC. Now type:

### This exercise is to illustrate the Ins, Del, Home, and End keys

Do not press [ENTER]. Instead, move the cursor to the first comma in the sentence (the comma after the "s" in Ins), press the [INS] key and type:

**ert**

Notice how the letters are inserted after the "s" and how the remainder of the sentence is shifted to the right to make room for the new letters. Also notice that the cursor has become a blinking square rather than a blinking line. The change in shape of the cursor indicates that any new text you type will be inserted into the line at that point. In computerese the blinking-square cursor indicates that you are in the *insert mode*.

Next, move the cursor to the second comma (the comma after the "l" in Del). Notice that the cursor returned to the standard blinking-line form. (As soon as you press a cursor control key, you leave the insert mode.) Press the [DEL] key. When you pressed the [DEL] key, the comma disappeared and the rest of the sentence moved left to close up the space. In BASIC the [DEL] key erases a letter but does not move the cursor.

Now move the cursor to the beginning of the sentence, then press the [END] key. Just as you expected, the cursor moved to the end of the line.

Finally, press the [HOME] key. This will cause the cursor to move to the upper left corner of the screen. That is the home position for the cursor in the PCjr.

In your work and play with the PCjr, you will make considerable use of these cursor control keys, so continue to experiment with them until you feel confident you know exactly what each will do. Now that you have mastered the cursor control keys of the PCjr, you are ready to embark on your next adventure.

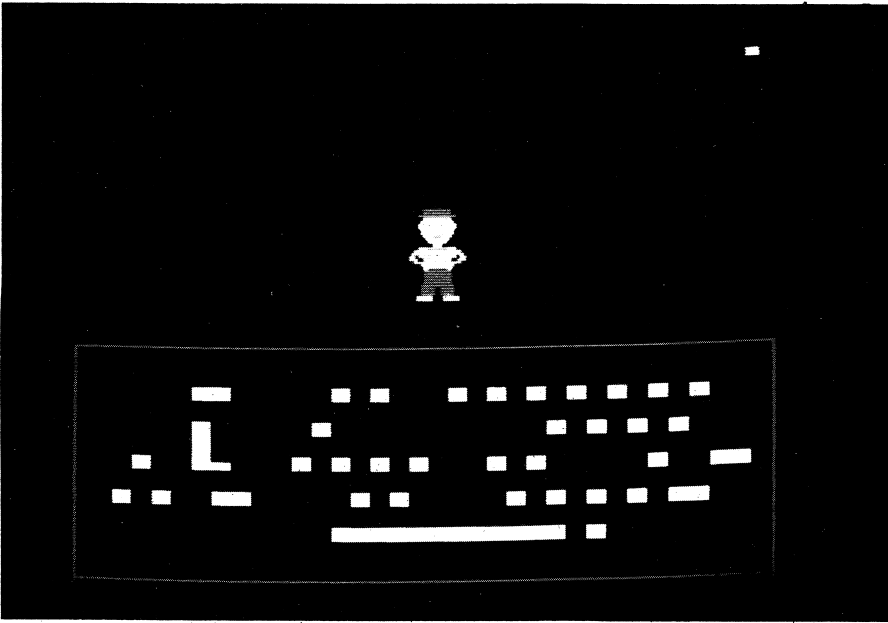
### **Taking A Keyboard Adventure**

Be sure you are in BASIC and have the cursor placed after an "Ok" message. If you do not, return to the beginning section of this chapter (which describes turning on your PCjr) and get back to BASIC and the "Ok" message.

You have just gone through a basic (no pun intended) introduction to the cursor and how it moves. Now you will learn more about the cursor and all the other parts of the PCjr keyboard. At the same time you will see just how expert and patient a teacher the PCjr can be.

Built into every IBM PCjr is a special teaching program called Keyboard Adventure. This is an *interactive* program, which means you talk to the computer and it talks to you. It will teach you how to use all sixty-two keys of the PCjr keyboard, including the special function keys.

To use Keyboard Adventure, you must remove any cartridges or diskettes from the computer. Keyboard Adventure can only be called up from Cassette BASIC and must be the first program you



The PCjr has a built-in program, Keyboard Adventure, to illustrate the use of the keyboard's special keys.

run after turning on the computer. So after you have cleared the computer of any cartridges or diskettes, simply turn it off and—after a five-second pause—turn it back on.

When you see the "Ok" from BASIC, press the Escape [ESC] key, located in the upper left-hand corner of the keyboard and color coded red. This is your key for entering the Keyboard Adventure program.

As you go through Keyboard Adventure, follow along in your *Guide to Operations* manual, beginning on page 2–6. Take your time and enjoy running your first computer program. When you return, we will explore the cartridge programs available for the PCjr. Then you will get a chance to write programs of your own. In case you forget what the various keys do, no need to interrupt your work to return to Keyboard Adventure. The following keyboard checklist contains the name and function of each key or key combination on the PCjr.





## ***BASIC Commands Coded By Function Keys***

---

<b>Fn</b>	<b>1</b>	<b>LIST</b>	Lists the program currently in memory
<b>Fn</b>	<b>2</b>	<b>RUN←</b>	Runs the program currently in memory
<b>Fn</b>	<b>3</b>	<b>LOAD"</b>	Loads a program into memory
<b>Fn</b>	<b>4</b>	<b>SAVE"</b>	Saves the current program under the specified name
<b>Fn</b>	<b>5</b>	<b>CONT←</b>	Resumes running a program (after a break or error)
<b>Fn</b>	<b>6</b>	<b>, "LPT1:"←</b>	Prints data on the printer
<b>Fn</b>	<b>7</b>	<b>TRON←</b>	Turns on the tracer
<b>Fn</b>	<b>8</b>	<b>TROFF←</b>	Turns off the tracer
<b>Fn</b>	<b>9</b>	<b>KEY</b>	Sets or displays the function keys
<b>Fn</b>	<b>0</b>	<b>SCREEN 0,0,0←</b>	Sets screen function to text mode, black and white

## ***PCjr Cartridges***

Cartridges, as we have pointed out, are sturdy, self-contained units of read-only memory (ROM), about the size of a pack of cigarettes. They can contain a variety of programs. When IBM first introduced the PCjr, five different cartridges were available: four games and PCjr Cartridge BASIC. The four games are "Crossfire," "Mine Shaft," "Mouser," and "ScubaVenture."

We will take "Crossfire" as a typical example of how cartridge games work. In this game you are in control of a missile-firing

spaceship assigned to protect a futuristic city from swarms of attacking insects. "Crossfire" can be played alone or with another player, and you can use either the PCjr joysticks or the keyboard to control your spacecraft (Figure 4-3).

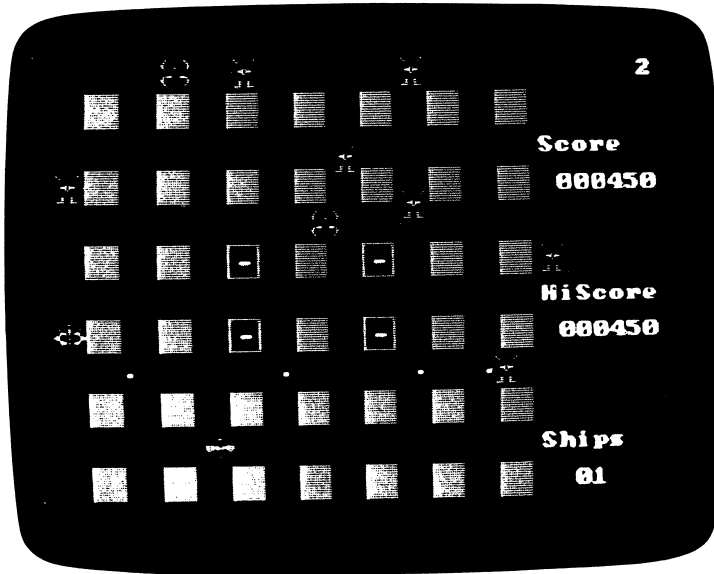
Before we discuss the actual game, let's learn about using cartridges in general. When a cartridge is inserted into either cartridge slot, it sends a signal to the computer to clear its internal memory and to enter the program contained in the cartridge. This is the equivalent of the computer's being turned on anew with the cartridge in place. As such, the screen clears and the opening message or title of the new program is displayed. Whatever you were working on before will be *lost* whenever you insert a new cartridge, so remember to take the appropriate precautions to save your work before going on to game time. This warning is particularly important if you are sharing your PCjr with younger members of the family. Few things are more frustrating (or more damaging to family harmony) than returning from a break only to find your last two hours of work on the "Budget Report" erased because your child rushed home from school with a friend to play "Mouser."

With these precautions in mind, insert your "Crossfire" cartridge, if you have one. Otherwise, just follow along, because most of the principles are similar for all of the cartridge games. Remember, never force or jam a cartridge (or any other piece of computer equipment) into position. It should snap securely into place with a minimum of effort. Always insert the cartridge with the label side up.

You may wonder why there are two cartridge slots on the PCjr. The two slots allow Cartridge BASIC to be inserted to run a second cartridge containing a BASIC program.

After you have inserted the game cartridge, the screen should show the introductory graphics (pictorial) display that lets you know you are about to play "Crossfire."

Before you begin playing, you must give the PCjr some details about how you want the game played. You need to specify whether you will be using the joysticks or the keyboard to control play, what color scheme you desire (if you have a color monitor), the skill level (easy or hard), and the number of people who will be playing. To do this, you simply answer the series of questions displayed at the start of the game session. This minor cross-examination is conducted in a similar manner for almost all computer games.



**Figure 4-3** "Crossfire" is one of the cartridge games available for the PCjr.

Now you are ready to actually play "Crossfire," so go ahead and give it a shot. The first few times you play, notice how the joystick controls the action. If you don't have a joystick, this is a good chance to see if using the keyboard conveys enough of the "feel" of the game to make it enjoyable. If not, investing the \$40 or so in a joystick may be a good way to prevent the money you spent on the game cartridge from being wasted by loss of interest in the game. See how it feels to play a game on your new PCjr computer. And above all, enjoy!

## Writing And Running Your Own Programs

So far, we have used the computer for education (to teach us about a part of itself—namely, the keyboard) and for entertainment (game playing). Later, we will tackle some of the more sophisticated applications of the PCjr, such as word processing, budget management, and record keeping. But now it is time to write some fairly simple computer programs yourself—just so you know you can do it.

We will program in a language that you already have been introduced to—BASIC. If you have the IBM PCjr Cartridge BASIC, you can now insert it into the cartridge slot. But, as we said at the start of the chapter, all the commands you will use in this section are built into the PCjr's ROM. So if you don't have Cartridge BASIC, just return to the section "Turning On the PCjr" to get back to Cassette BASIC and the BASIC prompt "Ok."

All the dialects of BASIC understand two kinds of entries: *commands* and *statements*. A command is just that—an instruction to be executed immediately by the computer. A statement is a sentence, understandable to BASIC, that forms a part of a program. Every statement has its own number. When BASIC receives the command to run the program, it executes each statement of the program in numerical order, from the lowest statement number to the highest. The statement numbers used in a program need not have any specific increment. For example, statements numbered 1, 5, 68, 120 will execute in the same order as those numbered 10, 20, 30, 40.

Statements are also called *indirect commands*, because they are not executed immediately. They are only executed when the program is run.

As we promised, BASIC is designed to be simple, and it is. We will start with an easy program and work our way up to more difficult examples. Our first program will add three numbers (4, 5, and 6) and tell us the result—a simple enough start. It will illustrate a number of features of BASIC programming.

## Program 1: A Simple Arithmetic Problem

After you get the "Ok" from BASIC, press the [CAPS LOCK] key and type the following two lines:

```
10 PRINT 4 + 5 + 6 ↵  
20 END ↵
```

The ↵ symbol is a reminder to press the [ENTER] after each line of a BASIC program.

Notice that each line of the BASIC program is numbered in units of ten. The first line could have been numbered 1, and the

second 2, and they would have executed in the same order, as we mentioned earlier. A better program style, however, is to leave some "room" in the statement numbering scheme in case you want to add any statements to the program later. Numbering in units of ten (which is standard practice in BASIC) gives you plenty of space for additional statements—nine available numbers between lines.

Unlike earlier, when you were typing English language sentences into the computer, you should get no syntax errors after you press [ENTER]. This is because words like PRINT and END have meaning in BASIC, even though they also have meaning in English (but perhaps in a different way). In fact, one of the strongest assets of the BASIC language is its use of English words as key words. This aspect makes BASIC easy to read, even if you are not familiar with programming languages, as you can see.

Back to your program. Nothing earthshaking should be happening now. BASIC should just have swallowed your statements and be patiently holding them until you indicate what you want done. For starters, let's issue a command that will check to see how much BASIC remembers of what you typed. That command is the LIST command. Type:

**LIST**

and press [ENTER].

The LIST command causes the BASIC program statements stored in memory to be displayed. BASIC commands are executed immediately (remember the rules about commands), so you should see the program displayed as written, followed by the "Ok" prompt.

The screen should look like this:

```
10 PRINT 4 + 5 + 6
20 END
LIST
10 PRINT 4 + 5 + 6
20 END
Ok
```

Everything looks fine, so let's run the program. Reasonably enough, the BASIC command for running a program is RUN. Type:

**RUN**

and press [ENTER]. The results displayed are:

```
15  
Ok
```

Congratulations! Your first program was a success. Not much of an achievement, you say? It would have been quicker and a lot less expensive to use a hand calculator? You're only partially right. Some calculation jobs are easier to do with a hand calculator, but many, many more are much easier with a computer.

We chose this arithmetic problem to illustrate some features of BASIC, rather than to accomplish a useful task. Let's review these features.

1. A BASIC program is composed of statements.
2. Each statement has a line number followed by a key word that indicates the operations to be performed by the computer. The key words used in this program were PRINT and END. The PRINT statement told the computer to add the three numbers and display the results. The END statement told the computer that the program was finished. It indicated that line 20 was the last statement in the program. (There are some seventy such key words in the BASIC language that are available for the PCjr. A list of these key words is in Chapter 11.)
3. When required, the key words are followed by a list of items, numbers, or words that the computer must use to carry out the action. The PRINT statement listed three numbers to be added. In programming these items have special names: *constants* and *variables*. Actual values of numbers like the 4, 5, 6 listed in the PRINT statement are called constants. A variable is merely a quantity in a program that can assume any value during the execution of a program.
4. You must press [ENTER] to enter the BASIC statement into the computer memory or to activate a BASIC command.
5. BASIC commands are executed immediately. Usually, they are one word long and do not use line numbers. The RUN command causes the program to be executed by the computer, whereas the LIST command displays the program statements stored in the com-

puter memory. (There are about twenty-five commands available with PCjr BASIC.)

That is quite a bit to learn from a two-line program, isn't it?

## Program 2: Metric Conversions

This program will be a little more challenging. A few years ago there was a tremendous rush to go metric. Distance was going to be measured in kilometers, volume in liters, and temperature in degrees centigrade. At the same time, there was an explosion in gasoline prices and the older gas pumps could not handle the dollar-plus-per-gallon prices because of their mechanical calculators. Some oil company genius decided to save the cost of replacing all those outdated pump mechanisms, and to give the gasoline-consuming public a lesson in metrics at the same time: Gasoline would be dispensed in liters.

Program 2 will be your ready reckoner to convert from gallons to liters and back again so that you can keep track of the miles per gallon your car is getting from that high-priced fuel. Get back to the "Ok" prompt, then type:

### NEW

The NEW command erases the computer memory. Always start your BASIC programming session with a NEW command. This command is equivalent to erasing a blackboard before writing on it. It gets rid of any garbage, in the form of old BASIC statements hanging around from previous programs and threatening to mess up your new one.

Just type the entire Program 2 for now. Be sure to press [ENTER] after every statement. Because Program 2 is much larger than Program 1, you will find the simple but powerful editing features available with PCjr BASIC handy to correct any typing errors. List the statements of the program occasionally with the LIST command to be sure you have not made any typing errors.

This time use the function key [F1] to obtain the LIST command. Remember to first press the green function key and then the

[1] key. The word LIST will then be displayed. If you press [ENTER], the entire program will be listed starting with the lowest-numbered statement. The LIST display really is not designed for people, because it flashes the BASIC statements on the screen in rapid-fire fashion. If you press the Function [FN] key and the [Q] key ([PAUSE]) at the same time, you will pause the display. To start it again, press any key at all.

However, you can use the LIST command to display only a portion of the program as shown below:

### LIST - 100

This will list the portion of the program starting with the lowest-numbered statement, to statement 100.

### LIST 100 -

This will list the program starting with statement 100, to the largest-numbered statement.

### LIST 100

This will list only statement 100.

### LIST 100-200

This will list from statement 100 to statement 200.

Type Program 2:

```

10 CLS
20 PRINT " ** THIS PROGRAM CONVERTS LITERS TO GALLONS AND VICE VERSA **"
30 PRINT "ENTER 1 AFTER THE ? IS DISPLAYED TO GO FROM LITERS TO GALLONS"
40 PRINT "ENTER 2 AFTER THE ? IS DISPLAYED TO GO FROM GALLONS TO LITERS"
50 PRINT
60 INPUT CHOICE
70 IF CHOICE = 2 THEN GOTO 150
80 PRINT
90 PRINT "ENTER NUMBER OF LITERS AFTER THE ? IS DISPLAYED"
100 INPUT LITERS
110 GALLONS = LITERS/3.785
120 PRINT
130 PRINT "THERE ARE ";GALLONS;" GALLONS IN ";LITERS;" LITERS"
140 GOTO 999

```



```
150 PRINT
160 PRINT "ENTER NUMBER OF GALLONS AFTER THE ? IS DISPLAYED"
170 INPUT GALLONS
180 LITERS = 3.785*GALLONS
190 PRINT
200 PRINT "THERE ARE ";LITERS;" LITERS IN ";GALLONS;" GALLONS"
999 END
```

Once you have typed Program 2, try running it. Remember, the BASIC command RUN executes a program. This time, use the function key [F2] to produce the RUN command. Notice that as soon as you depress [F2], the program begins executing. [F2] not only generates the RUN command, it enters the command as well. Be sure to press [ENTER] when you enter data into the program.

You will notice that the first thing the program does is clear the screen. The Clear Screen (CLS) command in line 10 does that job. This also shows that you can use certain BASIC commands in programs as well as during program preparation. You can use the CLS command to clear the screen any time.

Notice the screen display is limited to forty characters. With the enhanced version of the PCjr, however, you can improve the appearance of the output very easily by typing:

### WIDTH 80

after you receive the Ok prompt in BASIC.

This extends the screen to its full eighty-column capacity. An eighty-column screen is the standard display width in data processing, so remember this command and use it whenever you write BASIC programs.

Another new statement is INPUT (used in lines 60, 100, and 170). The INPUT statement allows the program to receive data entered at the keyboard. The value you enter at the keyboard is given to the variable named in the INPUT statement.

In the INPUT statement at line 60, for example, the value 1 or 2 is given to the variable CHOICE. Variables are given names in BASIC for easy reference. In Program 2 the variables are called CHOICE (in line 60), GALLONS (in lines 110, 130, 170, 180, and 200), and LITERS (in lines 100, 110, 130, 180, and 200).

## Getting Results: Running A Program Makes It Happen

In lines 110 and 180, the program performs the required calculations. Notice that the arithmetic operators are not always the same as the symbols you are used to:

Addition	+
Subtraction	-
Multiplication	*
Division	/

The calculation done in line 110 is typical of those in BASIC programs. Here the value entered for LITERS is divided by 3.785 to produce the value of GALLONS.

The PRINT statement in this program is used in three different but similar ways. In lines 20, 30, and 40, for example, the PRINT statement is used to display the comment between the quotation marks as directions to the user of the program. In lines 50 and 80, the PRINT statement is used, but there is no comment for it to display. This use of the statement, known as using an "empty PRINT statement," merely creates a blank line on the screen to make the display easier to read.

In lines 130 and 200, the PRINT statement is used to display both a comment and the values of the variables GALLONS and LITERS. The PRINT statement:

```
130 PRINT "THERE ARE ";GALLONS;" GALLONS IN ";LITERS;"
LITERS"
```

shows how the PRINT statement generally is used in BASIC.

The comments in the statement between the quotes:

```
THERE ARE
GALLONS IN
LITERS
```

are displayed directly on the same line. The numeric values entered or calculated for the variables GALLONS and LITERS are displayed in the line at the designated positions. The semicolon in the PRINT statement indicates that the values are to be embedded in the display line. The semicolon in the PRINT statement is called a *cursor*

*control character* because it dictates where the next item in a PRINT statement will be positioned.

This program contains two more statement types: the IF/THEN statement at line 70 and the GOTO statement at line 140. Both are “flow of control” statements. That is just computer talk for statements that change the number of the next statement executed by the BASIC program. Let’s look at the GOTO statement first. When BASIC encounters a GOTO statement, it “goes to” the line number indicated in the GOTO statement and executes that program statement next. For example, if the statement said:

**140 GOTO 999**

the program would go to line 999 as the next one to be executed. As you can see, BASIC uses the GOTO statement to move the processing operation to a different portion of the program. In Program 2, the GOTO statement at 140 avoids the liters calculation if the gallons conversion is selected.

The IF/THEN statement:

**70 IF CHOICE = 2 THEN GOTO 150**

is also called a decision statement. It works like this:

1. The value in the variable CHOICE is compared with the number 2.
2. If CHOICE is equal to 2, then the next statement executed will be at line 150.
3. Otherwise, the statement immediately following the IF statement will be executed.

Every programming language has a statement similar in action to BASIC’s IF/THEN statement.

Again, that was a lot of material to be covered by a program containing only 21 lines, so let’s recap:

1. The three statements—INPUT, GOTO, and IF/THEN—and the commands—NEW, CLS (Clear Screen), and WIDTH—were added to your BASIC vocabulary.

2. Variables, as well as constants, were used in the program.
3. BASIC arithmetic operators for addition, subtraction, multiplication, and division were discussed (although only division actually was used in the program).
4. The use of the PRINT statement was extended to include the capability to display comments, generate blank lines, and show the values of variables.

### Program 3: Guessing Game

The third, and last, program of this section is different from the first two. This time, you will write a simple guessing-game program. The program structure is slightly more complicated than either of the others, but there really isn't anything new in the BASIC statements within the program.

#### Typing Tips

Type:

#### NEW

clear the screen, and limber up your fingers. First, we're going to give you a hint to save you quite a bit of typing. PCjr BASIC takes advantage of the Alternate key of your keyboard to make entering BASIC key words a lot easier. If you simultaneously press the Alternate key [ALT] and the key marked [P], this combination of key strokes will generate the PRINT key word. PCjr BASIC offers a number of these Alternate-key-and-letter pairings to produce BASIC key words.

The Alternate key checklist below gives the key combinations and the key words they generate. If you remember to use the Alternate key, you can save yourself much typing in your program preparation. And when you start producing a lot of programs of your own, you'll really learn to depend on this shorthand method. (Complete details on these key words are presented in the BASIC manual.)



## *BASIC Keywords Coded By Alternate Keys*

ALT	A	AUTO	ALT	N	NEXT
ALT	B	BSAVE	ALT	O	OPEN
ALT	C	COLOR	ALT	P	PRINT
ALT	D	DELETE	ALT	Q	(none)
ALT	E	ELSE	ALT	R	RUN
ALT	F	FOR	ALT	S	SCREEN
ALT	G	GOTO	ALT	T	THEN
ALT	H	HEX\$	ALT	U	USING
ALT	I	INPUT	ALT	V	VAL
ALT	J	(none)	ALT	W	WIDTH
ALT	K	KEY	ALT	X	XOR
ALT	L	LOCATE	ALT	Y	(none)
ALT	M	MOTOR	ALT	Z	(none)

Now type Program 3:

```

5 CLS
10 PRINT "          ***** HIGH - LOW *****"
20 PRINT "          ** THIS IS A GUESSING GAME IN **"
30 PRINT "          ** WHICH THE PCjr WILL TRY TO **"
40 PRINT "          **      GUESS YOUR NUMBER      **"
50 PRINT "          *****"
60 PRINT : PRINT
70 PRINT "TO PLAY - YOU SELECT A WHOLE NUMBER BETWEEN 1 AND 99"
80 PRINT "THE PCjr WILL GUESS A NUMBER, IF THE GUESS IS WRONG"
90 PRINT "JUST ANSWER H - FOR HIGH, OR L - FOR LOW, IF THE GUESS IS LOW"
100 PRINT "OR C - FOR CORRECT IF THE GUESS IS CORRECT"
110 PRINT : PRINT "THE PCjr HAS SIX CHANCES TO GUESS THE NUMBER"
120 PRINT "WHEN YOU ARE READY TO PLAY PRESS THE ENTER KEY";

```

```

125 PC=0 : YOU=0 : GAME=1
130 INPUT START$
135 COUNT = 0
140 UPPER = 99
150 LOWER = 1
160 GUESS = INT((UPPER + LOWER)/2)
170 COUNT = COUNT + 1
180 IF COUNT > 6 THEN GOTO 600
185 PRINT
190 PRINT "GUESS NUMBER";COUNT;" IS ";GUESS
200 PRINT
210 PRINT "IS THE GUESS HIGH, LOW, OR CORRECT";
215 INPUT ANSWER$
220 IF ANSWER$ = "H" THEN GOTO 300
230 IF ANSWER$ = "L" THEN GOTO 400
240 IF ANSWER$ = "C" THEN GOTO 500
250 PRINT "THE ANSWER SHOULD BE H, L, OR C. PLEASE ENTER ONE OF THESE";
260 INPUT ANSWER$ : GOTO 220
300 UPPER = GUESS
310 GOTO 160
400 LOWER = GUESS
410 GOTO 160
500 PRINT : PRINT "THE PC/Jr WON THIS ONE. WOULD YOU LIKE A REMATCH?"
510 PC=PC+1
520 GOTO 700
600 PRINT : PRINT "CONGRATULATIONS -- YOU WON "
610 PRINT "CAN I HAVE ANOTHER TRY?"
620 YOU = YOU+1
700 PRINT "IF SO JUST ENTER Y, IF NOT ENTER E TO END THE GAME";
710 INPUT REMATCH$
720 IF REMATCH$ = "Y" THEN GOTO 730 ELSE GOTO 800
730 GAME = GAME+1
740 CLS
750 PRINT "HAVE YOU SELECTED A NUMBER? GREAT!"
760 PRINT "WILL YOU PRESS ENTER TO START GAME ";GAME; " ";
770 PRINT : PRINT
780 GOTO 130
800 CLS
810 PRINT "*****"
820 PRINT "*"
830 PRINT "          SCOREBOARD          *"
840 PRINT "*"
850 PRINT "*"      PC/Jr  ";PC;"          YOU ";YOU;"      *"
860 PRINT "*"
870 PRINT "*"
880 PRINT "*****"
999 END

```

## Using The Printer

When you are working with a large program, usually you'll find it much easier to handle printed copy than to work directly from the screen display. For one thing, you can't follow all the material as easily when you are limited to a small screen output. A printed copy, on the other hand, gives you the opportunity to read the entire program. Instead of struggling with the soft copy on the screen, print the program and have a hard look at your progress (or your problems) as often as necessary.

Fortunately, it is easy to generate a printed listing with your PCjr. The [F6] key generates the LPI command. This command directs your output to the printer, as well as to the screen, so that the printer prints exactly what is appearing on the screen. For example, whenever you want to print a listing, just use the [F6] key after the LIST command. It's as simple as that, from a programming point of view.

The mechanics of using a printer with a personal computer can be troublesome, until you get the hang of it, but the IBM Compact Printer is particularly easy to use. To be sure everything is set up properly, review your *Guide to Operations* on installing the Compact Printer.

The IBM Graphics Printer and the Epson printer are a bit more complicated to use. Chapter 8, on word processing, includes a checklist for plain-paper, dot-matrix printers like the IBM Graphics Printer and the Epson. If you are using that type of printer on your PCjr, skip forward to Chapter 8 and look at the Printer Checklist before you begin issuing the LPI command. If you follow these steps, you can avoid start-up problems that can be frustrating for even the most zealous novice.

It's time to get back to the program at hand. As you can see, we have made use of several additional features of PCjr Cassette BASIC. The first is the ability to put several BASIC statements on the same line. This is done by using a colon to separate each statement. Lines 60 and 110, for example, use a blank PRINT statement, followed by a second PRINT statement.

In a way the use of multiple statements on a line is a little dangerous, because it can be confusing. But for statements as simple as blank PRINT statements or GOTO statements, putting them on the same line can save time.

Another feature of BASIC illustrated in this program is the use of built-in *functions*. BASIC offers prepackaged programs (or functions, as they are called) for common mathematical operations such as those you have encountered if you have studied trigonometry and algebra. To incorporate these functions into your programs, all you need to do is to use the proper three-letter abbreviation for them.

In this program we are not interested in numbers with fractions. Therefore each guess by the computer must be a whole number. The INT (or integer) function in BASIC handles the job of converting any number to an integer. In our program the computer's guess at the number is calculated in statement 160:

```
160 GUESS = INT ((UPPER + LOWER)/2)
```

Here we're using the INT function to make sure the arithmetic produces whole numbers. The PCjr performs the calculation in this statement as follows:

1. It adds the value of UPPER and LOWER and then divides the result by 2.
2. The INT causes the computer to drop any remainder of the division, and the whole number that results becomes the value of GUESS.

For example, if UPPER was 99 and LOWER was 50, the computer would first add 99 and 50 to get 149. Then it would divide 149 by 2 to yield 74.5. Then the computer would set GUESS equal to 74. (Notice, the INT function does not round off the number, it merely drops the fractional part of the number.)

If this is giving you a math anxiety attack, don't take it too seriously. The example merely is meant to show you another powerful feature of the BASIC language. There is no test on this material at the end of the chapter.

The remaining BASIC feature illustrated in this program is the use of a *loop*. A loop structure allows a program to repeat indefinitely—until you stop it. The statements in lines 250 and 260 remind the player to enter the proper response. Until an H, L, or C is entered,



statements 220 through 260 will be repeated over and over. Hence we use the name "loop" for that type of program structure.

The last new feature in this program is the use of letters instead of numbers as inputs for variables. In BASIC, whenever the value of a variable is to be a word or a letter (alphanumerics, as we call them in computer terminology), the name of the variable must end with a dollar sign. The variable ANSWER\$ (in line 215), for example, expects the user to enter an H, L, or C.

You've learned a lot about BASIC and a lot about running programs in these examples with Cassette BASIC. Unfortunately, Cassette BASIC has a severe limitation. It can be used only with an audio cassette, which makes storing and retrieving programs extremely awkward. If you don't have an audio cassette, you can't save these programs at all. And because you probably don't have an audio cassette attached to your PCjr, this program, like the others you've typed, will be lost as soon as you turn off the computer.

Therefore we are saying a fond farewell to Cassette BASIC—not because of any lack of programming power, but because of its weak storage features. In Chapter 11, when we do some serious BASIC programming, we will use Cartridge BASIC to make sure all our work doesn't disappear when the power is turned off.

## Further Reading

*The Best of Creative Computing.* A series of several volumes featuring extracts from the monthly magazine. (Ahl Computing, Inc., a subsidiary of Ziff-Davis Publishing, Morris Plains, New Jersey, 07950.) Offers a number of games in BASIC, as well as articles about programming and a potpourri of other material.

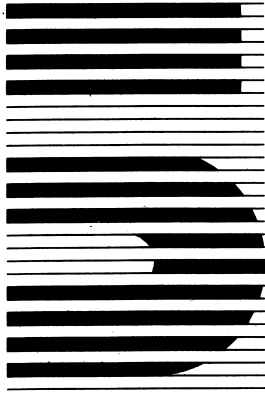
*Illustrating BASIC*, by Donald Alcock (Cambridge University Press, 1979). An overview of BASIC, with an English accent. Contains a lot of examples of the basic BASIC statements and some clever commentary.

*Instant BASIC* (2d ed.), by Jerald R. Brown (Dilithium Press, 1982).

Although the book is not quite as magical as it sounds, it has a number of BASIC programs worked out. It's missing some of the features that make IBM PCjr BASIC unique. However, this text has been a popular introduction to the language.

**Getting Results: Running A Program Makes It Happen**

*Your First BASIC Program*, by Rodnay Zaks (Sybex, 1983). A general introduction to generic BASIC. The book is not specifically oriented to the PC or PCjr environment, but it does have some interesting introductory exercises.



# Operating The Operating System: It's Easier Than You Think

Every large industrial computer center has a specially trained group of individuals called systems programmers. These specialists have one job and one job only—to develop programs that make using the computer easier, more reliable, and more efficient. They function in the background, but they keep the system shipshape and ready to handle the workload.

The tool they work with is the *operating system* (OS), which is actually a collection of special-purpose programs or instructions supplied by the computer manufacturer to increase the productivity of the computer system. One of the exciting parts of owning your own computer is that you avoid these middlemen and get involved with the guts of the OS yourself. This may sound frightening, but it's easier than you think!

## The Operating System

Every computer, including the PCjr, must have an operating system. Without this special collection of programs at the helm, you can't run any of the other programs. Just as a car or train can't move without a driver, a computer can't do anything without a program. So the operating system is actually the first program the computer uses when you turn it on. It gives the machine instructions on how to run all the other programs—the ones for which you really bought the computer.

Why do we need an operating system at all? Well, if there were no general-purpose system available, then all of us would have to

supply our own set of instructions to the computer each time we turned it on. That might not be so bad, once you got the hang of it, but what if you wanted to use a new program that someone else had just written? If it was written on that person's system, the program might well not work with the set of instructions you devised for your computer. You would have to choose between learning a whole new system just to use one program, or not using that program at all.

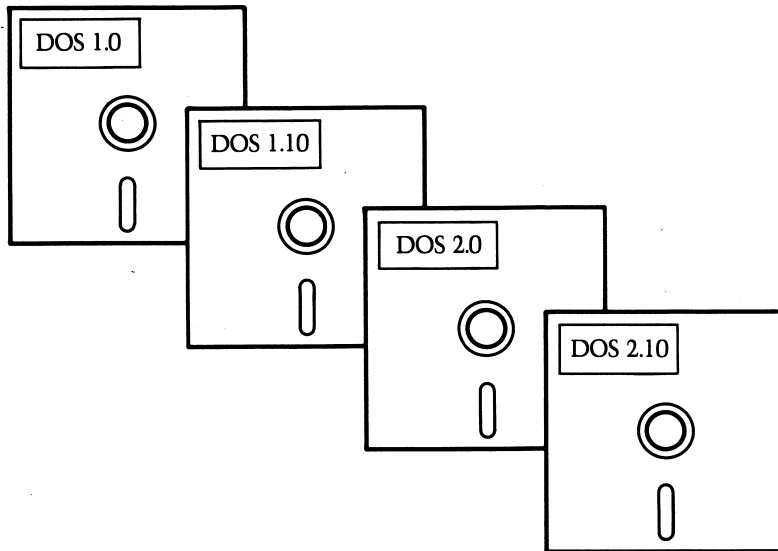
The operating system is a major part of what we call *compatibility*, the ability of different computers to use the same or similar programs. To develop a single operating system that prepares the computer for every possible event that can occur in the course of running a program is an enormous undertaking. Developing a "friendly" operating system that can accommodate all the programs available today and in the foreseeable future takes years of serious effort by knowledgeable and experienced programmers.

## PCjr DOS 2.10

The operating system used with the enhanced version of the IBM PCjr is called DOS 2.10 (for Disk Operating System, version 2.10). If you do not have the enhanced version, you cannot use this operating system. Eventually, though, you will probably want to upgrade your system to take advantage of this extra power, so we suggest that you read this chapter to learn more about DOS.

Only programs designed for use with DOS 2.10 will run successfully on your PCjr, so be careful when you buy new software. Even if the program claims to be "IBM PC compatible," it may not be compatible with the IBM PCjr using DOS 2.10. Compatibility is based on features of the operating system, not the brand name of the computer. Another point to remember is that operating systems are updated regularly, so you should stay informed about the latest version available (Figure 5-1).

In addition to allowing your IBM PCjr to run all other programs, the operating system performs a variety of additional tasks. These functions are necessary regardless of the application you want your computer to perform. For example, all diskette manu-



**Figure 5-1** There are various versions of DOS in use. The latest version is DOS 2.10 for the IBM PCjr, PC, and XT personal computers.

facturers produce a standardized product. The  $5\frac{1}{4}$ -inch double-sided, double-density, soft-sectored diskette you purchased for your PCjr will operate on any number of different computers. But before you use these floppy diskettes, you must format, or prepare, them, no matter what type of use you have in mind. DOS 2.10 supplies a formatting program to set up diskettes for use with the PCjr and the other computers that use the DOS 2.10 system. We'll describe this procedure a little later on in this chapter.

## DOS's Duties

The operating system is able to format diskettes and to check that they are correctly formatted. It can also copy files from one diskette to another (it doesn't matter what information is contained in the file), erase files, list the names of all the files on a given diskette, determine how much space (bytes of memory) is left unused on a

diskette, and much more. You'll be performing these operations time and time again, like housekeeping chores.

To really know all the facets of the PCjr operating system is a big job. The reference manual that comes with your DOS 2.10 operating system diskette is thicker than this book. You probably will find the writing style of the *DOS Reference Manual*—and most other technical computer manuals, for that matter—boring and difficult to read. But don't worry. You can make perfectly good use of your PCjr without understanding any more about DOS 2.10 than we'll cover in this chapter.

IBM and Microsoft, the designers of DOS, understood that the mysteries of the operating system could be extremely formidable to a novice user, so they supplied an excellent and readable *User's Guide* for DOS 2.10. Unfortunately, the guide is set up to cover the entire IBM personal computer product line, and therefore it can be confusing to PCjr users. You'll get more out of the IBM material if you read this section before you tackle the multiple options the IBM literature presents.

If you intend to become a programming expert, you must take time to familiarize yourself with the organization of the *DOS Reference Manual*. You should also have access to the *DOS Technical Reference Manual*, which gives even more details about DOS. Understanding the shorthand notation used in these manuals is difficult, but we will help smooth the way with examples and illustrations. Just remember, if you need detailed technical support, these two manuals are the ultimate references for questions and functions of DOS 2.10.

Computer manuals are deliberately written in a technical style to avoid ambiguity. Once you understand how the DOS manuals are organized, you will be able to refer to a certain page and find the specific answer to any question about the operating system more quickly than if you tried to look it up in a general-purpose book like this or the *DOS User's Guide*.

To give you a clearer overview of the operating system's functions, we have summarized some material about DOS and have listed the major DOS commands. Many of these commands you will use quite frequently. What is more important than memorizing DOS commands, at least for the moment, is understanding the kind of things the operating system does.



## Summary Of DOS 2.10 Commands

---

<i>Command</i>	<i>Action</i>
CLS	Clears display screen
COMP	Compares two files
COPY	Copies one or more files
DATE	Allows date to be set or displayed
DIR	Displays names, size, and time of creation of files on diskette
DISKCOMP	Compares the contents of two diskettes
DISKCOPY	Copies the content of one diskette to another
ERASE	Deletes one or more files
DEL	Deletes one or more files
FORMAT	Prepares a new diskette for use
RENAME	Changes the name of a file
TIME	Allows time to be set or displayed

Let's get more familiar with DOS and the IBM PCjr, by taking the operating system for a spin, with you doing some hands-on work. Take this book, along with your cartridges and diskettes, over to your computer work area. Prop up the book in a convenient location, and you're ready to go.

## Loading The Operating System

For starters, get your DOS diskette out and insert it into the disk drive. You will find two diskettes supplied in the DOS manual: one labeled DOS and the other labeled DOS Supplemental Programs. Be sure you use the one labeled DOS. Always insert the diskette with the label side up and to the right. Be sure to close the door of the disk drive after you have inserted the diskette, or the PCjr will not work. Turn on the system, piece by piece, if you don't have a power bar that switches on all components at once. First turn on the printer, then the monitor, and finally the computer, in that

order. The switch for the computer is located at the rear of the right-hand side of the system unit.

You will hear the whir of the disk drive as the computer automatically reads one of the operating system programs into memory. When it has finished, the computer will ask you for the date. Type in today's date as a sequence of two-digit numbers. For example, June 6, 1984, would be typed:

**06-06-84**

Then press the [ENTER] key.

DOS is sticky about the form of entries, so be sure to enter the date as shown. Slashes can be used instead of hyphens to separate the month, day, and year (for example, 06/06/84 is fine). If you make a mistake in entering any data in a DOS command, just hit the Backspace [←] key and type the material over.

Next, DOS will ask you for the time, so look at your watch and type in the time (in hours and minutes, separated by a colon). For example, 9:30 a.m. would be typed:

**09:30**

DOS will accept seconds, but they are optional. Should you feel compulsive enough to enter a value for seconds, set off the seconds with a second colon. For instance, to enter 11:42 a.m. and 10 seconds, just type:

**11:42:10**

Again, press [ENTER] to finish entering the command.

The computer operates on a 24-hour clock. That means you must add 12 to any p.m. time value. For 8:14 p.m., therefore, you would enter the hours as 8 plus 12, or 20. In other words, the time entry would be:

**20:14**

If you have an urge to start your computer at midnight, you would enter the time as:

**00:00**



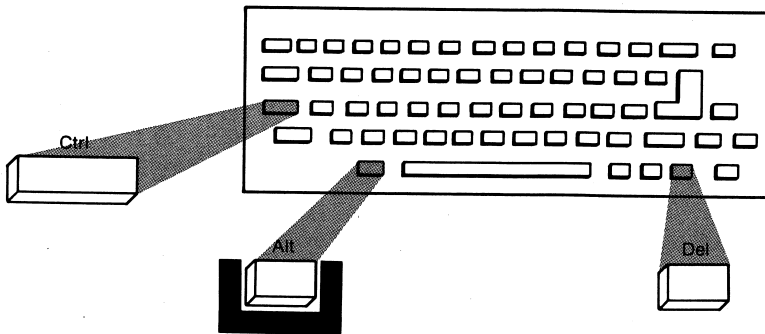
Now that you've set the computer's calendar and clock, they will keep accurate date and time until you turn off the machine. (Larger IBM personal computers have printed circuit boards with built-in calendars and clocks that automatically enter the correct date and time without any effort on your part. This feature probably will be a future option for the PCjr.)

If you don't want to type in the date and time, you can just press [ENTER] twice, without typing anything, and the computer will assume it is 12:00 midnight on January 1, 1980. This short cut has a drawback, though. DOS uses the date and time to indicate the last time you updated a program or file. If you did not specify the date, it will appear as if everything was created or revised on January 1, 1980. Later you will see that often it is important to know when you made the last change or revision to a program or data, so as to avoid the possibility of working with old data or an old program that still had rough spots.

The exercise you have just performed has a special name in the computer world. It is called "booting the system." The term came from the idea that the computer literally was pulling itself up by its own bootstraps by loading a program (the operating system) that still has to read another program into memory before any useful work can be done.

When you did the exercise, you activated the operating system when you turned on the computer's power. This is called a "cold boot," because the machine was "cold" (off) when the operating system was called up. During a boot, a portion of the operating system called COMMAND.COM is loaded into the memory of the PCjr. This part of DOS 2.10 actually performs a few basic functions, but its main responsibility is managing other programs and files and loading them into memory as they are needed.

The computer doesn't have to be turned off before the operating system can be started. If the computer is already on, you can perform a "warm boot" at any time, simply by pressing three special keys: [CTRL], [ALT], and [DEL] simultaneously (Figure 5-2). Doing this automatically terminates any program you are working on and reloads COMMAND.COM into the memory. In order for a warm boot to work, either the DOS diskette or a diskette that has a copy of the COMMAND.COM program must be in the disk drive when the three keys are pressed. If a diskette other than one of these two is in the disk drive, you will get an error message asking you to



**Figure 5-2** Pressing [CTRL], [ALT], and [DEL] keys will restart the DOS operating system. This procedure is known as a “warm boot.”

place the proper diskette in the drive so the boot can be carried out. If no diskette is in the disk drive when a warm boot is performed, cassette BASIC will be loaded into memory instead of DOS.

Warm boots are used when a program gets a little out of hand and you think it would be better to restart the whole process from the beginning. You may also find it necessary to re-boot DOS when you switch from one program to another. Warm boots come in handy because they're a lot faster to do than turning off the PCjr and then starting it up again later. Doing a warm boot is better for the equipment as well, because the process of turning on and off is always stressful to computers.

At first, you may find pressing all three keys at once a little awkward, but once you get the hang of it you'll have no trouble at all. And there is a good reason for this cumbersome design. This three-key combination was conceived as a precaution against booting by mistake—because once you boot, everything you were working on at the time is lost forever. It's quite unlikely that you would accidentally hit all three keys at once—hence the three-key code.

## Diskette Directories

Now you are ready to do some serious talking to the operating system. Basically, when you deal with the operating system, you have a very structured conversation. The operating system always signals you when it is your turn to talk. This signal,

A>

is a *prompt*. (Any symbol that the computer types to indicate that it is waiting for you is known as a prompt.) That is, the operating system is "prompting" you for a reply. Whenever the screen displays the "greater than" symbol

>

you know that DOS is waiting for your next command. So oblige DOS and type:

### DIR

then press [ENTER].

What happened? (If nothing happened or if you got an error message, something is wrong. Check to see that the monitor is turned on and that the door to the disk drive is closed.) Typing "DIR" asks DOS to display a "directory" of the diskette that is in the disk drive. (By the way, you could have typed "dir" instead, because DOS treats upper- and lower-case letters as identical.)

By way of this directory, DOS tells you the names of all the various programs on the DOS diskette (Figure 5-3). These twenty-three programs make up the operating system known as IBM PCjr DOS 2.10. The DIR command also displays the size of a program, in bytes, and the date and time a file or program was created or revised.

As part of this exercise, we will conduct a computerized scavenger hunt. The following is a list of items you can find from the directory listing of your DOS diskette:

How many bytes are free?

What is the three-letter code that follows the program ANSI?

Which is the largest program (the most bytes)?

What program precedes the program ASSIGN?

How many bytes does the program FORMAT occupy?

At what date and time was the DISKCOPY program placed on the diskette?

You can get all this information the first time you run through the directory. For some types of display monitors, you may need to use

A>dir

Volume in drive A has no label

Directory of A:\

COMMAND	COM	17792	10-20-83	12:00p
ANSI	SYS	1664	10-20-83	12:00p
FORMAT	COM	6912	10-20-83	12:00p
CHKDSK	COM	6400	10-20-83	12:00p
SYS	COM	1680	10-20-83	12:00p
DISKCOPY	COM	2576	10-20-83	12:00p
DISKCOMP	COM	2188	10-20-83	12:00p
COMP	COM	2534	10-20-83	12:00p
EDLIN	COM	4608	10-20-83	12:00p
MODE	COM	3139	10-20-83	12:00p
FDISK	COM	6369	10-20-83	12:00p
BACKUP	COM	3687	10-20-83	12:00p
RESTORE	COM	4003	10-20-83	12:00p
PRINT	COM	4608	10-20-83	12:00p
RECOVER	COM	2304	10-20-83	12:00p
ASSIGN	COM	896	10-20-83	12:00p
TREE	COM	1513	10-20-83	12:00p
GRAPHICS	COM	789	10-20-83	12:00p
SORT	EXE	1408	10-20-83	12:00p
FIND	EXE	5888	10-20-83	12:00p
MORE	COM	384	10-20-83	12:00p
BASIC	COM	16256	10-20-83	12:00p
BASICA	COM	26112	10-20-83	12:00p
23 File(s)		28672 bytes free		

**Figure 5-3** Directory of all the programs on the IBM PCjr DOS 2.10 diskette.

the Pause combination ([FN] plus [Q]) to temporarily halt the directory before it runs off the end of the screen, never to return. After you've used the Pause combination, you can reverse it simply by pressing the [SPACEBAR] or any other key.

The number of bytes on the diskette used by a program is displayed in the third column of the directory printout. Write down all your answers and save them. We will need some of them later.

## Formatting A Diskette

You're going to be making extended use of your disk drive in your encounters with the computer, so now is as good a time as any to become more familiar with this device. Before a diskette can be used for the first time, it must be *formatted*, or prepared by the computer for use. In this operation the computer checks the quality of the diskette and establishes the number and location of tracks and sectors. This procedure is quite simple and takes only a few moments.

Right now, if you have been following along with our exercises, the DOS 2.10 diskette should be in the disk drive with the DOS prompt A> showing. If it is not, then warm boot the system. (Remember, that means putting the DOS diskette in the drive and pressing the [CTRL], [ALT], and [DEL] keys all at the same time. Then enter the date and time and get the DOS prompt.) Issue the formatting command by typing:

### FORMAT

Press [ENTER]. Listen for the whirl of the disk drive as the computer reads the formatting program into memory. When the computer has finished, you will get a message such as the one shown below:

```
Insert new diskette for drive A:  
Strike any key when ready
```

Wait until the red power light turns off. Then remove the DOS diskette from the drive and put it safely out of the way. Next, insert a new diskette in the drive. Be sure there is no data or programs on the diskette that you want formatted, because formatting will erase the diskette, destroying any data that may be on it.

Once you have the new diskette in the drive, press any key (normally the [SPACEBAR]) to start the formatting process. DOS will respond with the following display:

```
Formatting . . .
```

and after a minute or so, it will show:

```
Formatting . . . Format complete
362496 bytes total disk space
362496 bytes available on disk
Format another (Y/N)?
```

Answer:

Y

(for yes) and format one more new diskette. You will be using it later in the chapter. As soon as you format a diskette, put a blank label from your diskette box on the diskette to distinguish it from an unformatted one.

Congratulations! You just got through the first phase of working with your IBM PCjr operating system and disk drive, and you have your first diskettes ready for use. Most of the exercises will be no more difficult to do than the formatting process. If you handled this without too much difficulty, you should have no trouble with the rest. Because formatting is such a common task, we have summarized the entire process in a step-by-step checklist.



## Disk Drive Checklist

1. Open disk drive door by lifting the lever.
2. Gently remove any diskette currently in drive and replace it in its cover.
3. Insert new diskette as follows:
  - a. label facing up
  - b. label on the right-hand side
  - c. label nearest to the front

The square write/protect notch and the small circular hole on the diskette should be to the left. **Warning:** Do not attempt to insert or remove a diskette if the red light is lit.

4. Close the disk drive lever gently but securely so that it snaps shut audibly.



## ***Diskette Formatting Procedure***

---

1. Insert the DOS diskette into the disk drive.
2. Turn on the power to the computer (if the power is already on, simultaneously depress the [CTRL], [ALT], and [DEL] keys).
3. Enter the day and date (for example, 01-15-84).
4. Enter the time, using 24-hour time (for instance, 11:23, or 14:57 for 2:57 p.m.).
5. After the A> appears on the screen, type:

### **FORMAT**

and press [ENTER].

The computer should respond:

Insert diskette for drive A:  
Strike any key when ready

6. Insert the new diskette (the one to be formatted) in the disk drive, with the label facing up in the right-hand corner nearest you. Close the door securely. **Warning:** *Be sure to remove the DOS diskette and replace it with the new, previously unused diskette before proceeding.*
7. Depress the [SPACEBAR] (or any other key).
8. When the computer notifies you that it has completed formatting that diskette, it will ask if you wish to format another. If you do, type:

**Y**

(for yes) and press [ENTER], then go back to step 6.

9. If not, type:

**N**

(for no) and press [ENTER].

10. Remove the formatted diskette and affix a blank label to it.

## Diskette Handling

Diskettes are useful and convenient storage devices, but they require tender loving care. The basic principle of data storage on a diskette is magnetization, just like with reel-to-reel or cassette recording tape. Diskettes must be protected from light, heat, cold, magnetic fields, bending, coffee, and children's fingers. A secure, clean work area, as we discussed previously, is essential to successful computing.

If you follow the do's and don'ts in our checklist, you can save yourself a lot of grief in handling your diskettes. For any of you who are teachers, chalk dust and diskettes definitely don't mix. Keep diskettes clear of chalkboards, or you can quickly ruin both the diskette and the drive.



### ***Diskette Do's And Don'ts Checklist***

---

1. *Do* keep your diskette in its protective jacket whenever it is not in use.
2. *Do* affix a label to each diskette as soon as you format it, and write the diskette name and the date on the label as soon as you record any programs or data on it.
3. *Don't* use a ball-point pen or pencil to write on the label; use a felt-tip pen instead.
4. *Don't* touch the exposed diskette surface.



5. *Don't* expose the diskette to magnetic fields. The system unit, the monitor, and power cords all generate magnetic fields. Keep diskettes away from these items. **Warning:** *Your stereo speakers contain powerful magnets; never place a diskette on or near a speaker of any kind.*
6. *Don't* use paper clips or clasps of any kind to hold your diskettes.
7. *Do* store your diskettes upright—not stacked like pancakes—in a secure, crushproof container.
8. *Do* protect your diskettes from direct sunlight, and keep them stored at room temperature (between 60 and 80 degrees Fahrenheit).
9. *Don't* eat, drink, or smoke anywhere near a diskette.
10. *Don't* leave loose diskettes in desk drawers. That's a sure way to damage them, even if they are enclosed in their protective jackets. Desk drawers are dirty and full of pointed objects.

Most probably you will mail a diskette to a friend or for business purposes. When you do, use a special diskette mailer and label the package "Contains Magnetic Media." Diskette mailers are available at office supply and computer stores. Such mailers are manufactured by both Sentinel and Dennison and are well worth the small extra cost to protect valuable data and programs. We have mailed a number of diskettes and can attest to the fact that unless precautions are taken, there is a good chance the diskette will be damaged.

## **Write Protection: Safeguarding Information On Disks**

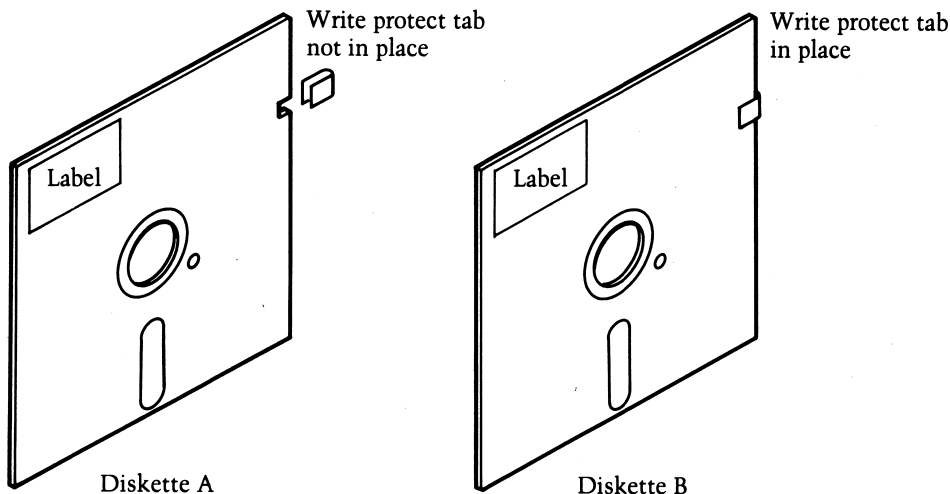
It's extremely important that you protect data and programs stored on a diskette from being inadvertently destroyed by an attempt to format or to copy data onto that diskette. The write/protect notch on your diskette offers this protection. When this notch is covered by the special tab supplied with the diskettes, the computer can only read information from that diskette. In a way covering the write/protect notch converts your diskette into read-only storage.

If, after a time, you want to record new information on the diskette, merely remove the tab and use the diskette in the normal fashion.

Many program diskettes are supplied without a write/protect notch. You cannot store any information on these diskettes. This safeguard prevents any accidental loss of the programs. If you look at your DOS diskette, for instance, you will see that it has no write/protect notch. This built-in feature is designed to safeguard the valuable information on that diskette from loss at the hands of a novice or absentminded computer user (Figure 5-4).

### Copying A Diskette

Next, we will practice copying some files from one diskette to another. This is an important function of the operating system. Whenever you buy a new program, you should make a backup copy as soon as possible. That way, even if something unexpected should happen (and you lose part or all of the contents), you will always have a spare copy. In fact, you should use the backup copy as your



**Figure 5-4** Diskette A's write/protect notch is uncovered. When the notch is uncovered, new information can be recorded on the diskette. Diskette B's write/protect notch is covered, which means that information can only be read from the diskette, not recorded on it.

working program diskette and keep the original safely locked away. Accordingly, the first thing we will do is make a backup copy of the DOS 2.10 operating system diskette—and store the original out of harm's way.

You just formatted two diskettes. Now get ready to use them. If you didn't format any diskettes (or if you already have used them for some other task), go back to the section on the `FORMAT` command and format two new diskettes for this exercise.

Place a label on the upper right-hand corner of one of the diskettes, and with a felt-tip pen gently write "DOS 2.10 backup" and the date. One reason for keeping track of the date on which the copy is made is to be sure you are using the most recent copy of the data or program. Another reason is quality control. If something is wrong with one or more copies made on a given day, chances are there may be a problem with other copies made that date. (Even a speck of dirt or a slight voltage irregularity can cause a series of poor copies.)

Now let's make a copy of the operating system onto this blank, formatted diskette. Later, you can use this backup copy for all future tasks or whenever we say to use your DOS diskette. So that the original and the copy can be distinguished, each is known by a different name. The diskette containing the information to be reproduced is known as the *source diskette*; the new one you create is called the *target diskette*.

Place your original factory version of DOS in the disk drive. The screen should show the DOS prompt,

```
A>
```

and the cursor should be flashing merrily right alongside. If that's not what you've got, try the cold boot procedure. If that doesn't do the trick, turn off your computer and go back through the steps we described in "Loading the Operating System."

All set? Good. Let's begin.

After the DOS prompt, type:

```
DISKCOPY
```

then press `[ENTER]`.

DOS should respond with:

Insert source diskette in drive A:  
Strike any key when ready

Because the DOS diskette is the one you are interested in copying, you can simply touch the [SPACEBAR] (or any key on the keyboard). If you wanted to copy a different diskette, you would have exchanged that diskette for the DOS diskette, closed the disk drive door, and then tapped the [SPACEBAR] or any other key on the keyboard.

Now, watch your PC*jr*. Listen to the whir as the disk drive spins your DOS diskette and see the red power light on the disk drive, which indicates that the disk drive is reading a segment of the diskette into its internal memory. *Warning!* Don't pull out the diskette or turn off the computer while the red power light is on or you will lose the data and possibly damage the diskette or drive.

When the internal storage area has been filled, you will get the following message:

Copying 9 sectors per track, 1 side(s)  
Insert target diskette in drive A:  
Strike any key when ready

Take your previously formatted diskette, exchange it for the DOS original, and tap the [SPACEBAR]. The drive again will whir and the red light will come on, this time indicating that the computer is copying the contents of its internal memory (that is, the contents of the DOS diskette) onto the new diskette.

You will have to repeat this procedure of swapping diskettes three times. When the process is complete, DOS will display the message:

Copy complete  
Copy another (Y/N)?

For now, just enter

**N**

(for no). You can make a copy of the DOS Supplementary Programs diskette with another formatted diskette a little later. Do not remove your new copy of DOS 2.10. We'll be using it in a moment.

## Verifying The Copy Process

Generally, the disk copy program supplied with DOS is accurate, but once in a while it can lose characters and even blocks of data. And that's just not good enough if you are copying a program or if your data must be correct down to the last decimal point. What do you do when such a level of accuracy is required? The only way to be sure the copy is identical to the original is to compare one with the other on a detailed, byte-by-byte basis.

We can compare the directory listing for the copy of DOS with the results we got from the scavenger hunt. You should have the copy you just made of DOS 2.10 in the disk drive, with the prompt showing on your display. Now type:

**DIR**

Using your scavenger hunt list, compare the items you wrote down with the ones now being displayed on your screen. They should match exactly. However, this cross-check is only a crude way to verify the copy, because even if the directory is perfect, parts of the diskette could have been copied incorrectly. Although this possibility may seem unlikely, it has happened. We have seen incorrect information on diskette copies, even though their directories were accurate. Particularly, this happens with copies made of commercial software.

Of course, you can always verify your copy of the program by putting it to the test. "The proof of the pudding is in the tasting," goes the old cliché. Testing, however, is time consuming and difficult. There is a much better approach. The DISKCOMP program supplied with DOS provides such an assurance of accuracy.

Let's verify the results of the copying process by using the DISKCOMP command. At this point you should have the copy of DOS in the disk drive, and the DOS prompt should be displayed. (If not, follow the warm boot procedure and obtain the prompt.)

Type:

### **DISKCOMP**

and press [ENTER]. DOS will respond with:

Insert first diskette in drive A:  
Strike any key when ready

You probably have noticed that DOS is a little inconsistent about the names it gives to the original diskette in a copy or compare process. "Source" and "first" diskette mean exactly the same thing to DOS, as you have no doubt deduced.

Because you are going to compare the original DOS with your copy, and the copy is already in the drive, just press [ENTER]. After the usual whirring, DOS will ask you to:

Insert second diskette in drive A:  
Strike any key when ready

Insert the original DOS diskette and press the [SPACEBAR].

You will have to repeat this process several times. If the comparison is successful, you will see the following message:

Diskettes compare ok  
Compare more diskettes (Y/N)?

Because the copy of DOS is identical to the original, we'll use it as our working copy from now on. Place a write/protect tab on the copy and store the original in a safe place. In the unlikely event that the copy is not an exact replica of the original, the DISKCOMP program will indicate the location of the errors.

Use this new DOS to make a copy of the DOS Supplementary Programs diskette. When the computer has finished copying your DOS Supplementary Programs diskette, it will inform you and will ask you again if you wish to copy another diskette. Type:

**N**

for no.

Now, the diskette in the disk drive will be your new copy of IBM DOS Supplementary Programs. The response of DOS may come as a surprise to you, but the message displayed will be:

```
Insert COMMAND.COM disk in drive A:  
Strike any key when ready
```

What actually happened is this: To speed up the copying process, DOS used some of the memory space allocated to the COMMAND program. Now DOS is asking you to insert a diskette that has a copy of COMMAND so that the entire program can be restored. When you get this message, just place the DOS 2.10 diskette in the drive and press the [SPACEBAR]. You should now have the old familiar DOS prompt,

```
A>
```

on your display.

Don't forget to place a write/protect tab on the new DOS Supplementary Programs diskette, run a DISKCOMP to verify the copying process, and put the original in a safe location.

## Buying A Second Disk Drive

Now that you've mastered your first diskette manipulations, we're going to introduce some further topics. As you went through these copy, compare, and disk formatting exercises, you may have come to appreciate how tedious it is to switch back and forth from one diskette to another. In point of fact, it probably took more of your time to do all that switching than it took the computer to do all that copying. To avoid this thankless job, most computer owners choose to purchase a second disk drive as one of their first options.

Having a second disk drive allows you to place both diskettes (the target and the source diskettes, to use DOS's definition) directly into the computer at the beginning. The computer then automatically switches back and forth between them as needed, freeing you from the necessity of swapping diskettes manually. Along with making things easier for you, a second disk drive also speeds up the computer. Furthermore, if you use any large program package

that performs many different functions (like some of the word processing or data base management packages that are available), you probably will need the additional capacity of a second drive to avoid running out of storage space. Otherwise, you will spend a great deal of time swapping diskettes.

At the present time, however, the *PCjr* doesn't offer the option of a second drive, so you can only have a single disk drive. But you can bet your bottom dollar there are several groups of engineers out there working day and night to be the first to come out with an add-on drive for the computer. Chapter 13 covers the anticipated developments with the *PCjr*, but for now don't worry. You won't miss the announcement and advertisements for this valuable addition to the *PCjr* when it bursts on the market.

## The "Phantom" Disk Drive

Until you purchase your second disk drive, though, what can you do to make your diskette handling simpler? Your *PCjr* responds to instructions for a second disk drive even when there is only one drive on the system. And DOS labels drives alphabetically, so your drive is drive A. In DOS commands the drive is indicated with the drive letter followed by a colon, as in the command

FORMAT A:

The second, "phantom," drive is named drive B and is indicated in DOS commands as B:

There are several advantages to using the formal phantom drive notation. For starters, the notation of drive A: or drive B: is the same whether you have a phantom drive or a real drive. So, when you have mastered using the phantom disk drive and then purchase a second drive, you will be ready to use it without needing to learn new methods to do the same things.

Also, the phantom disk drive gives the *PCjr* a better way to keep track of which diskette is which. We will see this best in the next section on the Checkdisk command. Some diskette commands could not be executed on the *PCjr* if it were not for the phantom disk drive.

All the commands we have covered can be performed using the phantom drive configuration. To format a diskette, you type



the standard FORMAT command as well as specifying the disk drive name (either A: or B:). If you do not specify a disk drive name, DOS will assume you always mean drive A, the actual disk drive, unless you have told it otherwise. So the new command might look like this:

#### **FORMAT B:**

The message DOS returns will be similar to what you got when you did not specify a drive name, but it will be changed to reflect the new drive name:

```
Insert new diskette for drive B:  
Strike any key when ready
```

The same rules apply for DISKCOPY and DISKCOMP. When these two commands are used with phantom drive specifications, the first (or source) diskette will be referred to as the diskette for drive A, and the second (or target) diskette will be referred to as the diskette for drive B. But since the PCjr has only one disk drive, you will be directed to insert the target diskette in the actual drive A.

Specifically, the two commands would be written:

#### **DISKCOPY A: B:**

and

#### **DISKCOMP A: B:**

This is exactly how the same commands would be written if you had two disk drives and one diskette was actually in drive A and the other in drive B.

## **CHKDSK Command**

The CHKDSK (Checkdisk) command is one of the basic commands available with DOS to verify the formatting process and to determine the capacity of a diskette. Use it anytime you have any doubts about the accuracy of a diskette or if you want to verify the diskette. It's a good idea to take this precaution before you make a backup

copy of an important program or data. As you use CHKDSK, you also will be seeing the usefulness of the phantom disk drive concept. Let's try the command on the copy of DOS.

Obtain the DOS prompt and place your copy of DOS in the disk drive. Then type:

### **CHKDSK**

and press [ENTER].

This will cause the computer to check your diskette to be certain it is correctly formatted. In addition to checking for formatting errors, the CHKDSK command also determines the amount of storage area remaining on the diskette and in the memory of the computer. With this feature, you can tell if there is enough space on the diskette for any additional files. If you are creating a large file or working with a lengthy program, you don't want to overrun the room on the diskette and you must be sure there is enough space in memory.

For the DOS diskette the CHKDSK command should display:

```
179712 bytes total disk space
22528 bytes in 3 hidden files
128512 bytes in 23 user files
28672 bytes available on disk
```

```
114688 bytes of total memory
900000 bytes free
```

By the way, hidden files are part of the operating system. They occupy space on the diskette, but you have no access to them and they never appear on the directory listing.

We're going to use the phantom disk drive capability to execute the CHKDSK command on another formatted diskette you will prepare.

Insert the DOS diskette, obtain the prompt, and type:

### **CHKDSK B:**

Then press [ENTER].

When the computer responds, remove the DOS diskette, insert the diskette you want to check, and press [ENTER] to continue.

Notice how using the phantom drive allowed you to perform a function requiring two diskettes at once (DOS for the CHKDSK command and the second diskette to be checked). Obviously, having two drives would be easier, but the phantom drive capability is the next best thing.

## External And Internal Commands

DOS has two types of commands: external and internal. The commands that ask you to insert the DOS diskette are called *external commands*. For instance, any command to format, copy, compare, or check a diskette is an external command because it requires that you insert the DOS diskette. *Internal commands*, on the other hand, stay in the computer's memory once the DOS system is loaded.

## Starting And Stopping DOS Commands

By now it is clear that you must press the [ENTER] key to start all the DOS commands. Occasionally, you will wish to stop a command before it is completed. To stop any DOS command, press the green [FN] key and the [B] (for "break") key. Once the DOS prompt appears, you can issue a new command.

## Recap

If the exercises we've just done were your first experience with diskettes and operating systems, you may be somewhat confused at this point. To set things straight, let's recap the whole process.

First, we examined the contents of a diskette by using the DIR command. This command showed a listing of the directory of the diskette. It also displayed the size (in bytes) of each file and the date and time the file was created.

We then used the FORMAT command to prepare diskettes for use on the PCjr.

Next we used the DISKCOPY command to copy the entire contents of one diskette onto another.

Then we verified that the copying process was correct by using the DISKCOMP command, which compared the information contained on the two diskettes.

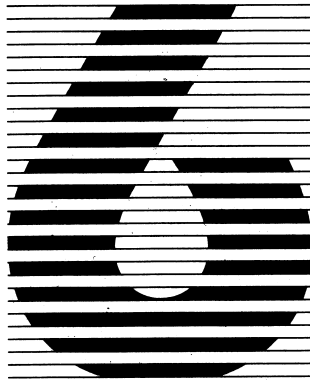
Finally, we verified that a diskette was properly formatted, and we received a readout of the remaining storage space on the diskette by using the CHKDSK command.

As this book was being prepared, we frequently made copies of chapters and forwarded them to the publisher for review. We used all the commands we have discussed in this chapter. In so doing, we caught several problems and thereby prevented difficulties with the use of these checks and comparisons. If you plan to do any serious computing, you will appreciate the assistance these commands can offer. They are not difficult to learn, so take the time to get to know them, and take advantage of their power. The DOS 2.10 Operating System is one of the best buys in the entire computer marketplace.

### Further Reading

Because DOS 2.10 is brand new, not much literature is available about it. However, DOS 2.10 is similar to the previous version designed for the IBM PC and XT computers, therefore some material on DOS 2.00 (such as the IBM manual, *Learning to Use DOS 2.00*) may be of some help. Just remember that the information was not intended for the PCjr.

Your best sources of additional information about DOS 2.10 are the *Disk Operating System User's Guide*, the *Disk Operating System Manual*, and *DOS Technical Reference Manual*. Keep your eyes open, though, because other books on the subject soon will be popping up on bookstore shelves.



## Let's Get Organized: Setting Up And Using Computer Files

The key to using your computer successfully is understanding how to instruct it to handle information. The secret to this is setting up and using files effectively.

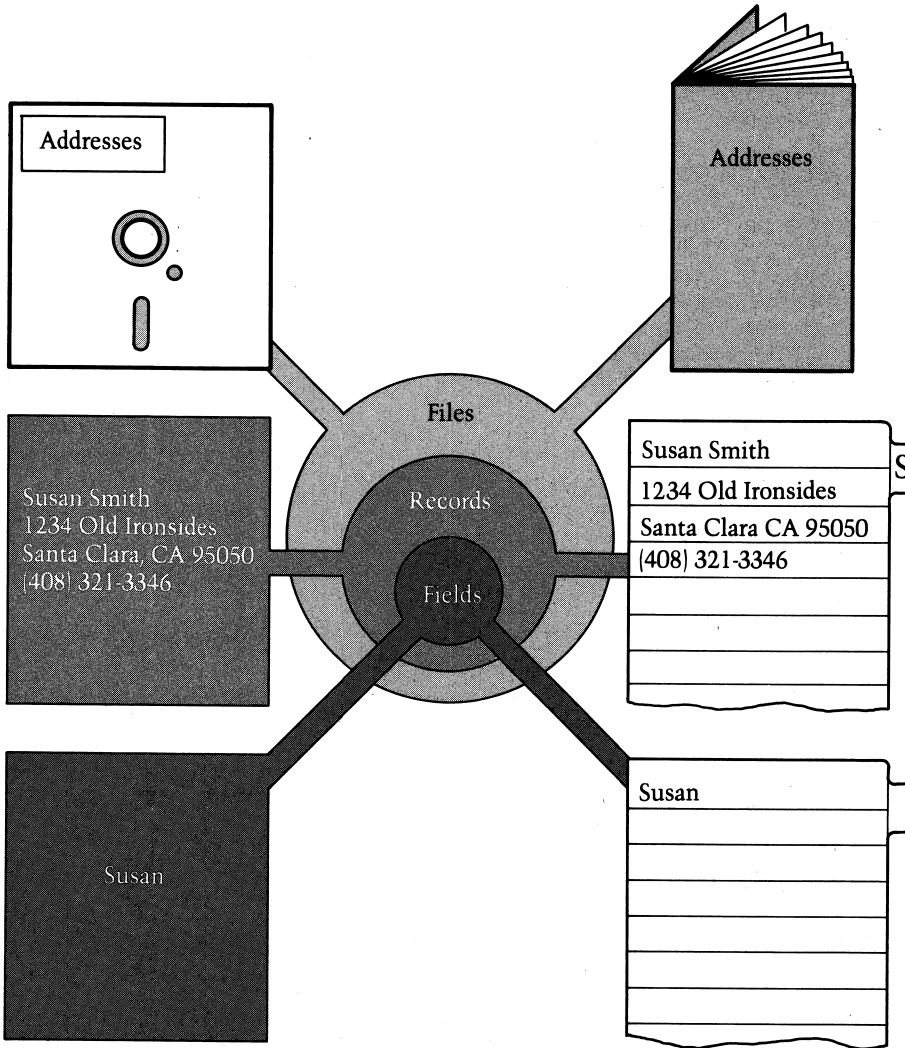
A *file* is a collection of related data. The idea of a file is simple, yet the concept is quite powerful. We make use of files all the time. When you organize your bills and place them in a folder, you are creating a file. When you record names and addresses in your address book, you are making another file. With the PCjr the concept of a file is a bit more formal although not really any different.

To a computer a file is a collection of related data on any storage media, such as a diskette, cassette, or cartridge. Each file has a name to identify it, and each is composed of individual records.

The *record* is the basic unit of information within the file. A single bill in that collection of bills you have can be considered a record. One of those names and addresses in your address book is another example of a record.

Every record within a file must be identifiable. It wouldn't do much good to write down addresses and phone numbers without names to go along with them. The computer needs the same kind of information. And so the records themselves are divided into fields (Figure 6-1). A *field* is the smallest unit of information within a file. It can be as small as a single character (the middle initial of a name) or a number (for instance, a phone number), or the field can contain longer descriptions (such as an entire address).

If you choose to make your fields lengthy, however, you will have to live with the fact that you cannot individually access a part



**Figure 6-1** Understanding the file structure of the PCjr. Information that you enter into the computer is organized by files, records, and fields. You use the same method of organization when you record names and addresses in an address book.

of a field. For example, if you set up a record with an address field that lumps together the house number and street name, you will not be able to access or process either the number or street name separately. The computer treats all the data within a field as a single unit. Keep this fact in mind when you set up your fields.

A file is extremely valuable in data processing for many reasons. It provides easy and rapid access to information. It can be updated or modified. And it can be processed, via computer programs, to produce reports and other types of output. The records in the file can be sorted in ascending or descending order, or by individual fields, to make them easier to use. Separate files can even be merged into one larger file to present information about a more comprehensive group. Data processing files are stable, compact, transportable, and extremely easy to reproduce, as you already have seen.

In the last chapter we talked about files, but we could not handle them on an individual basis. We had to treat every file as one unit, and that isn't always practical or convenient. With the help of the DOS commands, however, you can manipulate individual files. This allows you access to each file separately so that you can copy, compare, examine, create, erase, and manage these files. This capability is important because of the flexibility it offers in handling files.

Although normally you might not think of computer programs as files, the computer does. And DOS makes no distinctions between types of files, so program and data files both are treated in the same way. This saves you the time and trouble of learning a separate set of rules for each. In a BASIC program, for example, each statement represents a record in that file, and the line number used in the statement identifies that particular record.

There are no restrictions on the size of a file. You can even create a file from a single record containing only one field. Word processing files are examples of this single-unit file. A letter to a friend is not just a set of separate sentences, paragraphs, and even pages. The entire letter (or file) must be read as a single entity. Parts of the message probably wouldn't have any meaning out of context.

## **Filename And Extensions**

The name of the file is what identifies it as a unique collection of data. Therefore the name should have meaning and should suggest the contents of the file. What you name a file makes no difference to the computer, but you will find it much easier to keep track of things if your filenames ring a bell.

An aerospace company learned this lesson the hard way when one of its programmers made a million-dollar mistake. The programmer, who was told to run a test on program XXXXII14, got the "name" of the program mixed up and ran the test on program XXXII14. Needless to say, the program names now used by that organization have more meaning than those early combinations of Roman numerals and digits. Names like ADRESSES or CHAPTER4 are much better than strings of letters and numbers, because they are easier to recognize and remember.

No, we didn't misspell "addresses" or forget to space between "chapter" and "4." In DOS there are several restrictions on filenames. You can choose any name as long as it is no more than eight characters in length, and you are free to compose the filename of letters, numbers, or even some special symbols. But you cannot leave any spaces between the characters. Obviously, we couldn't spell addresses correctly and still conform to the size limitation, nor could we skip the space for Chapter 4. But both of the filenames we selected are close enough to prod our memories about the contents of those files, don't you agree? The list of valid and invalid filenames shown in the filename checklist provides more examples.



## *DOS Filename Checklist*

---

1. Filenames should suggest their contents to help you remember the type of information you have on the file.
2. Filenames can be from one to eight characters in length, but try to use at least three characters.
3. Filenames can contain the letters A through Z, the digits 0 through 9, and the special characters: \$ # & @ ! % ( ) - \_ { } ' ' ~ ^.
4. Filenames can contain either upper- or lower-case characters, because the computer treats both exactly the same way.
5. Starting filenames with a letter rather than a number is good practice, although it is not mandatory.



6. You cannot have the same filename for two different files on the same diskette.
7. You can use the same filename for files stored on different diskettes, but this is sure to confuse you, so we suggest you avoid the practice.
8. In DOS commands you can use the "wild cards" (\* and ?) when referring to filenames. The asterisk can be used as a substitute for any number of characters in a filename, and the question mark can be used as a substitute for any single character in that position. Be careful when you use these wild cards, because you can get some unexpected results, such as copying over new files with backups.



### *Examples Of Valid And Invalid Filenames*

<i>Valid</i>	<i>Invalid</i>
PCJR1-5	PCJR 1-5 (contains a blank)
LTTRHOME	LETTERHOME (contains too many characters)
ADDBOOK	ADD,BOOK (contains comma)
RECIPES	RECIPES:file (contains colon)
BILLS\$	BILLS\$*+ (contains invalid characters)
CHAP6	CHAP"6" (contains quotes)
HOMEWORK	HOMEWORK.ASSIGNMENTS (contains too many characters)
%APPLIED	% Applied (contains blank)

In addition to the eight-character name; a filename can include a three-character filename extension, which is separated from the filename by a period. Filename extensions often have specific meanings, like .BAK for backup copies of word processing files and .COM for command files. (Don't get hung up on the specific types of filename extensions; just be aware that they exist.)



## ***DOS Filename Extension Checklist***

---

1. The DOS filename extension is optional. If you choose to use it, make sure it adds meaning to the filename.
2. The filename extension begins with a period and consists of one to three characters, but we recommend that you consistently use three characters.
3. The extension can consist of the letters A through Z, the digits 0 through 9, and the special characters: \$ & # @ ! % ' ' { } ( ) - \_ / \. No other characters are valid.
4. An extension becomes part of the filename and must be included in all DOS commands.
5. Certain extensions have special meaning in DOS and should not be used for any other functions. These are:

---

<b><i>Extension</i></b>	<b><i>Meaning</i></b>
.ASM	ASSEMBLER program
.BAK	backup file
.BAS	BASIC program
.BAT	BATCH file
.COM	COMMAND file
.DAT	data file
.DOC	document file
.EXE	execute file
.LF	LOGO program
.LIB	LIBRARY program
.MSG	message file
.OBJ	OBJECT module
.TMP or .\$\$\$	temporary file
.TXT	text file



## ***Valid And Invalid Filenames With Extensions***

---

<b><i>Valid</i></b>	<b><i>Invalid</i></b>
BILL.MAY	BILL.JUNE (extension too long)
PROJECT.123	PROJECT.*** (invalid characters in extension)
STORY.BAK	STORY."BAK (invalid character in extension)
SALES.DAT	SALES.DATA (extension too long)

We have already seen how DOS can help you manipulate files, either individually or in groups, and how it can copy all the files from one diskette onto another. With the use of DOS, individual files can be copied, deleted, or renamed. Space can also be allocated on the diskette for enlarging files or for creating new ones.

As you did with your DOS diskette, make backup copies, on separate diskettes, of your key files to prevent a loss of data. It's no fun to learn from experience that it doesn't take much to destroy a file.

## **Preparing A Work Diskette**

Because one example is worth a thousand words, let's set up a work diskette, copy some files, and then create and edit a new file, which we'll name DEMO. Sounds like a lot of work, but it isn't really. That's where DOS comes in. It makes tasks like these easy and straightforward. When you really start to put your PCjr to work, you will be performing this sequence of tasks routinely, without a second thought.

First, place the DOS 2.10 disk in the drive and start the computer. It's time to format a fresh disk. You probably remember how we did this before, but this time we're adding a few new wrinkles, so pay attention.

To begin this exercise, after the DOS prompt, type:

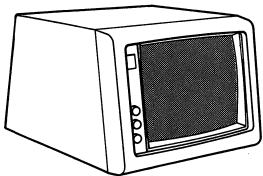
**FORMAT A:/S/V**

After the usual disk drive rumblings, the display will indicate that the formatting is complete and that the operating system is transferred to your diskette. This is the result of using the /S option in the FORMAT command. But right now, the computer is asking you for a "volume label." What is that?

By using the /V option in the FORMAT command, you can write, into the directory of the work diskette, the electronic equivalent of the paper label that's on the diskette's plastic jacket. Once you've accomplished this, the volume label will be the first item displayed every time the directory of the diskette is shown.

Enter EXAMPLES as the volume label for this diskette (Figure 6-2).

Because we put a copy of the system portion of the operating system on the newly formatted diskette (the /S option did this), the FORMAT command tells us we have 321,536 bytes of space remaining on the diskette and that 40,960 bytes were used to store the



```
A> format a:/s/v
Insert new diskette for drive A:
and strike any key when ready

Formatting . . . Format complete
System transferred

Volume label (11 characters, ENTER for none)? EXAMPLES

362496 bytes total disk space
 40960 bytes used by system
321536 bytes available on disk

Format another (Y/N)?n
A>
```

**Figure 6-2** Format display. This screen display shows that formatting is complete and that the operating system has been transferred to your diskette.

system. If we had used the `FORMAT` command without the `/S` option, we would have 362,496 bytes available. You can see that the request to add the system portion to the diskette cost us about 11 percent of the total space available.

Then why add the system to a diskette at all, when we already have a copy on the DOS diskette? The reason is convenience. Now, this new diskette can stand alone. That is, we can use it to cold or warm boot the system without inserting the DOS 2.10 diskette. This option saves us both time and effort. We don't have to put in the DOS 2.10 diskette, start the computer, remove the DOS 2.10 diskette, or insert the work diskette with our files. You pay the price for this convenience (in storage space), but you'll find it's worth it in the long run.

Now, let's find out exactly what that price means in terms of bytes lost from the work diskette. Obtain the DOS prompt by answering

**N**

to the question

Format another (Y/N)?

and get the directory of the work diskette by typing

**DIR**

DOS tells you that `EXAMPLES` is the volume in the drive and that `COMMAND.COM`, which contains 17,792 bytes, is the only file on the diskette. According to the `DIR` command, the price is 17,792 bytes.

Now run the `CHKDSK` command on the work diskette. This indicates that there are 22,528 bytes in three hidden files and 18,432 bytes in one user file on the diskette. How much space does the `COMMAND.COM` file actually consume? DOS is not consistent about the size of the `COMMAND` file as measured by the two different programs. The only thing you can count on is the amount of storage space available on the diskette. This figure is the same with both commands—321,536 bytes.

## Copy Command

The FORMAT command moves the COMMAND file from the DOS diskette to the work diskette. But that is the only file FORMAT can transfer. You'll have to use the COPY command to move files from one diskette to another or to make a second copy of a file on the same diskette.

Because one of the reasons for this exercise is to do some file editing, let's transfer EDLIN, the line editor supplied with DOS, to the work diskette.

Put the DOS 2.10 diskette back in the drive. (The computer should be showing the DOS prompt.) Type:

```
COPY EDLIN.COM B:EDLIN.COM
```

and press [ENTER] to execute the command. (Remember, you must press [ENTER] to activate any command. But don't worry if you occasionally forget. Pressing [ENTER] will become second nature after a short time.)

The COPY command normally requires *two* pieces of information: the name of the file to be copied and the name and diskette to be used for the copy. If you use filename extensions, be sure to use them when specifying files, or DOS will reply with a "file not found" message.

If you are making a copy of a file on the *same* diskette as the original, the copy must have a different name. The copy can have the same name as the original only if it is created on a *different* diskette.

In this case we'll be copying EDLIN onto a different diskette, so use the phantom drive B in the command. The COPY command will direct you to change diskettes and then will indicate successful operation by displaying:

```
1 file(s) copied
```

Now that our work diskette contains the COMMAND and EDLIN programs, it can stand alone. But before you start creating your own files, you have more to learn about the COPY command.

The COPY command offers a number of useful options. First, it saves you some typing. When you are copying a file to *another*

diskette and the name is the same on both, there is no need to retype the filename. The statements:

```
COPY EDLIN.COM B:EDLIN.COM
```

and

```
COPY EDLIN.COM B:
```

both will produce the same results.

## **Wild Card Filename Conventions: The "\*" And "?" Have Their Uses**

Before we go into more detail on the COPY command, let's go over some of the rules of the road used in driving around DOS. Formally, they are called the DOS *conventions*. The first of these conventions is the global filename characters.

Two special characters—the asterisk (\*) and the question mark (?)—are useful in specifying filenames and extensions in DOS commands. These two characters act as “wild cards,” in that they can represent any other characters used within the filename or extension. Because the asterisk is the more powerful of the two, let's start with an explanation of this workhorse and see how it can add more muscle to your use of DOS commands.

Whenever the asterisk is used in a DOS filename or extension, it represents *any* character or characters that can occupy that or *any remaining positions* in the name or extension. That means, as far as a DOS command is concerned, that E\*.\*, ED\*.\*, EDLIN\*.\*, and EDLIN.COM all are the same thing. If we use the asterisk, the same COPY command we performed before can be written as:

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

The asterisk indicates to DOS that it should copy all file names starting with ED. While this shortcut can save some typing, it can also cause problems—if you have several files beginning with ED.

Suppose you have files named ED, EDWARD.BAK, EDDIE, EDITH.BAS, and ED.COM, as well as EDLIN.COM—all stored on the same diskette. Then, specifying

**COPY ED\*. \* B:**

would cause DOS to copy *all* of these files onto diskette B. That's fine if it's what you want, but if you really intended to copy only file EDLIN.COM, you probably will be frustrated with the results.

Let's compare the question mark with the asterisk. When you use a question mark in a filename or extension, it tells DOS that any character can occupy that position. The question mark differs from the asterisk in that it requires a character to be present in the position.

For example, if your diskette contains the files ED, EDDIE, EDITH, and EDLIN, and you issue the DOS command:

**COPY ED??? B:**

DOS will copy EDDIE, EDITH, and EDLIN, but not ED, because there are only two characters in the name ED. However, if you issue the command:

**COPY ED\* B:**

all four of those files will be copied.

Because using the question mark requires just as many key strokes as entering the correct filename, why use it? Well, if you are as absentminded as we often are, you'll find the question mark to be a big help. It allows you to use a DOS command to access a file, even if you can't remember the entire name. If you know that the file you need is called either TEST7 or TEST8, the command

**COPY TEST? B:**

will make copies of both of these files (and possibly a few others). If you use the command

**COPY TEST\* B:**

you might get a lot more files from that crowded diskette.

Let's put some of this knowledge to use. Get the DOS prompt, insert the DOS 2.10 diskette, and execute the command



**COPY BA\*. \* B:**

Copy these files to the EXAMPLES diskette. Notice how many files were copied with only one command. This means quite a saving in time and typing over copying those files individually. In fact, the single command

**COPY \*. \* B:**

copies the entire contents of one diskette onto another. Note that using COPY \*. \* B: is somewhat slower than using the DISKCOPY command, which is the preferred way to copy an entire diskette.

## Verify Option

Although the COPY command usually is extremely reliable, there is always some possibility of an error in making copies. Therefore, when you must be absolutely sure your copy is correct, use the verify option. When you receive the DOS prompt, insert the DOS 2.10 diskette and execute the command

**COPY FORMAT.\* B:/V**

Everything might seem to be going along as usual, but if you had timed the copy process, you would have found that the command executed somewhat slower with the verify option in effect. Checking the transfer of files takes time, or, to quote an old adage with a new acronym—TINSTAAFL (There is no such thing as a free lunch).

## COMP Command: Comparing Files

If you are at all like the absentminded professor on the author team of this book, you occasionally will forget to use the verify option in the COPY command while copying an important file. DOS provides for this by supplying the COMP command. With this command individual files or groups of files can be compared and you can make sure your backup copy is identical to the original.

The COMP command is similar to the COPY command. To do a comparison, it needs the names of the files. DOS calls the

original file the "primary file" in this command. To compare the FORMAT file on the work diskette with the original on the DOS diskette, get the DOS prompt, insert the DOS diskette in the drive, type:

**COMP**

and press [ENTER]. DOS will display:

Enter primary file name

At this point you should insert the diskette containing the original file. Because that file is FORMAT on the DOS diskette, you need only type:

**A:FORMAT.COM**

and press [ENTER]. DOS will display:

Enter 2nd file name or drive id

Type:

**B:FORMAT.COM**

and DOS will perform the file comparison. (As you probably guessed, because the copy and the original files had the same name, you could have shortened your typing of the last response by merely entering B:.)

## **ERASE Command**

With all the COPY command practice we've had, our EXAMPLES diskette has become cluttered. The ERASE command can come to the rescue here. The ERASE command does not operate exactly as its name implies. What it actually does is delete the file from the directory listing so that the space used by that file can be allocated to a new file. Only when a new file actually is written onto that area is the data lost. However, once a file is "erased," you cannot access it by the normal DOS commands.

These details may seem more like a distinction than a difference, because once a file is erased, it may seem to be gone forever. However, we mention the details here for a good reason. No matter how careful you are, sometimes you may accidentally erase a file you really need. If you have been following our advice on backing up diskettes and files, the backup copy could save the day. But, if there is no backup, don't do anything else with the diskette. An expert programmer may still be able to retrieve the data. However, the moral of this story is: Back up important files! Of course, diskettes do fill up, and files become obsolete, so on with the ERASE command.

The main reason we had you copy the FORMAT file onto the EXAMPLES diskette was to erase it. We already have a copy of FORMAT on the DOS 2.10 diskette and a backup copy of that diskette as well, so erasing FORMAT from EXAMPLES is perfectly safe. Get the DOS prompt, insert the EXAMPLES diskette, and execute the command

### **ERASE FORMAT.COM**

The computer will respond with the DOS prompt when the command is finished. Get the directory of EXAMPLES to assure yourself that the ERASE command actually worked. (If you improperly entered the file extension in the ERASE command, you will get the reply:

File not found

Reenter the command with the proper file name and extension.)

With the ERASE command, like the COPY command, it is not necessary to insert the DOS diskette before you execute the command. The ERASE command program is contained in the COMMAND file that was transferred to EXAMPLES when the diskette originally was formatted. As we pointed out in Chapter 5, DOS commands contained in the COMMAND file are called internal commands, whereas the commands that require their own files and ask you to insert the DOS diskette are called external commands.

If you type the command

**ERASE \*.\***

you are asking DOS to erase all the files on the diskette. To save you from yourself, DOS will respond with:

Are you sure? (Y/N)

If that is what you want, fine. Type:

**Y**

Otherwise, N may prevent a disaster.

The word DEL (for delete) can also be used to activate the ERASE command. That is, typing:

**DEL FORMAT.COM**

or

**ERASE FORMAT.COM**

will achieve the same results.

## Renaming A File

DOS supplies a command that allows you to change the name of a file. Although you won't use this command nearly as often as some of the other commands we have discussed, you will still find it valuable.

Throughout the text we have been stressing the importance of backing up diskettes and files. Many applications programs and word processing programs automatically produce a backup copy of a file. The name given to this generated backup file normally has the extension .BAK. And, to safeguard the backup files, most of these programs do not allow you to work with a file that has an extension of .BAK. You must rename these files before you can use them.

In data processing it is common practice to keep a copy of the current and previous months' data. That way, if something happens, the status of the operation can be reconstructed. As each new month's data is added, the current month's information becomes the previous month's data and the file is simply renamed.

The RENAME command is easy to use. Just obtain the DOS prompt, insert the diskette containing the file to be renamed, and then type:

### **RENAME**

followed by the current filename and the new name. To give a file called CURRENT the new name of PAST, you would type:

### **RENAME CURRENT PAST**

Note that in the command, the old name is separated from the new name by a space.

## **Displaying The Contents Of A File**

DOS supplies the TYPE command to let you examine the contents of a file. What the TYPE command does is display the contents of the file on the screen. If your file is composed of ordinary text, it will be readable. However, some program files use special characters that make no sense in English.

When you want to use the TYPE command, get the DOS prompt, insert the diskette that contains the proper file, and enter the DOS command

### **TYPE**

followed by the filename. For example, to display the contents of FORMAT, you would type:

### **TYPE FORMAT.CON**

Don't expect to make much sense of the output, however. As you well know, the output will race across your screen. If you want to freeze the screen to read any part of your DOS display, press the [FN] key and then the [P] key (for Pause, as we discussed earlier in this book).

## Printing What You See And Type

Up to now all the displays have appeared on the monitor screen. To make a hard copy printout of what's on the screen, merely press [FN] and then [S] (PRTSC, or Print Screen) and the entire screen will be printed. Be sure your printer is on when you issue this command.

To print what you type as you type it, press [FN] and [E], for Echo. Now every time you press [ENTER] or every time a line of data is displayed on the screen, it will be "echoed" to the printer. To turn off the printing, merely press the same two keys again.

You may find it valuable to print the directory of a diskette and to store it with that diskette.

## Editors

EDLIN is the editor supplied with your DOS 2.10 package. In computer talk an *editor* is the name given to a program used to create, modify, and display files. Editors like EDLIN have been around for a long time. In fact, for twenty years, EDLIN was the only answer to the problems of editing files on a computer.

EDLIN is a line editor. That is, it builds files one line at a time. EDLIN commands allow you to insert, delete, copy, modify, and move lines of information within the file. Function and control keys on the keyboard let you make changes within a line.

Now the home computer user has other editing options, which we will discuss presently, but line editors are still popular. And you can learn a lot about the sophisticated word processing packages currently available by utilizing this helpful text editor.

By the way, there are two types of editors. In addition to line editors, like EDLIN, there are screen editors. The editor you used in Chapter 4 to edit BASIC statements was a screen editor. It allowed you to move about the screen using the cursor control keys.

Line editors are used primarily for preparing programs and numeric data files (files containing numbers as data) as input for programs, because they are naturally organized as a series of discrete lines. EDLIN is a little awkward to use for correspondence and reports, which are organized in a sentence/paragraph structure, rather than in separate lines.

Compared with some of the latest advances in computing software, EDLIN is more like an old jalopy than a new car. But, to take the auto analogy a step further, it still has plenty of mileage to offer. And since your computer is already equipped with EDLIN, why not take advantage of this valuable tool?

If you intend to do much word processing with your PCjr, you will find the screen editors supplied with word processing applications packages considerably easier to use than EDLIN. In fact, we'll devote a whole section (in Chapter 8) to screen editors and word processing on the PCjr.

## EDLIN Commands

Now that you can comfortably manipulate files with the DOS commands, you are ready to create a file of your own. The EDLIN editor, supplied by DOS, is the tool for this purpose.

Thus far you have done relatively little typing. The input required for DOS commands usually is simple and short, and the data needed from the keyboard to play the PCjr games is limited. The lines in the BASIC statements were not very long either. If you made any typing errors in any of the DOS command lines, you just used the Backspace [←] key to delete the characters to that point and you retyped the line. For most of the errors, the correction amounted to inserting or deleting a single character, but you had to retype the entire line nonetheless. For the short DOS or game command lines, this retyping was annoying, but bearable.

If, on the other hand, the lines are long and complex, retyping an entire line to correct a single error is just not acceptable. Imagine trying to write a scientific formula, draft a letter, or prepare a homework assignment if every time you found a typing error, you had to retype everything from that point on. You might never finish your job. That is why EDLIN was designed—to make the job of typing data, programs, and correspondence easier and more efficient.

To use EDLIN, you enter a filename. If this is the first time the name is being used, a new file is created. Then you instruct EDLIN—by the simple one-letter command "I"—that you want to *insert* data into the file. As you type each line, it is automatically assigned a number, beginning with 1. If you make a typing error, you can instruct EDLIN to leave the insert mode—by pressing the

Control [CTRL] and Break [SCROLL LOCK BREAK] keys simultaneously. Then you can correct the error or modify or delete the line by using any of the several EDLIN commands. If you wish, you can return to any of the lines for further editing by using the appropriate EDLIN command. After the typing is completed, you can tell EDLIN to save the file for later use.

If the file name is already in the directory when you call EDLIN, the file can be updated or changed with the appropriate EDLIN command. This procedure may appear complicated at first, but don't despair. A few examples will show you how to use EDLIN in practical situations.

### **The Q Command: Undoing What You've Done**

Using EDLIN, at least for a beginner, can be somewhat frustrating at first. But the designers of EDLIN have thought of that and have included what we call an "undo" command. If your editing session gets messed up and things are happening that don't appear in the book, just press the Control [CTRL] and [BREAK] keys. Then type:

**Q**

and press [ENTER]. This will allow you to abort the editing session and undo any changes made to your file during that session.

### **Starting An EDLIN Session**

To be sure everything is on an even footing, insert the EXAMPLES diskette into the drive and get the DOS prompt. Now examine the directory of the diskette by using the DIR command. The EDLIN file should be present. (If it is not, use the COPY command to get it back onto the EXAMPLES diskette.) The first file you create will be a demonstration of how EDLIN works, so let's call it DEMO.

With the DOS prompt present, type:

**EDLIN DEMO**

and press [ENTER]. The monitor screen will look like this:



```
A>EDLIN DEMO
New file
*
```

The asterisk is the EDLIN prompt. It has *two* meanings. When it is at the left edge of the screen (as it is in this illustration), the asterisk means that EDLIN is waiting for a mode command. You want to be in the insert mode so that you can enter data into the file, so type:

```
I
```

(A lower-case letter will work just as well for EDLIN as for DOS, but because upper-case letters are easier to see, we will use them in our examples.) Then, press [ENTER]. Remember, all EDLIN commands are just one letter long and, like the DOS commands, you must press the [ENTER] key to activate them.

The screen should look like this:

```
A>EDLIN DEMO
New file
*I
1:*
```

The 1:\* shows that you are at the first line of the file. Again, the asterisk is the prompt for EDLIN. The asterisk by itself indicates that EDLIN is waiting for a command. However, if the asterisk appears *after* a line number, it indicates that the designated line is the one currently being edited.

Now we are ready to enter some text into the file. Type:

**This is a demonstration of the use of EDLIN.**

and press [ENTER]. In this case the [ENTER] key tells EDLIN that the line of data is finished. EDLIN then shows you it is ready for a new line of data by displaying the next line number and the asterisk. The screen will look like this:

```
A>EDLIN DEMO
New file
*|
  1:*This is a demonstration of the use of EDLIN.
  2:*
```

The first line of text is now entered into DEMO, and EDLIN is waiting for the next line. (For now, if you make a typing error, just use the Backspace [←] to delete a character and retype.)

For the second line, type:

```
0123456 ABCDEFGHI abcdefghi.
```

The screen will look like this:

```
A>EDLIN DEMO
New file
*|
  1:*This is a demonstration of the use of EDLIN.
  2:*0123456 ABCDEFGHI abcdefghi.
  3:*
```

(Note, if you are using a forty-column display, this and other lines that are longer than forty columns will be split on your screen. The computer treats these lines as one unit, however.)

That's enough of typing lines. Let's save the file. Remember, pressing the Control [CTRL] and Break [SCROLL LOCK BREAK] keys at the same time tells EDLIN you want to change the command. Now, press the Control [CTRL] and Break [SCROLL LOCK BREAK] keys. The screen will look like this:

```
A>EDLIN DEMO
New file
*|
  1:*This is a demonstration of the use of EDLIN.
  2:*0123456 ABCDEFGHI abcdefghi.
  3:*C
  *
  _
```

Now type:

```
E
```

and press [ENTER] to *end* the EDLIN session and save the file. The screen will look like this:

```
A>EDLIN DEMO
New file
*I
  1:*This is a demonstration of the use of EDLIN.
  2:*0123456 ABCDEFGHI abcdefghi.
  3:*^C
*E
A>_
```

To verify that you actually have created DEMO, examine the directory of the diskette in drive A by typing:

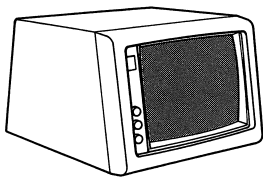
### DIR

You should find that DEMO exists and occupies 77 bytes. Good work, you have created your first text file. Figure 6-3 shows how the entire screen would look after you have completed this exercise.

EDLIN is an editor, but you really did not do any editing in this last example. Let's try some editing of the DEMO file now.

Return to EDLIN by typing:

### EDLIN DEMO



```
A> EDLIN DEMO
New file
*I
  1:*This is a demonstration of the use of EDLIN.
  2:*0123456 ABCDEFGHI abcdefghi.
  3:*^C

*E

A>
```

**Figure 6-3** EDLIN Example 1. Trace of screen during editing session.

and pressing [ENTER]. Because the DEMO file already exists, EDLIN will display the message:

```
End of input file
```

and then will display the asterisk to indicate that it is ready for a command.

Editing, at least in computing, is no more than adding lines, deleting lines, and modifying the contents of lines. You will be doing all three of these operations to the DEMO file.

For a starter, let's add a line at the beginning of DEMO. DEMO is a short file, and because you just typed it, the contents are clear in your mind. But, to get your bearings, let's follow the normal practice of listing the contents of a file before starting an editing session. The EDLIN command to list a file is "L." So, type:

```
L
```

EDLIN will respond with:

```
1:*This is a demonstration of the use of EDLIN.  
2: 0123456 ABCDEFGHI abcdefghi.
```

```
*
```

Notice the asterisk after the 1 in the first line. This symbol shows you that you will be able to edit line 1 when EDLIN is first called. Next, note the asterisk at the left of the screen, after the last line. This asterisk tells you that EDLIN is waiting for another command. Oblige EDLIN by issuing the command

```
11
```

which tells EDLIN to insert lines, starting *before* what was formerly line 1.

All EDLIN commands dealing with line manipulation can have line numbers preceding them. In computerese these line numbers are called *parameters*. That is, they give the commands a specific, rather than a general, meaning. In this case the "parameter" 1 tells

the insert command to start inserting material before line 1. EDLIN will display:

```
1:*
```

Now type:

**This should be the first of three lines.**

and press [ENTER]. Then list the file by leaving the insert mode (that is, pressing Control [CTRL] and Break [SCROLL LOCK BREAK] at the same time) and then listing the file with the L command. EDLIN will display:

```
1: This should be the first of three lines.  
2:*This is a demonstration of the use of EDLIN.  
3: 0123456 ABCDEFGHI abcdefghi.  
*
```

Mission accomplished. The line was inserted properly. But it appears EDLIN has been playing tricks. The original lines are now renumbered, one digit higher. This renumbering is standard procedure. When lines are inserted or deleted, EDLIN renumbers the entire file. The line numbers in EDLIN are not like the line numbers you used in BASIC. In EDLIN these numbers are not part of the file. They are established just for the editing session, and they disappear right after the session is over. To finish the line insertion exercise, type:

```
4|
```

and press [ENTER]. This causes the next line to be added before line 4—that is, after line 3, the last line in the file. Now type:

**This is the last line.**

and press [ENTER]. Then list the file again. Remember to leave the insert mode (by pressing Control [CTRL] and Break [SCROLL LOCK BREAK] at the same time) and then list the file with the L command.

EDLIN will display:

- 1: This should be the first of three lines.
- 2: This is a demonstration of the use of EDLIN.
- 3: 0123456 ABCDEFGHI abcdefghi.
- 4: This is the last line.

\*

In such EDLIN commands as list and delete, line numbers can be used to specify a range. A comma is used to separate the line numbers in the range specification. To illustrate this use of line number parameters, list only lines 2 and 3 by issuing the EDLIN command

### **2,3L**

The results will be as expected:

- 2: This is a demonstration of the use of EDLIN.
- 3: 0123456 ABCDEFGHI abcdefghi.

\*

Issue the EDLIN command

### **E**

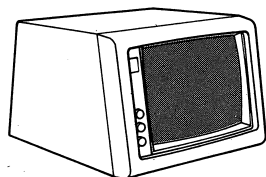
to end the session and save the file. Figure 6-4 shows a trace of the screen for this exercise. Get the directory of the EXAMPLES diskette with the DOS DIR command.

Surprised? Are you wondering how that DEMO.BAK file got there? Well, EDLIN backs you up after each session by saving the previous copy of the file. That way, you can avert a disaster, if you inadvertently ruin data during an editing session. Here is where that RENAME command in DOS can come in handy—to change the name of the backup file to a new name for editing.

## **Line-Editing Commands**

It is now time to put EDLIN to the test by using the editing features. You will also see why you typed that odd line of numbers and letters into the DEMO file.

A line can be edited with the use of the cursor control, editing, and function keys. As soon as you press [ENTER], the edited line



```

A>EDLIN DEMO
End of input file
*L
  1:*This is a demonstration of the use of EDLIN.
  2: 0123456 ABCDEFGHI abcdefghi.
*1|
  1:*This should be the first of three lines.
  2:^C
*L
  1: This should be the first of three lines.
  2:*This is a demonstration of the use of EDLIN.
  3:0123456 ABCDEFGHI abcdefghi.
*4|
  4:*This is the last line.
  5:**C

*E

A>

```

**Figure 6-4** EDLIN Example 2. Trace of screen during editing session.

replaces the original line in your file. Should you decide that the change was wrong or even worse than the original, just press the Escape [ESC] key to abandon the change and return to the original.



## ***Summary Of EDLIN Commands***

<b><i>Command</i></b>	<b><i>Action</i></b>
D	Delete lines
E	Ends session, saves edit
I	Inserts lines
L	Lists lines
Q	Quits session, aborts edit



## ***Summary Of Editing Key Functions***

---

[DEL]	Skips one character in template
[ESC]	Cancels editing
[F1]	Copies one character
[F2]	Copies all characters up to specified one
[F3]	Copies all remaining characters
[F4]	Skips all characters up to specified one in template
[F5]	Causes editing to become template for additional editing
[INS]	Starts and stops insertion on editing line

First EDLIN will display the original line (known in the DOS manual as the *template*). Beneath this display, you will see the line number. As you press each function key, different portions of the original line or template will be displayed. If you use the Insert [INS] or Delete [DEL] key, you can add new material or remove anything you want deleted.

Now, we're ready to go.

We'll start by calling up the DEMO file. Get the DOS prompt and type:

### **EDLIN DEMO**

After the asterisk is displayed, get to the numbers and letters line in the DEMO file by typing:

**3**

and pressing [ENTER]. EDLIN will display the template for line 3 and will show you that the line is ready for editing by repeating the line number and displaying the asterisk below the template. At this point the screen will look like this:



```
A>EDLIN DEMO
End of input file
*3
3:*0123456 ABCDEFGHI abcdefghi.
3.*
```

**[F1] Key** The [F1] key copies one character at a time from the template to the editing line. Just press [FN] and then the [1] key until you have displayed all the numbers on the editing line. The easiest way to perform this operation is to hold down the [FN] key with one finger and then repeatedly press the [1] key with your other hand.

**[F3] Key** The [F3] key copies any remaining characters from the template to the editing line. Now press the [FN] key and then the [3] key to complete the editing line.

**[ESCAPE] Key** Inasmuch as you really did no editing in this exercise, press the Escape [ESC] key and then the [ENTER] key to abandon the editing line. The asterisk prompt will reappear at the side of the screen to indicate that EDLIN is ready for a new line command. Press [3] and [ENTER] to return to editing line number 3.

**[F2] Key** The [F2] key copies all the template characters up to, but not including, a character you specify as a *terminator*. Press [FN], then [2], and type:

```
|
```

for our terminator character. The display will look like this:

```
*3
3:*0123456 ABCDEFGHI abcdefghi.
3:*0123456 ABCDEFGH
```

**[INSERT] Key** The Insert [INS] key allows you to insert characters into the editing line at the cursor position. As soon as you have inserted the desired characters, be sure to press [INS] again to halt the insert operation.

On with our exercise. We're going to use the Insert key to add the letters "JKL" to the editing line. First we have to get the "I" back on the editing line, so press [FN] then [1]. Next press [INS] and type:

**JKL**

Now press [INS] again to turn off the insert mode. Then complete the line by pressing [FN] with [3]. The display will look like this:

```
*3
3:*0123456 ABCDEFGHI abcdefghi.
3:*0123456 ABCDEFGHIJKL abcdefghi.
```

The desired letters were inserted properly. Press [ENTER] to save the edited line. The asterisk will appear at the side of the screen to show that EDLIN is ready for another command. Oblige by pressing [4] to edit line 4. The template for line 4 will be displayed.

**[F4] Key** The [F4] key skips over (does not copy) characters in the template up to, but not including, a terminator character you specify. The [F4] produces the opposite results of the [F2] key.

Getting on with the exercise, press [FN], then [4], and type:

**t**

Note that nothing is displayed on the screen. Now press [FN] then [3] to reveal the rest of the edited line. The display should show:

```
*4
4:*This is the last line.
4:*the last line.
```

**[F5] Key** The [F5] key shifts the editing line to the template line so that you can do additional editing. Let's do that now. Press [FN], then [5]. At this point the template and the editing line will look like this:

```
4:*This is the last line.
4:*the last line. @
```

The "at" symbol (@) shows that the shift has occurred and the editing line is now the template.

**[DELETE] Key** The Delete [DEL] key skips over just one character on the template each time. The Delete key produces the opposite results as the [F1] key. Again, the cursor does not move on the editing line and no characters are displayed on that line when the Delete key is depressed.

Press [DEL] once. Press [INS] key and type:

T

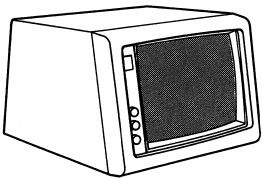
Now press [INS] again to turn off the insert mode, then press [FN] and [3] to show the rest of the line.

Relax, this is also the last line of this exercise.

This was a lot of work to end up with nearly the same line as you started with, but you did use all the EDLIN editing keys.

This time we want to save the edited line, so press [ENTER].

You can prove that the editing procedure worked properly by terminating the EDLIN (just press [E]) and examining the DEMO file with the DOS TYPE command. Figure 6-5 shows the trace of the screen for this editing exercise.



```
A>EDLIN DEMO
End of input file
*3
 3:*0123456 ABCDEFGHI abcdefghi.
 3:*0123456 ABCDEFGHI abcdefghi./
*3
 3:*0123456 ABCDEFGHI abcdefghi.
 3:*0123456 ABCDEFGHIJKL abcdefghi.
*4
 4:*This is the last line.
 4:*the last line.@
    The last line.
*E
A>
```

**Figure 6-5** EDLIN Example 3. Trace of screen during editing session.

## Recap

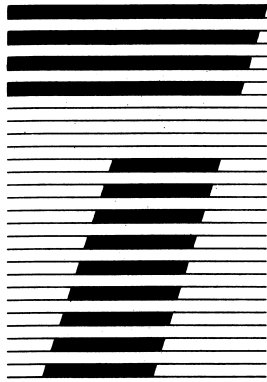
This chapter on files represents the first step in your journey to the core of the data processing experience. To quote an old Chinese proverb, "A journey of a thousand miles begins with a single step."

It takes awhile to learn how to manipulate files, but they are the crux of the computer's power. Once you master the concept, the rest of the computing challenges will be well within your grasp.

To use your PCjr effectively, you must know how to create, name, copy, examine, and edit files. As you'll see later, almost all of the applications for business and productivity enhancement involve files in one way or another. And, of course, files are the foundation of word processing. If you are catching on to this notion, you've got it made.

## Further Reading

Files and DOS are totally intertwined on the PCjr. Here again, your most dependable references are the *Disk Operating System User's Guide*, the *Disk Operating System Manual*, and the *DOS Technical Reference Manual*. Chapter 6 of the *Disk Operating System Manual* deals exclusively with EDLIN, so refer to that chapter for further details on this program.



## How Much Is That Software In The Window: Selecting And Buying What's Right For You

If you read magazines, listen to ads, or talk to other computer users, you'll soon discover that there's no such thing as the "perfect" software. What looks good in the "window" of your local computer store may turn out to be less than wonderful when you get home and start using it. There are, however, several keys to finding the best software for your needs. So, here's some scoop on software!

Now that your system is set up and operating successfully, it is time to put it to use. To do that, you must have *software*, the collection of programs in cartridges and diskettes that drives the computer. When you first obtained your IBM PCjr, the salesperson probably suggested that you purchase certain software packages, perhaps IBM PCjr BASIC and some games. This first software purchase may have added \$150 or more to the total cost of your computer.

In the excitement of getting your own computer, this relatively small additional cost may have seemed inconsequential, but a quick glance at the prices of some of the other software packages should have been sobering. If you selected only ten or twelve of the more interesting ones, the total cost could amount to almost \$1,000. The cost of the top twenty might add up to much more than you paid for your entire PCjr hardware.

All this points to an important fact about computers and computing—software can be expensive. We estimate that during the past three years, we have purchased more than \$7,000 worth of software for the two different computers we own. This software cost now exceeds the value of its accompanying hardware.

There is still another and sadder facet of the situation. We used a large number of these software packages only a few times, some only once or twice. In short, most of the money spent on software, if not completely wasted, at least was not wisely spent. The customer may always be right, but not always bright.

This situation is not unique in the field of computing. Industry experts estimate that over the life of a computer system some 60 to 70 percent of the system's cost is for software. No firm figures are available on the cost of underused software, but a safe assumption is that a lot of money is lost in this way. Careful planning and discipline are needed to get a good return on your software investment.

### **To Buy Or Not To Buy?**

You may have been told that with the BASIC cartridge and a little time and effort, you will be able to handle most of your own software needs. Don't you believe it. Writing your own production software is not an easy task. Although there are some very specific applications that only you can or should do for yourself, for the most part someone has already written just what you need. Buying a commercial package will be quicker, easier, and better than doing the programming yourself.

Your first reaction to spending several hundred dollars for a program may verge on hysteria, especially if you think you know how to do the job yourself. But don't underestimate the time and expertise involved in producing a good commercial program or package. Hundreds and probably thousands of hours have been invested in it. And the professional package is guaranteed to work.

Let's stop for a moment and make an important point about programming. In Chapter 4, we did some BASIC programming. The actual writing of a program, however, represents less than one-fourth of the entire work involved. Most of the rest of the time is spent on the developmental stages. In other words, if it takes only a day to write the program statements, the chances are it will take almost a week to do the entire job of testing and documenting the program. And these steps are necessary to ensure that the finished product will be a useful package.

Most of us just don't have many spare hours to invest in program development. We want to get the job done and get on with the best part—using the program to do something—either for fun or for work. The good news is that excellent computer programs for doing almost anything you need to do already are available, and they don't require much knowledge or expertise on your part.

The truth is the personal computer revolution didn't come about solely because of technological advances that permitted smaller, cheaper computers. The real breakthrough came when people in the industry realized that to make the idea of "personal computers" catch on, the industry would have to develop and provide programs that were totally different from those being used by specialized, highly trained computer programmers on multimillion dollar computers. These new programs would have to meet the specific needs of home computer users like you and your family.

## Computer Games People Play

In the beginning the only real "commercial" software for home computers was games. In fact, one of the first computer programs we wrote for a microcomputer was a little gimmick-game called "Catch the Bit." We wrote it for that first machine because there was little else the microcomputer was good for—and we wanted literally to play with our new toy.

Those were the days of giant computers; the microcomputer itself was considered little more than a toy. These games generally were "shoot 'em ups" with elementary graphics, but they still captured the fancy of the micro user. When the videogame frenzy really caught on, the microcomputer became a coinless arcade.

Contemporary games are far more sophisticated than those primitive games of hand/eye coordination, reflexes, and skill. A whole new breed of games has evolved from two separate sources: "Star Trek" (a strategy game using the theme of the popular TV series) and a fantasy game called "Adventure" (based on a Ph.D. thesis produced at Stanford). They involve the mind as well as the body, and they force you to become intellectually involved with strategems and designs. These games could not have been written on the early microcomputers because they require too much computational power.

But with microcomputers' new capacity for sight, sound, and storage, "games" for learning as well as entertainment have been popping up all over. You can bone up on everything from arithmetic to gambling skills with a set of PCjr games. By the way, a direct descendant of the original "Adventure" is available for the PCjr—and it's a lot easier to use than its prototype.

Modern computer owners with agile fingers and agile minds have a wide selection of packages to pick from, so don't be hasty. Shop around for one that appeals to you and your family. We'll give you some tips on game software selection to guide you through the maze of choices. And we can make you wary of some of the pitfalls. When it comes down to it, however, particularly in game selection, it's all a matter of personal preference.



### ***Checklist For Computer Games Software***

---

1. Have a "game plan." Many computer games are just minor variations on the same theme. If you purchase several "play-alikes," you'll probably end up using only one and regretting the other purchases. A little planning will avoid that situation.
2. Games are popular for a reason. Lists of best-selling games—akin to the best-seller lists for books—are published in the computer magazines. Your local computer store should have a copy for your perusal. Look at the top sellers before you invest in a new game. Better yet, try it out.
3. The quality of games can be measured just the way you evaluate any other type of software. The normal criteria for examining games include: entertainment value, ease of use, documentation, error handling, effective use of graphics and sonics, educational value, and warranty. Be a little hard-nosed and check for all these points when you consider the purchase of a new game. You'll be surprised how quickly you can eliminate the weaker products.
4. Games normally are classified in the following categories: games of strategy, instructional games, gambling games, detective games, arcade games, board games, and simulation games. Many of the categories overlap, but this guideline will



help you build a balanced assortment. With experience you will become more discriminating about games, and you'll probably be able to develop categories of your own.

5. Some excellent computer board games are available, such as chess and checkers. We know of one young chess champion who hones his skill by playing against the computer.
6. When you're examining games of strategy and skill, be sure you have the ability to set the level of play. In our experience of running a computer camp for youngsters, we discovered that if the skill level is just a little beyond the players, they become frustrated and bored immediately.
7. Many public domain games are available, so be on the lookout. Often it's worth the price of admission to a computer club just to get some of the exceptional games they have—free of charge.
8. Computer magazines and books offer listings of games in BASIC (like the game in Chapter 4 of this book). Because more and more libraries are carrying computer magazines these days, you don't have to go out and buy all these magazines. Children with a little experience in BASIC often can keyboard these listings and make the games work themselves. This is a rewarding experience, similar to the tinkertoy effect—the children can enjoy building the games and playing them.

## The Big Three Software Packages: Word Processing, Spreadsheets, And Data Base Management

Today, it is hard to justify the purchase of a computer such as the PCjr merely for playing games. One can acquire a supertoy in other, less expensive ways. Computer users want personal productivity enhancement as well as entertainment from their home computer investment.

Three major types of software packages have revolutionized the home computer world by making sophisticated computing technology available to everyone. The IBM PCjr is capable of handling examples of each of these. The "big three," as they have come to be known, are *word processing*, *spreadsheets*, and *data base management* programs. These three applications have accounted

for more sales of personal/home computers than all other uses combined.

In this chapter we will examine what each of these categories of program can do for you. Then, in the next three chapters we will examine specific examples of each type of program so that you can get an idea of which—if any—will fit your computing needs (and budget).

### Word Processing By Computer

Word processing means exactly what it says—processing words. More specifically, word processing programs are designed to let you write using the computer instead of a pencil and paper or typewriter. They also provide a number of advantages that other ways of writing don't. Word processing programs allow you to correct mistakes before they get onto the paper, to "cut and paste" sections of what you are writing from one place to another, to align both the left and right margins (a typewriter usually aligns only the left margin), and to change "formats" (styles) in midstream with ease (such as changing an entire document from single spaced to double spaced).

The major competition for word processing programs has come from "dedicated" word processors—machines similar to personal computers except that they do nothing but word processing. These machines, comparable or greater in cost (and larger in size) than your PCjr computer, have found a large market in industry. But for personal use most people—even frequent authors like us—prefer to have a computer, so that when we finish writing we can use our computer in other ways.

In Chapter 8 we will discuss specific word processing programs. However, as a small introduction to the subject of word processing, think back to your experience in the past two chapters, first with BASIC and then with EDLIN. You already have gained more word processing expertise than you realized.

For starters, word processing programs are screen oriented. That means they use the cursor (flashing bar of light) to let you know where you are within the material you are writing. The location of the cursor points to the spot where the next operation will take place (whether you're typing, correcting, moving, or deleting). If you see an error elsewhere in your writing, you simply move the



Displaywriter, a dedicated word processor.

cursor to the error by using the cursor control keys. When you have reached the error, you type in the appropriate changes and go on from there.

Word processing programs, however, are more than just a set of cursor control keys. Most such programs have powerful search and replace commands that let you change something you know is in your text without having to move the cursor right to it. Usually, word processing programs make heavy use of the function keys ([F1] through [F10]) as well as the other special keys (like [CTRL], [ALT], and [ESC]).

When you used EDLIN in the preceding chapter, you were actually using a crude word processing program. EDLIN is known as a line editor, which means it handles corrections and changes one line at a time. Word processing programs operate as page editors, so that you can make changes anywhere on the page. Simple use of the keys [PG UP] and [PG DN] (combinations of the [FN] key and [←] and [→]) permit you to move from one page to the next within your document.

As an overall assessment of computerized word processing, we think it is fair to say that once you get used to working with your word processing program (a process that may take as long as forty-five minutes) you will *never* willingly write anything longer than four lines with a pencil and paper again. Just mark our words!

### **What On Earth Is A Spreadsheet?**

Obviously, without certain technological breakthroughs (such as the development of microprocessors and floppy disks) the home computer revolution could not possibly have happened as it did. We don't tend to think of home computers as themselves having revolutionized the big computer industry, but in fact they have. The introduction of spreadsheet programming to personal computers was such a breakthrough that these programs were modified and adopted for use by big computer users.

Just what is a "spreadsheet" anyway? Spreadsheets are nothing more than computerized ledger books. Ledger books are accounting books made up of rows and columns of numbers, as you might use for sales records, inventories, or other number-oriented record keeping (Figure 7-1). What a computerized spreadsheet does is set up rows and columns of numbers on which the computer can add, subtract, multiply, divide, calculate interest, compute statistics, or perform any number of mathematical manipulations. The computerized spreadsheet can also update the entire ledger instantly when even one entry is changed. Additionally, some spreadsheet programs can draw graphs and charts directly from the data contained within the ledger columns.

The maximum size of the spreadsheet is different for different programs, but it can be as high as several hundred columns wide by several thousand rows long! Spreadsheet programs are powerful tools for a variety of needs, from household budget planning to forecasting sales for a product.

### **Data Base Management**

Any records you can store in a filing cabinet or index-card file can be kept in the PC*jr*. The advantages of storing records in your computer are several-fold. Retrieval of records is much faster with a computer than a filing cabinet. More important, the data manip-

APRIL 1984

DATE	PAYEE	CK No.	AMOUNT	GROSS WAGES	P/R TAX EXPENSE	PURCHASES			RJM
						FTN/MFG	COYS PAPER	MISC RECEALE	
4/5/84	P&Q PAYROLL		155196	138533	13163				
4/18/84	P&Q PAYROLL		156198	139452	13246				
4/19/84	SERVICE CHARGES		725						
-	NSF CHARGES		6150						
4/3/84	NSF CHECK GAIL BRENNEN		751						
5/4/83	MUNICIPAL TELEPHONE	1238	3633						
✓	PP+L	1239	78575						
✓	GEORGE BEVERLY	1240	50000						
4/6/84	CRYSTAL WINDOW SVC.	1241	2600						
✓	NEW CASTLE CREAM Co.	1242	67650			67650			
✓	PRATT UNIFORMS	1243	11920						
✓	INSTANT TOPPING	1244	14425			14425			
✓	PILKNEY PROTECTIVE SVC.	1245	14668						
✓	DIXIE SUPPLY	1246	27838			26413			
4/9/84	POSTMASTER	1247	1000						
✓	GEORGE BEVERLY	1248	20000						
✓	NEW CASTLE CREAM Co.	1249	56375			56375			
✓	B+W DIST.	1250	74044			50365	20345		
✓	P. BAUER + SONS	1251	62500						
✓	MORRIS + ASSOC., INC.	1252	34000						
4/11/84	FROZEN CANDIES INC.	1253	14269						
✓	N. BENNETT	1254	3500						3500
✓	COFFEE VENDING Co.	1255	2400			2400			

Figure 7-1 This type of handwritten ledger sheet has been used by businesses over the years. More and more companies (and individuals, too, for that matter) have switched to electronic spreadsheets to simplify their accounting procedures.

120 (U) +H20\*12

HOME BUDGET, 1979			
MONTH	NOV.	DEC.	TOTAL
SALARY	2500.00	2500.00	30000.00
OTHER			
<b>INCOME</b>	<b>2500.00</b>	<b>2500.00</b>	<b>30000.00</b>
FOOD	400.00	400.00	4000.00
RENT	350.00	350.00	4200.00
UTILITIES	110.00	120.00	1300.00
TRANSPORT	500.00	500.00	6000.00
ENTERTAIN	100.00	100.00	1200.00
SAVINGS	50.00	50.00	600.00
REMAINDER	40.00	30.00	1225.00
<b>EXPENSES</b>	<b>2460.00</b>	<b>2470.00</b>	<b>28775.00</b>
SAVINGS	30.00	30.00	360.00

Visicalc, an electronic spreadsheet. Compare this to the ledger shown in Figure 7-1 to see how a computerized worksheet can save countless hours of work.

ulation possible with a computer far outweighs anything you can do with a filing cabinet.

As an example, let's say you had a drawer full of folders, each containing the last month's sales invoices for each salesman in your company. Now, your boss asks for a report on all IBM PCjr computer sales for the past month. You would have to remove each folder and search each invoice for those involving PCjr computer sales. Then, you would have to tabulate the individual invoices and, eventually, you could produce your report.

However, if your records were computerized, you could quickly search for "IBM PCjr computer," and within seconds you would have a complete report showing the total PCjr computer sales of each salesman for the past month—subtotaled, if you wished, by sales district. You could have included a graph or two to indicate if PCjr computer sales were increasing or decreasing.

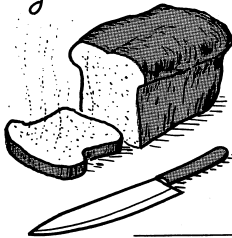
Here's another example, this time from the kitchen. Many people keep recipe card files, usually on three-by-five-inch index cards. Typically, the cards are indexed under the type of dish: "Des-

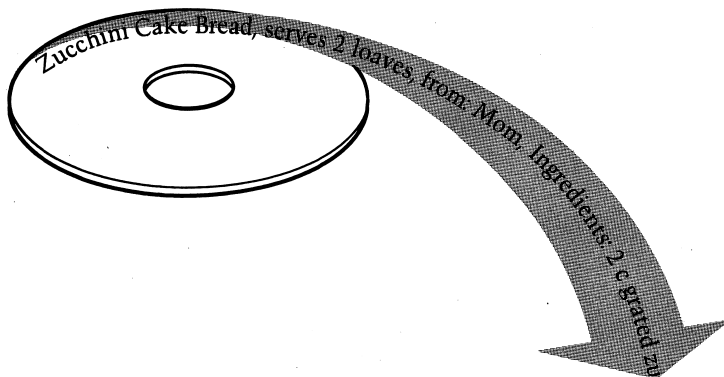
serts," "Fish," "Casseroles," and so forth. But what if you were cooking dinner for some people and you knew they wanted a dish with curry seasoning and they didn't care if the meat was chicken, beef, or whatever. You might have to search methodically through all your main dish recipe cards to find the few that had curry in the name or the ingredients. Or, you could have used your computerized recipe file to quickly pick out the three or four suitable curried main dishes from your hundreds of recipes (Figure 7-2). Which method would you prefer?

Here's what's cookin' Zucchini Cake Bread Serves 2 loaves  
 Recipe from the kitchen of Mom

ingredients: 2 c. grated zucchini  
1 c. oil 1 tsp. baking soda  
1 2/3 c. sugar 1 tsp. salt  
1/3 c. brown sugar 3 tsp. cinnamon  
3 eggs 1/2 c. nuts/raisins  
2 tsp. vanilla optional frosting  
3 c. flour 1 3oz. pkg. cream cheese  
1/4 tsp. baking powder 2 c. powdered sugar  
1 tbsp. butter

combine & bake at 350° ~ 45 min.





**Figure 7-2** Even recipes can be put on diskettes. Storing recipes on your computer won't make you a better cook, but you may find yourself creating more balanced meals.

These examples don't do justice to the full potential of computerized data bases as convenient and powerful time-savers. In Chapter 10 we will look in greater detail at further examples of data base applications. For now, consider the records you keep (or don't keep) by hand, and see if you can think of other areas where data base management programs would help you.

In the next three chapters we will discuss specifically the "big three" programs available for use with the *PCjr*. The programs discussed in greatest detail are those programs that are cross-compatible with the *PCjr*'s senior partner, the IBM Personal Computer. We do this for two reasons. First, many people who own a *PCjr* will have access—either through work or a friend—to an IBM Personal Computer and its software. This gives an opportunity to "test-drive" the software before you buy it. For some, the availability of an IBM Personal Computer at the office and a *PCjr* at home offers the chance to complete office work at home, if you have the right programs.

Also, the success of the IBM Personal Computer has ensured the availability of a wide range of high-quality, time-tested software. As the popularity of the *PCjr* grows, more and more of this extensive library of software will be modified to fit the *PCjr*'s specifications. So, an initial familiarity with the more sophisticated programs in the PC line will prove helpful as a wider range of programs becomes available for the *PCjr*.

## Where To Buy Software

Now that you see you may need some additional software, where do you get it? You have a number of alternatives: computer stores, specialized software stores, mail-order software firms, and the software vendors (manufacturers) themselves. Each has its advantages and disadvantages.

Buying a program from a computer store is somewhat like buying records at a stereo store: they may be for sale, but chances are they are not the main concern of the salespeople. This is reflected in the variety of software available, the knowledgeability of the sales force, and whether the store sells the most up-to-the-minute software. Most likely, they will try to sell you what they have on hand, even if there is a slightly better or newer alternative on the market.

With the growth of the home computer market, specialized software stores have sprung up. These are stores that do not deal



with computer hardware but concentrate exclusively on selling programs. Obviously, if these stores did not have a better selection and more qualified salespeople than the average computer store, they wouldn't stay in business long. Generally speaking, of all your purchasing alternatives, these software stores provide you the most support *after* the sale. For the novice, this can be a big plus.

The IBM Product Center, which sells the entire line of IBM equipment, is one place to shop for PCjr products when hand-holding support is a valuable commodity. Not only will the center have the proper software for your PCjr, its personnel are prepared to suggest software and other IBM-related products that will suit your needs. But don't expect to find other, non-IBM brands that may be compatible with the PCjr systems. In addition, the Product Center is not the cheapest place to get your hardware and software. But you can be sure of reliability and service, even after you've used your equipment or worked with the software. Particularly now, when you're still on shaky ground in the world of computers, you'll be well advised to consider these factors along with all the other attributes with which you judge a supplier. We have never had any problems doing business with service-oriented stores like the IBM Product Center, even when returns and repairs were necessary.

Mail-order software houses usually have good selections and sometimes are the only source of that hard-to-find program you need. Too, mail-order prices often are lower than you will find in stores—as much as 15 to 25 percent less. This can turn into big savings if you are talking about ordering a number of expensive programs. On the negative side, you can't expect much support from a mail-order firm once the program is in your hands. Usually they will replace programs damaged in the mail, but that's about all. You may also find that the version you receive in the mail is not the latest version of the program.

Whenever you buy from a mail-order house (or any computer store, for that matter), be sure to specify the exact version or release number you want, and of course the type of computer for which it will be used. Fortunately, the IBM PCjr is a popular computer, so finding compatible software is rarely a problem. But some of your friends with different computers may find themselves having to look long and hard for a specific program that runs on their machine.

Purchasing software directly from the manufacturer pretty much guarantees that you get the latest edition—sometimes hot off the



## ***Buying Software: Advantages And Disadvantages Of Alternative Sources***

	<b><i>Advantages</i></b>	<b><i>Disadvantages</i></b>
Computer stores	Convenience Some salespeople may be familiar with software	Software is not the main concern of salespeople Stores lack variety of software Software not always most recent version Salespeople not always knowledgeable
Specialized software stores	Salespeople very experienced Wide selection of software Good customer support	Higher cost Stores don't carry hardware
Mail-order houses	Wide selection of software Prices often lower than retail	No customer support Software not always most recent version Returning software can be difficult Delivery time can be long
Software manufacturers	Can obtain latest version of software Manufacturer usually is helpful answering questions by phone	No personal support Can be time consuming

press. Also, software manufacturers usually are helpful in answering your questions, over the phone. When it comes to face-to-face contact, though, forget it. Service is not their primary business.

Amidst this confusing jungle of software dealers, where should you go to get your software? Like everything else about computers, the answer will depend on the particular circumstances and needs of the moment.

If you know exactly what you need, and are in no particular hurry about getting it, then a mail-order house likely will be the most economical alternative. If you need a common, inexpensive program, and you need it fast, your nearest general computer store will fill the bill nicely. If you require the help of an expert to tell you exactly which program will meet your particular needs, go to a specialist at a software-only store. And, of course, if you are one of those people who just have to have a new program the day it comes out, contact the manufacturer directly.

In a nutshell, plan all your software purchases carefully, ahead of time—preferably after actually testing them out on a computer (either at the store or on another computer). Don't hesitate to use all the software supplier resources available to you when it comes time to purchase additional software.

## **"Borrowed" Software**

"Borrowing" copies of programs from friends or employers not only is a violation of the copyright laws, it also has the effect of increasing the cost of software to everyone. Industry statistics estimate that for every copy of a program sold, at least five are illegally duplicated. The software vendor is well aware of this fact and, as a result, sets prices of software accordingly. Just note the difference in price between cartridges that you cannot duplicate and reproduce and the price of a piece of software that is on a diskette. It's easy to see that you're paying for this pirating in the long run.

Aside from ethics and legality, "borrowing" software has some practical problems. To make your packages work effectively, you must have good documentation. And it's not so easy to reproduce a 300-page user's manual and all the other associated documentation. Furthermore, a legitimate owner of any programming package is entitled to support and counseling from the manufacturer and/

or dealer, things that obviously are not available to you if you "borrowed" the program.

Vendors supply updated documentation, maintenance material, and other related information to registered owners of programs. We have received material of this kind regularly for the program packages we own. This service is valuable, and you don't want to miss out on it. In addition, vendors often offer program owners special rates for other merchandise in the product line. All these factors add up to a lot of benefits for the legitimate software owner. Remember, if it's worth using, it's worth owning.

## **Public Domain Software**

Did you know that a large number of programs are now available just for the asking? When we purchased our IBM computer, the store owner offered us a set of games and music programs at no extra cost. The reason for these software giveaways is that a large body of programs is in the public domain. That is, these programs are not copyrighted and they are available for anyone who wants them. In fact, for a long time one of the best communications programs for the IBM PC was one of these public domain freebies.

How do you get your hands on these? For one thing, you have to ask. Most of them are available through users' groups, local schools and colleges, and other computer-oriented organizations.

## **Picking Your Software Packages**

The single most important criterion in selecting any software is that it does the job. You will find that there are a number of packages competing for your software dollar, each with some features that another does not offer. The wise software shopper does exactly what the wise grocery shopper does—makes a list of the things to get. There are always two different types of items in the list: the things you need and the things you want. The required items must be available or the package is not for you, regardless of the price or the other features. Unfortunately, too many shoppers confuse the two and purchase programs that don't really fill the bill in the "need" category because they happen to offer a few items from the "wish list." Don't fall into this trap. Demand the needed items, and

just treat the sizzle (as it's called in sales jargon) as a welcome bonus.

The recommendation to write these things down is very good advice, because in the excitement of shopping and running the programs, you can easily confuse the two and come home with the sizzle instead of the steak. Such purchases have a way of ending up collecting dust on your shelf—just as many of ours have.

## Documentation

What do you look for when you choose software? Start with the obvious—look at the package. The package should give you information about the purpose, requirements, and compatibility of the programs with the DOS 2.10 operating system. If you're in doubt, ask, but make sure you can return the software package without any restocking fees (which have become common, especially in mail orders) in the event it does not perform on your system.

A good software package contains a clear, readable *user's guide* that points out the basic functions without too much overwhelming detail or technical jargon. The *Disk Operating System User's Guide* is a good example of what we're referring to here. This kind of user support is more prevalent now than ever, but it is still not available on every package. And it is something you will have to pay for someplace in the package price. Keep in mind, however, that "bargain" packages are no bargains if you need a computer scientist to get them working.

The next ingredient is the *technical reference manual*. That document should answer any questions you have about the package. The manual should be comprehensive but not so formidable that you can't use it.

Judging the quality of a technical manual might seem beyond the capability of a novice computer user, but it really isn't. You can test the manual both for organization and readability. Take a look at the way the commands are described. The discussion should begin with a brief but clear explanation of the function of the command. Then it should give you a generalized model of the command, using a symbolic notation. These formulas or symbols will seem strange at first, but once you start using the package, they can be extremely helpful—if the manual is well designed. There

should also be a number of specific examples and illustrations showing what happens when the command is used and the manual should include an index, glossary, and appendices to help save you searching time.

Make sure the vendor has included a quick reference card. This sturdy foldout card should contain a summary of the commands, functions, services, or other goodies you spent your hard-earned money to get from the package. It can save both errors and time and should be a normal part of the documentation package. With this handy reference card by your side, you can make much better use of the system.

The documentation support for a software package is easy to examine, and you definitely should review it before making a purchase. If there ever was a case of "what you see is what you get," it is computer software documentation. If the proper documentation is not included at the start, either it is not available, or it will cost you a hefty extra premium. Either of these two cases should be avoided. A number of good manuals are available to use as a benchmark for evaluation. The *BASIC Reference Manual* for the PCjr is exceptionally well done. Don't expect less from other software vendors.

### **Software User Friendliness**

Before you purchase a program, you should try to run it yourself or at least see it running. What to look for in these programs is a property known in the trade as "user friendliness." What this means is that the program is designed to assist you every step of the way by including a help file to explain commands, by showing you a menu of the commands you can select, and by preventing you from self-destructing. The commands should be easy to understand, so if you have trouble following the instructions, this package is not for you.

### **Software Efficiency**

Too many people get hung up on the nitty-gritties of execution speeds, transfer rates, and other buzz words. If the package runs on your system and utilizes the resources you have, the few milliseconds or kilobytes of difference between packages is inconsequen-

tial. The magazines and computer salespeople tout benchmark performance, quoting statistics that one program will execute 10 percent faster than the others. If you stop to consider that you're probably spending ten seconds or more just to make a single entry on your PCjr, the speed of execution just isn't the factor that will increase your efficiency. So don't be intimidated or impressed by those "speed and feed" statistics.

Don't lose sight of your prime objective—to get the job done the easiest way. Your time is the most important element in the efficiency equation, not the speed or storage of the computer. The total time it takes to complete a task is the real test of the speed of any program.

## Information About Software

Your best sources of information about software packages are local schools and universities, computer magazines, computer stores, and your local newspaper. The IBM Product Center should be aware of all the PC computer clubs or user groups and should be willing to supply names, addresses, and times and places of meetings. Through such organizations, not only will you gain access to the large body of public domain software, but you can share experiences and learn from other computer users.

Another software information source is the "computer fairs" that are held regularly in most large cities. Because such fairs bring together a variety of vendors anxious to display their wares, they are well worth attending. The fairs are listed in the PC magazines and often are advertised in the financial or business section of your local newspaper. Most fairs charge an entrance fee, but often the discounts available to fair purchasers more than offset any admission fees. And you can't put a price tag on the wealth of knowledge you can gain from talking to these professionals and getting hands-on experience with a variety of software packages and hardware configurations.

## Further Reading

*Buyers' Guides.* These books are interesting to look at, but you probably won't want to own one. They become dated quickly. There are special buyer and data processing review services

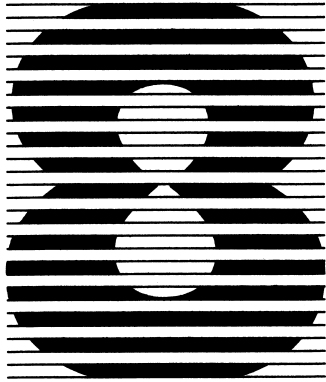
that do stay current by sending out updates. Usually these services are extremely expensive, though, and are designed for data processing professionals and purchasing agents at large industrial organizations rather than home computer owners.

*The financial or business section of your local newspaper.* More and more attention is being given to the computer user by the business and financial editors. Articles and advertisements about software and hardware appear frequently on these pages, and any sizable PC fair will advertise there as well.

*The Guide To Personal Computer Offerings From IBM* (IBM Corporation). The definitive document on IBM's offerings for the PC product line, both in hardware and software. Up to IBM's usual high standards, this booklet has excellent illustrations and clear descriptions. It is issued periodically, so be sure you see the latest copy.

*PC magazines and periodicals.* Mail-order house advertisements fill a sizable portion of the pages of the PC magazines and periodicals, offering you a baseline from which to compare current prices.





## **Word Processing, Or How To Make The Eraser Obsolete**

Word processing has become one of the most widely used personal computer applications—and with good reason. The development of word processing machines (both computers and so-called dedicated word processors) represented the first major advance in writing in over a hundred years. The basic tools used for writing—pens, pencils, and typewriters—all were developed over a century ago. No wonder the world was ready for a new way to write.

Let's examine the advantages of word processing by computer. Word processing programs allow you to create documents of any type—letters, memos, novels, term papers, or computer books. There is no restriction whatsoever on what you can process. If you can enter it on the keyboard, the computer can handle it. Correcting errors is far easier with a computer than in any other form of writing—something you already have seen with your BASIC and DOS programming efforts. Major changes, which would require redoing whole sections of a typewritten paper, can be accomplished instantaneously. Best of all, there are never any erasures or red pencil marks on your document as you edit it. So, word processing just might make the eraser obsolete.

For most people the most intimidating part of using a computer for word processing is the need to type everything. People often worry about not typing fast enough or accurately enough. Because the computer is infinitely patient, it doesn't mind at all if you "hunt and peck" with two fingers. Touch typing on the computer, however, can be faster and easier than on a typewriter. Why? The answer is simple.

The main factor limiting typing speed is the number of errors you make. Anybody can type quickly (well, almost anybody), but most people can't type quickly *and* accurately without a lot of practice. Because the computer makes correcting errors so easy, you can type at a faster rate than would be practical on a typewriter. Some people find that they always make the same mistakes if they type too quickly, like typing "hte" instead of "the." With a word processing program you can type "hte" a hundred times and correct every mistake with a single command. There is no need for time-consuming, sloppy white-out or for erasures.

Many people cite another, more subtle, factor that gives word processing an advantage. We like to call this the "perfection complex." Some people, including the authors of this book, are prone to falling into a trap, whether they write with a pencil and paper or compose at the typewriter. Imagine this scenario. A romantic young lover is composing a letter to his beloved. The message goes something like this:

December 23, 1545

Juliet Capulet  
2185 Napoli Boulevard  
Padua, Italy

Dearest Juliet,

Hark, what light in yonder window breaks.

Suddenly, the young writer realizes this just isn't the right first sentence for his love letter. So, in a fit of pique he tears it up and begins over. But again he isn't satisfied with his opening line (which, after all, must be poetic, clever, witty, and romantic all at once). After several tries, young Romeo looks down and sees that his wastebasket is filled with discarded sheets of paper, all of which contain only one sentence. Finally, frustrated and downcast, he takes to the streets to express his feelings verbally. The rest of the story, of course, is history.

While this tale may seem improbable, there is a surprisingly great deal of truth underlying it. If you're like many writers, what you are writing must be relatively free of errors before you are satisfied with it. You probably write the heading or title very neatly,

only to throw the paper away if the first line (the hardest line to write in any composition, by the way) isn't quite perfect.

What has all this got to do with the computer? Speaking from personal experience as charter members of the Perfectionists' Society, we have found that using a computer short-circuits the process by allowing as many changes as necessary on a section of a document without affecting the rest of the text at all. Therefore, as we write we have a neat, clean document, and even if we have to change the first line two or three times to get it just right, we never need to redo anything else.

Another major advantage of the computer is the ability to make major changes throughout the text with very little effort. For example, you can change an entire document from single spacing to double spacing or alter the margins without having to redo any typing. If you want to change a single word or phrase that appears multiple times throughout the document, a single command will accomplish this.

Form letters become a snap with the computer. You can change the name and address of each recipient while keeping the text of the letter unchanged, to produce personalized form letters. If you have a second file containing a list of names and addresses, you can merge the list and the letter. Then the computer creates a letter personally addressed to each name on the list, with essentially no extra effort on your part.

Well, enough of the possibilities. It's time to get down to facts.

First, let's examine the word processing programs currently available for the IBM PCjr. Then, we'll zero in for a closer look at EasyWriter 1.15—probably the best word processing package available for the PCjr (and fully compatible with the IBM PC and XT series computers). Finally, we will present a brief discussion of other word processing packages that are *not* currently available for the PCjr but are anticipated for the future.

## Word Processing—What's Out There

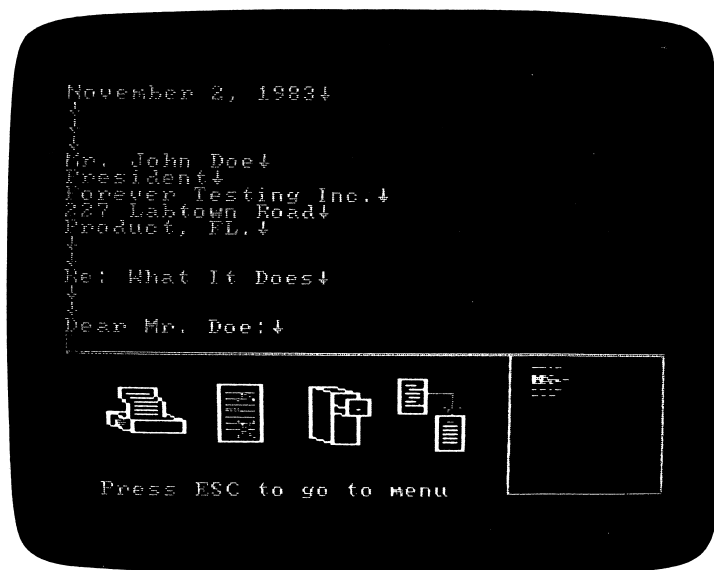
Currently, IBM recognizes six programs as compatible with the PCjr, using DOS 2.10 as the operating system. One of these, HomeWord, was designed especially for the PCjr and cannot be

used on other IBM computers. Another, called Peach Text, is designed for use with two disk drives, and IBM discourages its use on the PCjr. All the word processing programs available require the enhanced model of the PCjr computer, which has a disk drive and 128 kilobytes of memory.

The other packages available are EasyWriter 1.15, Professional Editor, Personal Editor, and WordProof. The last one, WordProof, is a proofreading program that checks spelling and corrects typographical errors. The other three are fairly similar, with the two editor packages having few advantages over the more popular EasyWriter.

### HomeWord

The unique HomeWord word processing program deserves special mention. Figure 8-1 shows the HomeWord screen as it appears in the course of composing a letter. Notice the distinctive three-part screen: The main text is in the upper section, the available options are represented as pictures in the lower left, and the entire page is



**Figure 8-1** HomeWord screen display. Designed for the PCjr, HomeWord has simple commands that make it easy for first-time computer users.

depicted in miniature in the lower right. To select from the available commands, you position the cursor over the picture of the option you desire (for example, over the printer, if you wish to print the letter).

HomeWord is designed for people who have no experience whatsoever with computers or word processing. From that standpoint it does an admirable job. However, it has two main drawbacks if you require serious word processing capability. Although the picture-controlled options may seem like a catchy idea at first, as you get more familiar with word processing, they become cumbersome. These symbols can be bypassed, but basically HomeWord lacks the sophisticated word processing capabilities of EasyWriter or similar programs. In addition, HomeWord's lack of compatibility with other computer systems means that you will need to learn to use a second word processing program if you wish to work with someone else's system or upgrade your own.

### **EasyWriter**

EasyWriter 1.15 is likely to be the most popular word processing package for the PCjr—at least until modified versions of other word processing programs come along. EasyWriter can be used on the IBM PC and XT computers as well as on the PCjr. EasyWriter is a screen editor, unlike EDLIN, the line editor you used in Chapter 6. This means that EasyWriter allows you to change anything on the screen by moving the cursor. EasyWriter also allows you to input your text one page at a time as you type.

We will begin our look at EasyWriter by overviewing some of the features EasyWriter shares with other word processing programs. The format of the screen is similar to most popular word processing programs. The bottom half of the screen is the area that contains the text you create—the document itself. Immediately above the text area is the ruler line (marked in the photograph with the numbers 1 through 7). This line indicates the left (L) and right (R) margins and the tab settings (number or +). It functions literally as a ruler against which you can measure the length of the lines you are typing. The ruler line separates the text from the command menus, which are above the line. These menus enable you to see the codes (usually a single letter or function key) that perform the many word processing functions available with EasyWriter.

```

                                A D D I T I O N A L   C O M M A N D S
-----
A - ALIGN TEXT                M - MARGIN SETTINGS          T - TAB SETTINGS
C - CENTER A LINE            N - PAGE #/# OF COPIES      W - WORD COUNT
H - HMI SETTING              P - PRINT TO SCREEN         F4 - GO TO EDITOR
J - JUSTIFY ON/OFF           S - SEARCH AND REPLACE     F10 - GO TO FILE SYSTEM

COMMAND?
L-----1-----2-----3-----4-----5-----6-----7-----R
The IBM Personal Computer is compact, easy to learn and easy to
use. It offers big-system benefits and advanced design features
which add to the rewards and enjoyment of having your own personal
computer. You don't have to be a computer expert to learn how to
use your IBM Personal Computer. You, or any member of your family,
can do it -- with just a little time and practice. A special set
of instructional materials comes with your IBM Personal Computer,
allowing you to get to know it on your own terms, at your own
pace. IBM instructional literature is written in your language,
not computer jargon. It was designed to help take the mystery out
of computing and make learning simple -- even fun.

```

EasyWriter screen display.

Every word processing program is organized in a somewhat similar fashion. When you enter text with the keyboard, you can make immediate corrections and adjustments by using the four cursor control keys (the arrow keys) along with the Insert [INS], Delete [DEL], and Backspace [←] keys. The [HOME], [END], [PG UP], and [PG DN] keys (combinations of the [FN] key with the four cursor control keys) allow rapid movement through your document.

EasyWriter has a "word wrap" feature common to most word processing programs. This feature makes typing with EasyWriter different from typing with a standard or electric typewriter. EasyWriter keeps track of the end of each line of typing without the need for you to use a carriage return, as you would on a typewriter. You use the [ENTER] key—the PCjr's version of a carriage return—only at the end of a paragraph or solitary line like a heading or title. Typing experts estimate that this feature alone cuts typing time by about 15 percent.

When you strike [ENTER], EasyWriter places a musical note symbol (♪/♪) on the screen to mark the end of the paragraph. (This symbol is not displayed on the final printed copy.)

Major changes in your document are made by using the EasyWriter command keys. Because some of these commands are executed by striking a single letter, EasyWriter needs some way to differentiate between what you are typing in (the text or document) and the command you wish to execute. All word processing programs share this dilemma, and all handle it somewhat differently. With EasyWriter, you use two special function keys, [F4] and [F10], to indicate that the next letter or letters are commands, not part of the document. When you have executed all the commands you desire, you can quickly return to entering text.

The last major feature that EasyWriter shares with other word processing programs is file management. You have already had an introduction to files and file management in Chapter 6, so some of this may be familiar. Each document you create with EasyWriter is kept in a separate file. When you wish to make changes in a particular document, you just tell EasyWriter to “get” that file from the diskette on which it is stored, then you edit it appropriately and put the “revised” copy back onto the diskette.

The rules for EasyWriter filenames are essentially identical to DOS filenames. They may be between one and eight characters in length and may use the letters of the alphabet, numerals 0 through 9, and any of the following characters: ! @ # \$ % ^ & ( ) - \_ { } ~ ` ' \ | > <. Spaces are not permitted within the filename. Unlike DOS filenames, however, you cannot specify your own three-character filename extension. This is because EasyWriter automatically assigns the filename extension .EWF (for EasyWriter file) to each filename.

Every filename on the diskette must be different, although you can have the same name on more than one diskette. This is not recommended, as it can be quite confusing. We strongly suggest you use filenames that clearly indicate the nature of the document in the file. For example, you might be writing three letters—one to that significant man or woman in your life, one to your favorite sports hero or TV star, and one to IBM to compliment them on their wonderful new PCjr computer. You could name these LETTER1, LETTER2, and LETTER3, but it wouldn't be long before you forgot which LETTER is which letter. The names LUVLETR, FANMAIL, and IBMLETR probably would have a lot more meaning, even if you came back days or weeks later.

There is one important restriction on EasyWriter files: They must be no more than 24,000 characters in length. EasyWriter will

give a warning signal when this limit is reached, but we recommend that you don't let your files run more than 10,000 characters in length. As the number of characters increases, the speed of commands like Search, Align, and End-of-File markedly decreases. When it is time to print a document you have split into two files, EasyWriter allows you to link them up again quite easily.

**Getting Started With EasyWriter** The first thing you must do to use your new EasyWriter diskette is transfer the operating system onto it. As you recall, COMMAND.COM is the part of the operating system that allows it to boot. Once you have transferred the operating system to the EasyWriter diskette, you will be able to cold boot the system directly into EasyWriter. In addition, the FORMAT, DISKCOPY, and CHKDSK "housekeeping" functions of DOS are transferred over to EasyWriter, allowing you to perform these important chores without switching back and forth between DOS and EasyWriter.

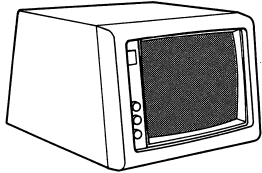
To do all this, consult your EasyWriter manual and run the SETUP1 program as instructed for single disk drive systems. When you finish that process, make a backup copy of EasyWriter, as you did for DOS. There is a major difference, however, between the procedure for copying DOS (which involved the DISKCOPY command) and for copying EasyWriter. EasyWriter is copy protected, which means it is specially encoded to permit only one backup copy to be made. Any subsequent attempts to copy EasyWriter will not produce a meaningful copy and possibly could damage the original. Use the BACKUP1 program as instructed to make your backup copy, then use the *backup* for all your subsequent EasyWriter programming.

Now you are ready to use EasyWriter. Take your new backup copy and insert it in the disk drive of your PCjr. If the power is off, perform a cold boot by switching the power on. Otherwise, perform a warm boot by striking [CTRL], [ALT], and [DEL] simultaneously. EasyWriter will automatically load DOS into memory, and then DOS will activate EasyWriter.

You will see the EasyWriter opening display, which will ask you to insert a storage diskette into drive A and then to press [ENTER]. Remove the EasyWriter diskette, insert a *formatted* diskette into the drive, and press [ENTER] again.

The first EasyWriter command menu you will see is the File System Menu (Figure 8-2). This lists all the available commands





---

FILE SYSTEM MENU

---

A - APPEND A FILE	L - LINK FILES	T - DISPLAY LINKS	1 - SELECT DRIVE A
C - CLEAR SESSION	M - DISPLAY CATALOG	U - UNPROTECT A FILE	2 - SELECT DRIVE B
D - DELETE A FILE	P - PROTECT A FILE	X - EXIT TO DOS	F2 - PRINT FILE(S)
E - EDIT A FILE	R - REVISE A FILE		F4 - ADDN COMMANDS
G - GET A FILE	S - SAVE A FILE	←/→ - SLOW/SPEED PRINT	F7 - STOP PRINT

---

FILESIZE = 1    AVAIL = 23999    DRIVE B  
0            FILES LINKED  
COMMAND:

**Figure 8-2** The EasyWriter File System Menu lists all the commands about files directly on the screen.

for the file system—the part of EasyWriter designed to handle the storage, retrieval, and processing of files. Below the menu is a blank area (if you are using a new diskette) where the name, size (in bytes), and date of creation or update of any files you create will be displayed. This section, called the catalog area, ultimately will be quite important in helping you keep track of which file is which.

At the bottom of the screen is an area that displays the current status of the EasyWriter file in memory. Inasmuch as you haven't input anything yet, this area is empty. The top line of this area displays the file size, the amount of space available, and the disk drive in use. The second line gives the name of the EasyWriter file currently in memory (again, none yet) and the number of files linked. The bottom line, which is where the cursor is located at this point, asks for the file system command you wish to execute.

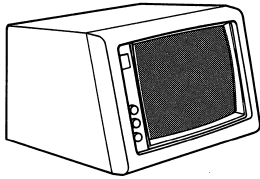
To begin your first EasyWriter document, type:

**E**

for Edit a File. You should now be looking at a blank screen with the cursor placed in the upper left-hand corner. All this space is available for entering text. However, most newcomers to word processing programs like a little more help than is provided by a blank screen. To get a listing of the available EasyWriter commands and controls at this level, just press the [F1] key, your signal to EasyWriter that you need help. You should now see the EasyWriter Help Menu (Figure 8-3) and the ruler line displayed above your cursor (now about halfway down the screen). Using the Help Menu costs you that much screen on which to visualize your document. Fortunately, the document doesn't disappear permanently, and you can make the Help Menu vanish just as easily as you made it appear. Type [F1] again to clear the Help Menu, then press the [F1] combination once again to get it to return. We will leave the Help Menu on during these introductory exercises.

Now for some real word processing. Type the following two paragraphs into EasyWriter. Try to type exactly as we have, errors and all, but don't worry if you make mistakes of your own. Just leave them, and at the end you'll be able to correct them like a pro.

Keep in mind that you press [ENTER] only when you reach the end of a paragraph—not at the end of a line as with a typewriter. (To indicate when to press [ENTER], we have placed this symbol [↵])



### HELP MENU

F1 - HELP ON/OFF	F2 - PRINT FILE(S)	END - END-OF-FILE	CTRL-
F3 - INSERT LINE	F4 - ADDN COMMANDS	DEL - DELETE CHARACTER	C - BLOCK GET
F5 - DELETE WORD	F6 - UNDELETE	HOME - TOP-OF-SCREEN	G - BLOCK PUT
F7 - STOP PRINT	F8 - BLOCK MARKER	INS - INSERT MODE ON/OFF	J - BLOCK COPY
F9 - ALIGN MARKER	F10 - FILE SYSTEM	CTRL-END - DELETE TO END-OF-LINE	O - PRINT TYPE
			PGUP - TOP-OF-FILE
			R
L			
0	+	1	+
		2	+
		3	+
		4	+
		5	+
		6	+
		7	+

**Figure 8-3** The EasyWriter Help Menu provides a quick reference of commands and controls. Having this information readily available can be a real grief-saver.

at the end of the two paragraphs. EasyWriter will use a different symbol ( ♪/♪ ) that marks the spot where you actually used [ENTER].) Note that the right and left margins of your typed material may not match exactly what is printed here in the book. Type:

**WORD PROCESSING WITH EASYWRITER IS JUST THAT—EASY!  
YOU CAN MAKE MISTEAKS AND CORRECT THEM WITH WITH EASE,  
EVEN IF YOU HAVE NO PREVIOUS EXPERIENCEWITH WORD  
PROSESSING.[← ]**

**AND RELAX—YOU CAN'T DAMAGE EASY WRITER EVEN IF YOU  
MAKE A MISTAKE IN TYPING THIS IN.WAIT, THIS SENTENCE  
BELONGS AT THE BEGINNING. [↵ ]**

Now that wasn't so bad, was it? Let's save all that work—mistakes and all—right now. To save this document in a file, we must get back to the File System Menu without destroying what we've just done (which would happen if we re-booted). The Help Menu will be of assistance here. As you look at the Help Menu, you will notice a command that says

#### F10 - FILE SYSTEM

at the end of the second column.

[F10] is the special function key that always returns you to the file system but keeps your document safe along the way. When you have reached the File System Menu, the cursor will be at COMMAND: in the bottom line of the screen. Type:

#### S

for Save a File. This command is used only the *first* time you save a file. If you make subsequent changes, you would type:

#### R

for Revise a File.

When you type S, EasyWriter completes the word "Save" and now asks you for the filename you wish to assign. EasyWriter also prompts you with a line of eight dashes to remind you that the filename may be no more than eight characters in length. Type:

**EXAMPLE**

as the filename and press [ENTER].

Now EasyWriter should tell you that the FILE IN MEMORY is EXAMPLE. The cursor is ready and waiting for the next command, so let's continue editing our EXAMPLE file by again typing E for Edit a File.

Let's correct our mistakes by using the cursor control keys to move to any errors that can be corrected simply by typing a different letter over the mistake. Use the Right Arrow [→] to reach the second "o" in the word "Yoo." When the cursor is underneath it, type:

**U**

Then use the Down and Left Arrows to position the cursor beneath the second "e" in "preveous," and type

**I**

instead. Now change the first "s" in "prosesssing" by typing:

**C**

So far, so good.

Now use the cursor control keys to position the cursor under the characters we need to delete—the "e" in "mistaks," the extra "s" in "prosesssing," and the extra blank space between "Easy" and "Writer." When the cursor is properly positioned, press:

**[DEL]**

to delete the character or extra space. Notice how EasyWriter repositions the other letters and words to close the gap created by deleting a character.

Our document is starting to shape up, but there is more work to do. We still must insert an "e" in the word "mistaks" as well as blank spaces between the "e" and the "w" of "experiencewith" and between the "." and the "W" of "in.Wait." Position the cursor below the "s" in "mistaks," then press:

**[INS]**

Notice the change in the cursor from a line to a solid box—the indication that EasyWriter is now in the insert mode. Type:

**E**

and watch how EasyWriter makes room for it in the word just before the “s.” With the cursor still indicating that you are in the insert mode, insert one blank space (using the [SPACEBAR]) between the two words “experience” and “with” and insert two spaces between “in.” and “Wait.”

Now, if we just get rid of the extra “with” in the second line (“with with ease”), things really will look spiffy. Turn off the insert mode by pressing:

**[INS]**

a second time, and then position the cursor underneath the second “with.” Look at your Help Menu now and notice

**F5 - DELETE WORD**

in the first column. Sounds like just the ticket, so press

**[F5]**

The entire word should have disappeared. This is precisely what we wanted. Sometimes, though, you realize only too late that you deleted a word or part of a word that you really needed. EasyWriter has planned ahead for this. In the second column of the Help Menu (right next to F5 - DELETE WORD), you will see:

**F6 - UNDELETE**

Try it and see what happens. Now try it again and again until the entire word “with” has reappeared, one letter at a time. Press

**[F5]**

again to remove the word once more, this time forever.

If you made any other typing errors that can be corrected using the commands we have discussed, make the appropriate corrections. When you are satisfied, save the revised version of your EXAMPLE document. To do this, you must return to the File System Menu by striking:

**[F10]**

Now type:

**R**

for Revise a File. EasyWriter will beep and ask you if you are sure this is what you want (remember, the old uncorrected version will be gone forever if you carry out this command). Because we have no use for a document full of mistakes, go ahead and type:

**Y**

for yes.


EasyWriter is asking for the next command, so type:

**E**

to resume editing the file. Whenever you are doing *any* word processing by computer, be sure to save your revisions frequently. Remember that editing done but not saved is vulnerable to instantaneous destruction by a power loss or current surge.

We've covered a lot of ground in a short time, so let's review:

1. Start by setting up your EasyWriter diskette with a built-in copy of the operating system. Then make a backup copy. Boot EasyWriter to get the File System Menu, with its list of file manipulation commands, the catalog area, and the current status of the file in memory.
2. Use the "E" for Edit a File command to begin working on your new document. To see the EasyWriter Help Menu, use the [F1] key. Press [F1] a second time to cause the Help Menu to vanish.
3. Type your document into the computer just as you would on a typewriter, with one exception: Don't press a carriage return key at the end of each line. EasyWriter keeps track of line endings

and adjusts the text on the screen appropriately, a function known as "word wrap." Use the [ENTER] key to mark the end of a paragraph. EasyWriter marks the spot where this key has been used with a special symbol (  ).

4. After you enter your document (complete with a few mistakes), use the [F10] key to return to the File System Menu without losing any valuable typing. Then use the "S" for Save a File command to save the document (in this case under the filename EXAMPLE).

5. Filenames conform to the rules set down by DOS. That is, they can contain no more than eight characters, which may be numerals, letters, or some special symbols. Filename extensions cannot be used, because EasyWriter automatically assigns the extension .EWF to all files. Filenames should serve as reminders of the contents of the file, and they may not be used more than once per diskette.

6. Use the "S" for Save a File command only when a brand-new file is to be created. Otherwise, to save revisions of old documents, use the "R," for Revise a File command. Using the "R" for Revise a File permanently erases the old version of the file, so EasyWriter double-checks with you every time to be sure you really want to go through with the Revise a File process. Files should be saved frequently during the course of the revision process, to prevent inadvertent loss of revised work.

7. To return to editing the file in memory, use the "E," for Edit a File command. Make appropriate corrections with the cursor control keys, plus the Insert and Delete keys. The Insert key activates the insert mode, represented by the solid-box cursor. Letters are inserted immediately to the left of the character designated by the cursor. The Delete key deletes the character designated by the cursor and closes the gap created by the deletion.

8. Use the [F5] key to delete an entire word, when the cursor is positioned within the word. Use the [F6] key to *undelete* a word, one letter at a time, beginning with the last letter.

**Block Operations With EasyWriter** Let's look further at the two paragraphs you typed. The last line of the second paragraph, although spelled correctly, is telling us it belongs somewhere else. This is a good chance to familiarize yourself with the "cutting and pasting" powers of EasyWriter.



If you are still following along with EasyWriter, then you should be ready to edit the EXAMPLE file. If you turned off EasyWriter, just go back through the steps described for booting EasyWriter and get to the File System Menu. EXAMPLE should be listed in the catalog area as an available EasyWriter document file. Call EXAMPLE into memory by typing:

### G

for Get a File command. (The "G" command is used anytime you want to begin editing a previously opened document file not currently contained in memory.) The status area at the bottom of the screen should now indicate that EXAMPLE is the file in memory. Edit EXAMPLE by using the "E" command.

Movement or manipulation of large segments of text is referred to as a "block" operation. The block to be moved first is marked on either side, then stored into memory (using a "get" command similar to that just described for files), and finally moved into the desired position.

Let's put theory into practice with our EXAMPLE file. The cursor should now be located at the "home" position; that is, in the upper left-hand corner. We wish to move the entire last sentence to this position. Move the cursor to the end of the file with the [END] key (a combination of [FN] and [↓]). The cursor should now be positioned below the last line of text. Activate the insert mode by pressing:

[INS]

then press:

[←]

twice until the cursor rests on the end-of-paragraph symbol (¶). Did you notice the way the cursor behaved when it reached the left-hand edge of the screen?

Now, mark the end of the block (in this case, the sentence) by pressing:

[F8]

the Block Marker key. The marker symbol ( $|>$ ) should have been inserted between the "." and the "♪/♪" symbol. Using the cursor control keys, move the cursor up and over to the "W" in "Wait," the first word of the sentence to be moved. Press:

**[F8]**

again to position the beginning of the block.

Enter the block into memory with the Block Get command, by typing:

**[CTRL C]**

([CTRL] and [C] keys pressed together). The block should disappear as it is absorbed into memory, as if it were cut out of the text with scissors. To paste the block down where you want it, type:

**[HOME]**

to return the cursor to the home position, the spot where the block is to be inserted.

The Block Put command replaces the block at the current cursor location. Type:

**[CTRL G]**

to move the block. Notice that now there is no space between the end of the relocated sentence and the beginning of the former first sentence. Use the cursor control keys and insert two blank spaces before the word "Word." Make sure the cursor is still in the insert mode before doing this.

Once again, save this revised version of the document by typing the sequence:

**[F10]**

(to enter the File System Menu),

**R**

(for Revise a File) and

## Y

(for yes—in answer to the question “Are you sure?”). Resume editing by typing “E” for Edit a File.

Let's review the material just covered.

1. To call a previously saved file into memory, use the “G” for Get a File command. Then use “E” for Edit a File once the file is in memory.

2. To move quickly to the end of the file, use the [END] key (combination of [FN] and [↓]). To return to the home position on the screen (upper left-hand corner), use the [HOME] key (combination of [FN] and [↑]). Note that the home position is not necessarily the same as the beginning of the file.

3. Mark the beginning and end of blocks using the [F8] (Block Marker) key. A block is any group of characters (large or small) to be moved or manipulated together in a single operation. The block marker symbol (|>) designates the beginning and end of a block.

4. To move a block of text, first enter the block into memory with the Block Get command [CTRL C]. Move the cursor to wherever you wish the block to end up, and reposition the block with the Block Put command [CTRL G]. Be sure to check the spacing between sentences after every block move.

5. Save your revised work frequently, especially after a major manipulation such as a block move.

***Inserting And Aligning Text*** As you have already seen, EasyWriter is not limited to manipulating text only one character at a time. We have used commands to delete entire words and move whole sentences. EasyWriter also can insert an entire sentence (or more) into the midst of a document with the Insert Line command [F3].

You should now have your corrected EXAMPLE file ready to edit, with the cursor in the home position. Move the cursor down three lines to the last line of the first paragraph. Insert a line of space directly below this line by pressing:

[F3]

There should now be a new blank line between the two paragraphs, and the cursor should be in the insert mode. Type:

**WE WILL INSERT ONE MORE LINE JUST TO SHOW HOW EASY IT IS.**

Press:

**[INS]**

to get out of insert mode, then press:

**[ENTER]**

to designate the end of the paragraph.

Now let's get our text in better alignment—it's beginning to look a bit ragged. The commands dealing with text alignment are not listed under either of the two menus we have looked at so far, so let's find the third and final command menu.

To access the Additional Commands Menu (Figure 8-4), use the [F4] key. Now the new menu appears, along with the EasyWriter request for a new command. Type:

**A**

for Align Text. Notice that the right-hand margin is still ragged, although the alignment process seems to have helped a little.

To get both the right- and left-hand margins to line up equally (a process called right and left justification), we must turn on the justification by typing:

**J**

for Justification On/Off command on the Additional Commands Menu. After confirming to EasyWriter that you do, indeed, want the justification turned on, you must now realign the text—again by typing:

**A**

for Align Text command. The margins should be even on both sides.



One of the most powerful of all word processing commands is the Search and Replace command. In EasyWriter it is found on the Additional Commands Menu. This command can seek out a word or phrase anywhere in your document and change it to any new phrase you specify. Let's try it. Type:

**[CTRL]-[PG UP]**

to return to the beginning of the EXAMPLE file. (Note that [HOME] would have worked just as well, but if the file were larger, it might not.) Now, with the Additional Commands Menu displayed (type [F4] to get it if it isn't), type:

**S**

for Search and Replace.

For this example let's say that IBM decided to change EasyWriter's name to EZWriter. Rather than go through line by line looking for the word "EasyWriter," we'll let Search and Replace do all the work.

Enter:

**EASY**

for the phrase to be searched out, being careful to capitalize and spell the word exactly as you did in your document. Replace "Easy" with "EZ" by typing:

**EZ**

EasyWriter then will ask if you wish to replace some or all of the phrases as they are encountered. We will answer:

**ALL**

for this exercise. Press:

**[ENTER]**

and watch the fun begin.

How many places did EasyWriter find and replace the word "Easy"? If you followed directions carefully, it should have performed three separate changes. However, one of those places was in the second sentence, where it wasn't part of the name "EasyWriter". Let's use the Search and Replace command again to change that one word "EZ" back to "Easy" (Figure 8-5).

Again press:

**[CTRL]-[PG UP]**

to return to the beginning of the file. Type:

**S**

and then type:

**EZ**

for the item to be replaced. Next type:

**EASY**

for the item to replace it. This time, though, replace only

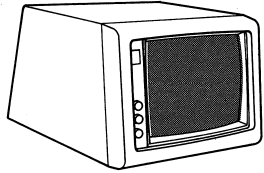
**SOME**

Now EasyWriter (or should we say EZWriter?) will ask you each time it comes across the phrase "EZ" if this is one you want replaced. Type:

**Y**

for yes only when the appropriate "EZ" has been found.

Now, save the revised version. Notice that you can go directly from the Additional Commands Menu to the File System Menu without returning to the standard Help Menu, merely by pressing [F10]. Let's summarize the commands introduced in this section.




---

 SEARCH AND REPLACE COMMAND MENU
 

---

A - ALIGN TEXT  
 C - CENTER A LINE  
 H - HMI SETTING  
 J - JUSTIFY ON/OFF

M - MARGIN SETTINGS  
 N - PAGE #/# OF COPIES  
 P - PRINT TO SCREEN  
 S - SEARCH AND REPLACE

T - TAB SETTINGS  
 W - WORD COUNT  
 F4 - GO TO EDITOR  
 F10 - GO TO FILE SYSTEM

COMMAND?

L

0    +    1    +    2    +    3    +    4    +    5    +    6    +    7    +    R

Wait, this sentence belongs at the beginning. Word processing with EZWriter is just that—Easy! You can make mistakes and correct them with ease, even if you have no previous experience with Word Processing.

We will insert one more line, just to show how easy it is. And relax—you can't damage EZWriter even if you make mistakes in typing this in.

**Figure 8-5** The Search and Replace Command is one of the strengths of word processing. With this single command you can correct errors or make other changes that would take hours on a typewriter.



1. To insert an entire line of text, use the [F3] key. This creates the space for an additional line below the line of the cursor, and it changes the cursor to the insert mode.
2. Access the Additional Commands Menu by using the [F4] key. This menu contains several commands not found on the other two menus. Examples include the "A" for Align Text, "J" for Justification On/Off, and commands affecting the margin and tab settings as well as line centering.
3. The "S" for Search and Replace command is a powerful command that can locate every occurrence of a phrase and change all or some to anything you specify. Be careful, because words or phrases you didn't expect may get hauled in by the powerful Search and Replace dragnet.
4. The [CTRL]-[PG UP] command returns the cursor to the beginning of the file, unlike the [HOME] command, which returns the cursor to the Home position on the current screen.
5. To leave the Additional Commands Menu, ordinarily you would press [F4] and return to editing. However, if you wish to go directly from the Additional Commands Menu to the File System Menu, you can do so by pressing [F10].

***Imbedded Commands*** In addition to the commands we have already discussed, which are accessed through the three menus, EasyWriter also has commands that are written right into the text—so-called imbedded commands. Generally speaking, imbedded commands are used to control the appearance of your document, either on the screen or—more often—when it is printed on your printer.

Examples of imbedded commands are the ".space2" command for double spacing and the ".eject" command, which instructs the printer to begin a new page when printing a document.

Notice that both of these commands begin with a period. Imbedded commands have a number of rules, which are summarized in the accompanying checklist. Each imbedded command must begin a new line, must start with a period, and must occupy its own line. An imbedded command remains in effect until another imbedded command changes it. For example, if you put ".space2" in your document, the document will be double spaced until a ".space1" command comes along. We've listed the available EasyWriter imbedded commands and what they do.



## ***The Seven Basic Rules For EasyWriter Imbedded Commands***

---

1. Imbedded commands must start a line.
2. There can be only one imbedded command per line.
3. Text cannot be typed using imbedded commands.
4. Imbedded commands are ended by pressing [ENTER].
5. Imbedded commands can be anywhere in the document.
6. Imbedded commands remain in effect throughout the session unless turned off or reset.
7. Imbedded commands do not print.



## ***EasyWriter Imbedded Commands***

---

.EJECT	Sets up page ejection
.EOL	Designates end-of-line character
.FORMSTOP	Stops printer at page end
.FORMSTOPOFF	Turns off .FORMSTOP command
.MARGIN	Sets left margin
.PAGE	Allows page numbering and sets up placement of numbers
.PAGELINES	Establishes lines per page
.SPACE	Sets line spacing
.TITLEA	Sets up headers and footers on page
.TITLEB	
.TITLEC	
.TOP	Establishes top margin
.USER	Allows user-defined command

***Printing With EasyWriter*** Once you have edited your document into its final form, using EasyWriter to print it out is a snap. But before you actually put it on paper, EasyWriter offers a special double-check feature not found on many other word processing programs. EasyWriter will "print out" your document in its final, ready-

for-the-paper form, but it will “print” it on the display screen. This gives you a last opportunity to view your document *exactly* as it will appear on paper, without any extraneous marks (like the ♪/♪ symbol, block markers, or imbedded commands). Then, if you want to change the text's appearance in any way, you can make the necessary corrections without wasting your time and printer paper.

To use the Print to Screen command, you must access the Additional Commands Menu by striking:

**[F4]**

Then type:

**P**

for Print to Screen. The “printing” whizzes by quite rapidly, so EasyWriter has provided speed controls. Press:

**[←]**

several times until the speed is slow enough for you to follow, then press:

**[→]**

to return to full speed. (Note that if you do not return to full speed by using the [→], the printer also will print at decreased speed.)

When you are satisfied, press:

**[F4]**

to return to the editing menu. Then press:

**[F2]**

to print the file on your printer. During the printing, typing “S” for Suspend Printing will temporarily halt the printer. Printing resumes when you press the [SPACEBAR]. Pressing [F7] will end the printing run altogether. It is also possible to print only part of a document by using the “N” command from the Additional Commands Menu.

This command will ask you to specify the page numbers you wish printed and the number of copies of each page. It appears on the Menu as "N - Page#/Copy#."



## ***Printer Checklist***

---

Your printer can be the most troublesome device in your computer system. The reason for problems is that printers are electromechanical devices. That is, unlike electronic devices (most of your computer), printers have a lot of moving parts. However, with care and a little knowledge, you can expect years of trouble-free operation from this important part of your computer system.

### ***Printer Dictionary***

---

Printers have been around a lot longer than electronic computers, and therefore they have built up a technical jargon of their own. To demystify printers, we have listed some of the key terms you will run across when you read about or discuss printers:

*Type Font or Character Font:* A term that refers to the shape of the character on the paper. The dot-matrix printer offers a variety of type fonts, ranging from the standard roman type to italics. Letter-quality printers require a different print element for each type font change.

*Character Set:* The number of characters the printer can produce. The normal character set for a printer consists of ninety-six characters (upper- and lower-case letters, numbers, and special symbols).

*Printer Mode:* Refers to the text mode (which prints out letters) or the graphics mode (which prints pictures). The IBM graphics printer and most dot-matrix printers can operate in either mode.

*Friction Feed:* The capability of the printer to accept a single sheet of paper (like your typewriter), as opposed to perforated continuous forms. This feature is desirable if you will be printing single letters or using letterhead, but it is not something you can

expect with every printer you purchase. If you want this feature, be sure to ask for it.

*Tractor Feed:* The capability that allows the printer to use different widths of paper. Most low-priced printers do not have tractor feed; they have pin feed, which means you are limited to just one size of paper. Letter-quality printers normally offer tractor feeds as an optional (usually expensive) extra. The option is valuable for letter-quality printers if you produce long reports or documents. Otherwise, you will have to stop the printer and put in a new sheet every time you print a page.

*Single-Sheet Feed:* An add-on printer device that automatically feeds a new sheet of paper for production printing of customized letters or announcements. Single-sheet feeders can cost \$1,500 or more, so unless you need professional word processing, this option is not practical.

*Forms Feed:* An instruction to advance the paper to the next sheet. The standard letter-size paper in the United States is eleven inches long, with a maximum of ten inches used for printing. Legal-size paper is fourteen inches long, with a maximum of thirteen inches for printing. The forms feed instruction can be set to handle either size paper.

*Top of Form:* The command to advance the paper to the first line of a new form, independent of the amount of typing on a form. This capability can really speed up the printer throughput if you have variable-length form printing.

### ***Common Printing Problems***

---

1. Before you begin a printing job, be sure there is enough paper in the machine. Running out of paper is a common problem for novice computer users.
2. If you try printing and nothing happens, the chances are you have forgotten to turn on the power switch to the machine. We have seen people frantically searching for a mechanic when the only thing wrong with the printer was that it had never been turned on. Using a power bar avoids this problem. If the power is on and the printer still doesn't operate, check the cables. They can become loose. Open the printer and look

for possible jams in the paper or ribbon. If none of these measures solves the problem, you may have a serious malfunction.

3. People frequently overuse the printer. Using the printer is expensive and time consuming, so avoid the temptation of re-printing every time you make a minor change. A simple annotation on the form probably is sufficient until you get the final version. Or printing just the corrected portion of your document may meet your needs.
4. Keep your printer clean. Paper generates a lot of dust and dirt inside the printer. A gentle vacuuming (at least after every five hours of service or when you run out of paper stock) definitely is recommended to increase the life span of this expensive and important part of your computer system.
5. Ribbons can be re-inked or reloaded. You can save a substantial amount of money by re-inking your ribbons or reloading your ribbon cartridges. Normally, this costs about half the price of replacing the entire cartridge, and the results are just as good. Ask a local data processing or office supply store for details about this money-saver.
6. Buy paper in bulk. Bulk paper can save you as much as four times the price of paper purchased on a per-sheet basis. Paper keeps indefinitely if it is stored properly (in a clean, dry area). Damp basements can ruin paper, so if storage is a problem, or if you do very little printing, buying bulk paper may not be worth the savings.

## Other Word Processing Options

Two other important word processing functions deserve to be mentioned: spelling verification and mailing list merging. These require separate programs, only one of which (WordProof) is currently available for the PCjr. A complete discussion of these programs is beyond the scope of this book, but you should be familiar with what these program packages can do for you.

Spelling verification programs like WordProof contain dictionary files, against which every word in your document is checked. If a word in your file does not match anything in the dictionary,

the program assumes you have spelled the word incorrectly and "flags" it as an error. Once the program has examined your entire document, you quickly run through the flagged words, changing them as necessary.

If you do a lot of writing in a specialized area—such as medicine or computing (two examples that are close to home for the authors)—you may use words that are not listed in the dictionary (like "agammaglobulinemia"). The spelling checker will presume that these words are misspelled because they are not found in its dictionary.

Most spelling verification programs provide you with "updating" capability. This means that, as you go through your flagged errors, you can designate which words were correctly spelled and should be added to the dictionary. Then, the next time you go through the spelling check on that file (or on your next document about "agammaglobulinemia"), the verification program will accept the word as correct.

Generally speaking, spelling verification programs have large dictionaries (usually over 100,000 words) and require quite a while to run through even a moderate-sized file. Still, the power and thoroughness of these computerized proofreaders is a major addition to your word processing capabilities. Full-time writers will find it hard to get along without one of these for their computer.

If your computerized word processing projects will include frequent preparation of form letters, then a mailing list merging program is for you. These programs allow you rapidly to merge two files—an address file and a document file (which contains your form letter minus the name and address)—so that you print out personalized form letters. Some programs even will address your envelopes as well.

Mailing List Manager is the name of the first mailing list merging program to be announced for the PCjr computer. EasyWriter is compatible with this merging program. In addition, Mailing List Manager "stands alone," which allows you to create your own mailing lists or to retrieve only certain names from a larger list. Mailing List Manager also is capable of printing custom-designed labels based on the information contained in the address files. If form letters and business correspondence will be a big part of your PCjr workload, Mailing List Manager deserves strong consideration.

## Other Word Processing Programs

Some of the most popular word processing programs ever written are not available for the PCjr at the present time. The main reason for this is that most of these programs were designed to operate using two disk drives and a lot of memory. These programs include WordStar, Volkswriter, and Word.

These programs are somewhat more sophisticated than EasyWriter, with larger numbers of commands and more menus to choose from. Other features of some of these programs include continuous reformatting (the margins are kept continuously in line without the need for using a realign command) and the ability to add different instructions for different types of documents. This latter capability means that you can predesign a format for all your letters—such as single spaced, today's date in upper-right corner, and paragraphs not indented. Every letter you write will automatically be set up the same way. A second predesigned format might be used to write contracts, a third to write term papers, and so forth.

As the number of PCjr users grows, pressures will increase to modify these more sophisticated (and generally more expensive) packages for use on the PCjr. Then, of course, will come the temptation to buy a new word processing program just to keep up with the Joneses'. It may be that a function provided by one of these other packages will be absolutely critical to your success and happiness, but probably not.

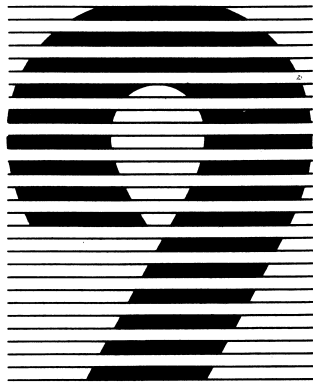
The best advice we can give you is to repeat what we said in Chapter 7: Make a list of the things you can't live without and the things that are desirable but not absolutely necessary. See if there are any items from column A ("must haves") that the new program offers. If so, it's probably for you. More likely, however, it's column B where the difference lies between what you have now and what you're thinking about. Ask yourself if these added features are worth the expense and effort of purchasing and mastering a new system. And above all, try out the new contender in the showroom before you buy it.

Using a computerized word processing system is one of the best ways your PCjr can increase your personal productivity. Use common sense and good judgement, and you will get extraordinary returns on your word processing investment.



## Further Reading

- EasyWriter Manual, Version 1.10* (Information Unlimited Software, Inc.): Explains the use of EasyWriter; includes a five-lesson tutorial.
- The Guide to Personal Computer Offerings from IBM* (IBM Corporation). The guide lists compatible word processing programs available for the IBM Personal Computer product line.
- Office Automation*, by David Barcomb (Digital Press, 1981) and *Office Automation*, by Dimitris Chorafas (Prentice-Hall, 1982). Both books look at word processing in the large context of office automation. They show the next step beyond typing letters, homework assignments, and memos on your PCjr.
- Word Information Processing*, by Walter Kleinschrod, Leonard Kruk, and Hilda Turner (Bobbs-Merrill, 1983). A formal review of word processing aimed at the office environment.
- The Word Processing Book*, by Peter McWilliams (Prelude Press, 1982). Much of the material is for people in search of a computer, and the information on equipment selection is out of date. But this was one of the most successful books on word processing when it was first issued.



## Spreadsheets, Or How To Play The Numbers Game

Your new IBM PCjr computer is a powerful tool that can help you in many ways. You've already seen some of the entertainment value of the games that can be played on the PCjr. In the last chapter we looked at word processing, one of the major applications of home computers today. In this chapter, and in Chapter 10, we will explore the two most widely used "personal productivity" applications that can be done on a home computer. After reading this chapter, you'll see how you *can* play the "numbers game."

### What "Personal Productivity" Means To You

"Personal productivity" programs allow you to do everyday tasks or sophisticated business reports faster and easier than you could by hand. The essence of the concept is the "personal" aspect. That means it's *you* who decides what you want done, and the program is flexible enough to accommodate you. Examples of personal productivity tasks include balancing a checkbook or keeping a budget, calculating tax returns, filing recipes, and compiling sales reports.

Personal productivity software can be divided into two major groups, depending on whether numbers or words are the major focus. For checkbook balancing, tax work, or sales figures, the mathematical aspects of the task are more important than the words involved. *Spreadsheet* programs have evolved to handle these types of projects.

The second major group of personal productivity software deals more with manipulating words than numbers. Recipe files or personnel rosters are good examples of this category. For these applications the computer is called on to sort through files and pick out only the desired information, and perhaps even to compile a report based on that data. *Data base management* programs are the home computer's response to your needs in this area. In Chapter 10 we will look at some of these popular programs.

## Is Using A Computer Always A Good Idea?

One of the biggest—and most common—mistakes in the computer world is trying to computerize a poorly organized and maintained record-keeping system with the hope of improving it. The result is a computerized, poorly organized and maintained record-keeping system—albeit a faster system, but one that costs a lot of money and is of no use to anyone.

Remember, for the computerized system to keep paying its way, you must be as compulsive about updating and maintaining records as you were before you got your computer. The drawback of computerized record keeping can be summed up neatly in a single phrase from computer jargon: **Garbage In, Garbage Out**, or **GIGO** as it is commonly called. A record-keeping system is only as good as the records that are entered into it.

Now for the good news. Computerization *can* help if you already are keeping adequate records. In that case the extra speed, neatness, and accuracy of computer-maintained records can more than pay for itself. But expect to spend almost as much time on record maintenance after you begin using the computer as before. If you are willing to put in this effort, computerizing your records can be a major means of increasing personal productivity. And isn't that why you bought the computer in the first place?

## A Closer Look At Spreadsheets

In Chapter 7 we introduced the concept of the spreadsheet—the computerized ledger book. Let's move in for a closer look.

A spreadsheet is made up of columns (usually designated by letters of the alphabet) and rows (designated by numbers). At each intersection of a column and row is an *entry position* or *cell*, which

is referred to by its coordinates. For example, the first cell in the upper left-hand corner would be A1 (column A, row 1). The 30th cell from the top in the 26th column would be designated Z30. So far, so good—but what about the 27th column? After all, we said earlier that spreadsheets could be hundreds of columns wide.

Columns beginning with the 27th are designated by a two-letter code beginning with AA. Thus, AB would be the 28th column, BA would be the 53d, and ZZ would be the 676th. So, if you can imagine the biggest spreadsheet on earth, cell ZZ10000 would be the 10,000th cell down in the 676th column. Fortunately, most spreadsheets don't get that big, but you get the idea.

Each cell can hold one of three types of data, and this ability is what gives the spreadsheet its power. The first type of entry is called a *label*. Examples of labels are: INCOME, %INTEREST, SALES, TOTAL, and NET PROFITS. Labels have no inherent significance to the computer, and are ignored when calculations are performed. The importance of labels lies in their use to name other cells so that we mere humans can keep track of what data is in which cubbyhole.

The second type of entry is referred to as a *value*. Not surprisingly, values are numbers—numbers representing the value of the data that we just referred to with a label. So, cell A1 may hold a label "SALES," and cell A2 may contain the value of the sales, \$400.00. Any numerical data is considered a value, whether it's money, a percentage, the number of items sold, or the year 1984.

The third type of entry for a spreadsheet is called a *formula*—and herein lies the power of the spreadsheet. Formulas can be simple or complex and can refer directly to values themselves (such as "1 + 2") or to the addresses of cells that contain the appropriate value (for instance, "B1 (account balance) + C1 (interest) - D1 (checks written) + E1 (deposits)" in a checkbook-balancing spreadsheet).

Some formulas are used so frequently that they are abbreviated into function commands. These commands, such as SUM, AVERAGE, and SORT, are used as shorthand in a cell to represent a longer formula. More than one function may be used in a formula cell, as in the AVERAGE of two SUMS.

Because formulas can refer to cell addresses (not just the values themselves), the spreadsheet can constantly update itself whenever the value in a given cell is changed. Let's see what this could mean to you.

Say you have a spreadsheet to keep track of profits made by selling PCjr computers. The values might include the number of units sold per month, the cost per unit, the dealer cost, and your overhead and expenses. The formula to calculate profits would subtract the dealer's costs and expenses from the total value of PCjr's sold.

Now, if your spreadsheet is set up properly, you can begin to play "what if" games. These games go something like this: "What if I sell 20 percent more computers?" Or "What if I decrease my overhead by \$1,000 dollars a month?" Or even "What if I hire another salesperson at \$1,800 a month? How many computers would that person have to sell for me to increase my profits by \$3,000 a month?" In seconds the spreadsheet can give you the answer to these and far more complicated "what if" questions.

## What Can Spreadsheets Do For You?

How many of your daily tasks can be turned over to the computer, with spreadsheet programs to carry the load? Here are some real-life situations in which spreadsheets *can* be useful. All of these jobs routinely are being handled by spreadsheet programs in households not unlike your own. And none of these applications requires sophisticated techniques or years of practice and expertise. As you'll see, spreadsheets can be worth their weight in gold, if you recognize the vast potential of this valuable resource.

*Budget Control*—One of the classic uses for spreadsheet programs is budget planning and control. With a spreadsheet you can enter a base budget for a single month and then make assumptions in terms of percentage growth or percentage increase in expenses for the remainder of the year. All of these formula-based projections can be automatically calculated by the spreadsheet program and entered into the appropriate monthly column.

As actual dollars are entered, the spreadsheet can automatically calculate the variance between budgeted and actual amounts. Your entire monthly budget can be handled with a few strokes, once you have set up this easy-to-use electronic spreadsheet.

*Personal Finance*—At a time when you must make your money really work for you, the electronic spreadsheet forms the basis for an excellent investment manager. Stocks, dividends, price-to-earn-

ings ratios and other key metrics about your current and planned investments can be entered into your spreadsheet, enabling you to monitor your stocks, bonds, real estate, and even art collection in terms of current value and anticipated return on investment.

*Your Spreadsheet Calorie Counter*—By entering in the typical caloric content and nutritional information about the foods you eat, you can develop your own meal plans and menus, based on your personal likes and weight-control goals. A spreadsheet format makes an excellent representation for tables showing calories versus serving sizes and including other elements such as food variety.

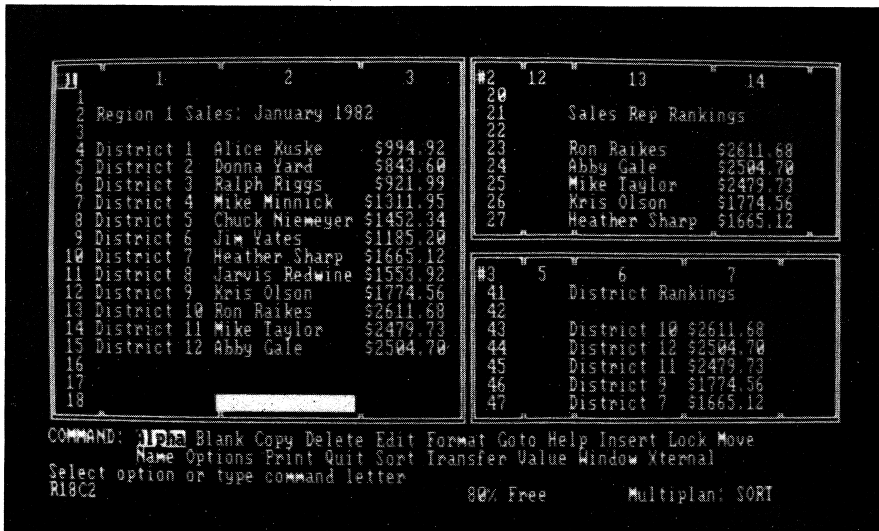
*Energy Management with Spreadsheets*—Your local electric company, through its consumer relations department, will offer you a detailed energy consumption table for your heating system and each of your appliances. By entering these values into your spreadsheet, along with your expected hours of usage for these household helpers, you can create an efficient energy management plan for your home. Then, by comparing the actual consumption with your projected consumption, you can institute ways of controlling this expensive resource. By the way, be sure to include your automobiles and any other vehicles in this energy management plan.

*Spreadsheets and Sports*—Little Leagues, bowling leagues, Pop Warner football, soccer leagues, and even joggers run on numbers. The spreadsheet is an excellent way to perform all those calculations needed to keep track of batting or bowling averages, miles or kilometers jogged per day, and the myriad of other statistical values required in analyzing sports performances.

Do these suggested applications give you ideas for increasing your personal productivity? If so, read on. In the next section we will discuss the spreadsheet programs currently available for the PCjr. Then we will focus on one—VisiCalc—to show you just what a good spreadsheet program can do for you.

## **Spreadsheets—What's Out There?**

Two main spreadsheet programs are featured for use with the PCjr: VisiCalc 1.20 and MultiPlan 1.10. VisiCalc was the original spreadsheet program and is still one of the simplest and most



MultiPlan screen display. MultiPlan is an electronic spreadsheet program that uses "windows" to display information.

efficient to use. MultiPlan features more sophisticated screen displays, such as colored fields and multiple "windows" or views of the same data.

For most users there is little difference between MultiPlan and VisiCalc when it comes to actually using the programs. The slightly more straightforward command structure of VisiCalc has led us to choose it as our model spreadsheet for this chapter.

Another name deserves mention, however, because of its popularity with other personal computers (such as the IBM PC)—Lotus 1-2-3. Lotus 1-2-3 is one of the most popular and powerful programs ever written, but it requires a second disk drive and more memory than is currently available on the PCjr.

Lotus 1-2-3 is an integrated spreadsheet, data base, and business graphics package. That means that it lets you not only produce a spreadsheet for numerical data but also create, store, and retrieve *any* data, as long as it can be organized in tabular format. The sophisticated graphics commands available with 1-2-3 allow you to take any of your data and quickly create graphs and bar or pie charts. Other spreadsheet packages require one or more additional programs to perform these three functions.

Even though 1-2-3 is not now available for the PCjr, we predict it won't be long before the demands of the PCjr market spawn a version of this spreadsheet program for your computer.

Back to the spreadsheets that are available, where once again, you—the PCjr owner—find yourself faced with a variety of choices and little in the way of hard data to sway you one way or the other in deciding what software to buy. Our advice to you remains the same. Decide what you really need (including the question of whether you really need a spreadsheet program at all) in advance—before you set out to shop for one. Don't be taken in by gimmicks or "sizzle" instead of substance. And above all, try the program before you buy it.

## Getting Started With VisiCalc

If you purchased a copy of the VisiCalc 1.20 spreadsheet program when you bought your PCjr, now is the time to take it out of the box. If you do not own VisiCalc, follow along with our examples and exercises anyway. As you will find out when you examine them more closely, most spreadsheets operate in fundamentally the same manner, and experience with one program will make any of the others easier to master.

The first thing you should do with your new copy of VisiCalc is transfer the operating system onto your program diskette so the program will be self-booting. To do this, boot DOS 2.10 and type:

**SYS B:**

then press [ENTER]. Now insert VisiCalc (be sure you have removed the write/protect sticker) and strike any key to begin. When you receive the next DOS prompt

A>

type:

**COPY A:COMMAND.COM B:**

and press [ENTER]. Replace DOS in the disk drive and strike any key. When requested, switch diskettes again (so that VisiCalc is in



the drive) and strike a key. When the DOS prompt reappears, remove VisiCalc and replace the write/protect sticker. VisiCalc is now ready to run without the need to load DOS beforehand.

Before you can use VisiCalc, however, you must be sure it is configured (adjusted) for the type of display screen you have. If you plan to use the standard eighty-column, high-resolution monochrome or color display, VisiCalc is all ready to use as is. If you will be using a forty-column display (such as a television set), reinsert VisiCalc in the disk drive and type:

### VCCONFIG

and press [ENTER]. You will be asked to remove the write/protect sticker. Then type:

40

and press [ENTER]. VisiCalc is now configured for use on a forty-column display, so replace the write/protect sticker and let's go.

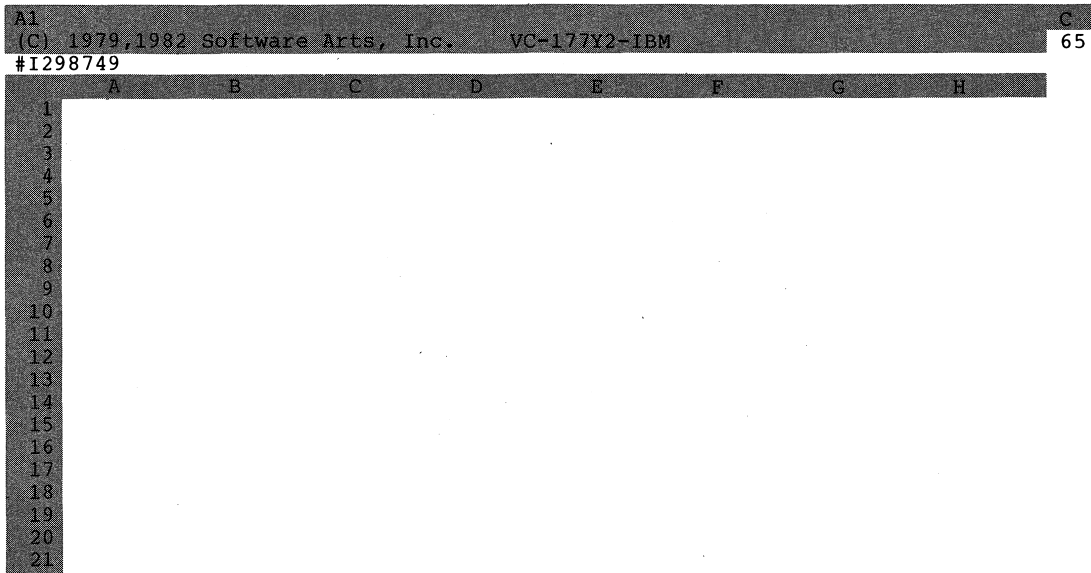
Regardless of which type of display you are using, you should now be ready to begin. Reinsert your VisiCalc diskette in the drive and warm boot it using the [CTRL], [ALT], [DEL] combination.

### Introducing The VisiCalc Spreadsheet

You should now see the basic framework of the VisiCalc spreadsheet on your screen (Figure 9-1). The most noteworthy and important features are the row and column markers. As you learned earlier, the columns are designated with letters, and columns A through H are displayed on the screen (if you have the standard eighty-column display). Moving down the left-hand border of the screen are the row markers, with rows one through twenty-one visible.

Notice the first cell—A1—in the upper left-hand corner. Like the row and column markers, this cell is backlighted (highlighted). This is the cell currently containing the VisiCalc cursor. Note that unlike EasyWriter, DOS, or BASIC (whose cursors indicate only a single character), the VisiCalc cursor occupies the entire cell.

Now look at the top of the screen. The first two rows are also backlighted. The top row is called the *entry line*. At present, this line says only A1 on the left (coincidence, perhaps?) and C on the



**Figure 9-1** The VisiCalc spreadsheet—a tool for managing numbers.

right. The C is an indicator of the method VisiCalc will use to perform its calculations (column first or row first). The A1, however, as you've probably guessed, designates the cell in which the cursor is located at the moment. Any data entered in the cell would be displayed on the entry line. Because cell A1 is blank, so is the rest of the entry line.

Immediately below the entry line is the *prompt line*. Right now the VisiCalc copyright information and version number are displayed on it. When you are actively using VisiCalc, this line will indicate any command options that are open to you, or it will prompt you for a response—much like EasyWriter's various menus did.

At the end of the prompt line is a number that is not highlighted. This indicates the amount of memory space currently available in your spreadsheet. VisiCalc will use all the memory in your computer that is not already occupied by itself or DOS to form your spreadsheet, so you should have plenty of room. Just the same, if you are working on a very large file, keep your eye on the number to avoid losing valuable data.

Last is the *edit line*, between the prompt line and the column markers. During data entry the information to be stored will be kept here while it is typed into the system. VisiCalc provides a second cursor for the edit line. This one takes up only one character and lets you see exactly where you are within the cell. Currently, your VisiCalc serial number is being displayed in this area. Write it down now (inside your VisiCalc manual), as you would the serial number of any valued piece of property.

When you use VisiCalc, you won't want the distraction of the serial number and copyright, so let's get rid of them now. Simply press [ENTER] and watch them disappear. Your screen should now look similar to the one depicted in Figure 9-2. Although the serial and version numbers have vanished from your screen, they are not gone forever. You can get them back by typing /V—the Version command.

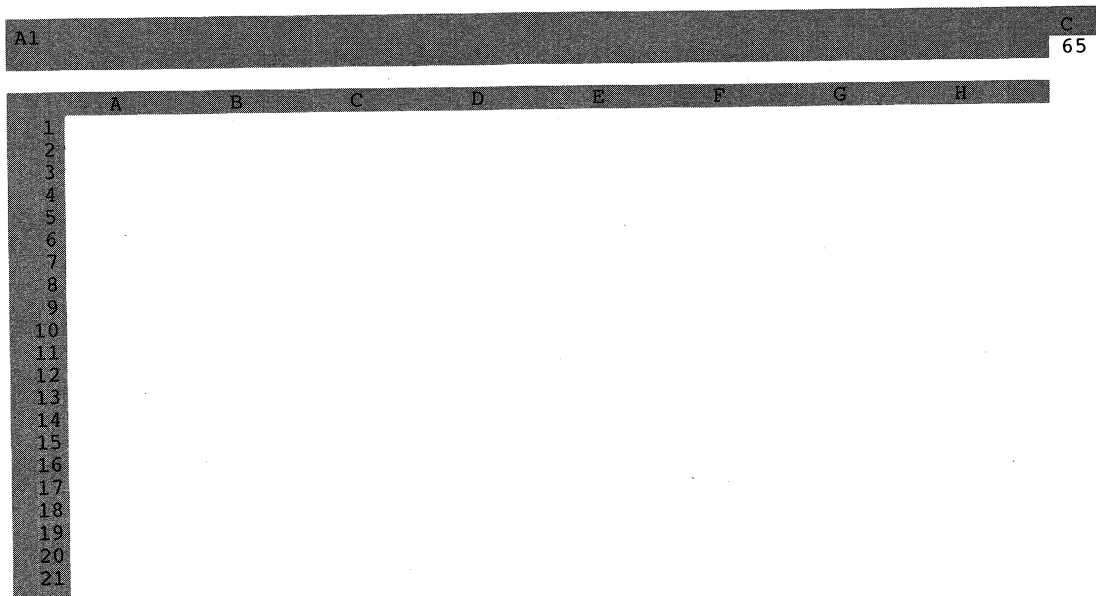
Let's explore more of the spreadsheet than just the corner. You may remember that we boasted about how big spreadsheets could be. The spreadsheet we see on the screen right now—eight by twenty-one cells—just isn't that impressive. Use the [→] key to travel to the right-hand edge of the screen with the cursor. Now, go beyond the edge and watch what happens. The screen *scrolls* along with you, exposing new columns of your spreadsheet.

Basically, the screen acts only as a window looking onto a small section of a larger work area. We can move this window around whenever we want to look at any part of the work area, but we can never see all of the spreadsheet at once. (In actuality, we can split the screen into two separate windows—much like the split-screen TV coverage sometimes used in baseball games—but we won't deal with that advanced topic in this book.)

So just how big is this VisiCalc spreadsheet, anyway? Press down and hold the [→] key until it won't go any farther, and prepare to go for a ride. Eventually, you should reach VisiCalc's equivalent of a brick wall, at column BK! That's sixty-seven columns wide. Not bad—certainly more than we can ever think of using up.

If you feel adventurous and you've got time on your hands, you can play the same game with the [↓] key until you reach the bottom. But there's a better way to travel around the spreadsheet, especially when you're in the outer limits like BK1.

To facilitate moving around the spreadsheet, VisiCalc has a Go To command. Press the [>] key and notice what happens. For



**Figure 9-2** The VisiCalc spreadsheet, minus the serial number and copyright notice. Eliminating these items gives a cleaner screen.

starters, the prompt line (line 2) indicates you have invoked the Go To command and asks you for the coordinate you would like to access. On the line below (the edit line) VisiCalc conveniently has provided a cursor to show you where to enter the coordinate. Climb aboard the express, and we'll head for the farthest reaches of the spreadsheet. Type:

**BK254**

and press [ENTER]. When you've reached BK254, try using the [→] and [↓] keys. No good, are they? You've reached the farthest corner of the spreadsheet. Lonely, isn't it? Let's get back to the friendly confines of our starting point—cell A1.

You now know of two ways to return to A1. You can use the [↑] and [←] keys, but that's the slow way. Or you can use the Go To command and specify cell A1. As it turns out, cell A1 is such a

popular place to go that VisiCalc has given us yet another way to get there. Because A1 is the home position to VisiCalc, you can reach it by just using the [HOME] key (the combination of the [FN] and [↑] keys). As quickly as that, you're back where you began.

Use these three techniques to move freely around the spreadsheet. In particular, try using the Go To command to reach a cell that doesn't exist, and see what VisiCalc does. VisiCalc has a rather uniform reaction to impossible or erroneous commands: It ignores them. When you are fully comfortable navigating through the spreadsheet, we can move ahead.

Let's take a quick review first.

1. VisiCalc is a spreadsheet program, composed of a matrix of cells 67 columns wide by 254 rows long. That's a total of just over 17,000 possible data entry positions. Each cell is addressed by coordinates. The column position is represented as a letter or letters from A to BK, and the row position is designated as a number from 1 to 254. Examples of cells are A1, BK254, K93, R100, and AS212.

2. The display screen is merely a window looking onto the larger VisiCalc work area. The entire spreadsheet cannot be seen at one time. However, any part of it can be reached and observed with the screen.

3. There are a number of ways to move the VisiCalc cursor around the spreadsheet. The four cursor control keys (arrows) move the cursor one cell at a time in any direction. When the cursor reaches the end of the spreadsheet itself, it can go no farther. If it is merely at the edge of the screen, however, the cursor will move right on, taking the screen with it (a process called scrolling). The Go To command (accessed by the [>] key) is a magic carpet to any cell you designate. Finally, the [HOME] key takes the cursor instantly back to A1—home again!

4. In addition to the matrix of cells, the VisiCalc display includes three lines at the top of the screen: the entry line, the prompt line, and the edit line. The entry line keeps track of the cell location of the cursor; the prompt line displays available options and asks for information (when necessary); and the edit line is like a magnifying glass, focusing on what's happening inside one single cell. In the upper right-hand part of the screen, a number indicates the memory space available for the VisiCalc spreadsheet.

## Let's See What VisiCalc Can Do—A Sample Problem

At the beginning of this chapter, we mentioned ways in which a spreadsheet program might help the owner of a small computer store. Let's pay a visit to the VisiCalc Computer Store and see exactly what's going on. Working through this example will give you experience with some of the VisiCalc commands and procedures.

Bring your VisiCalc cursor back to the home position (A1) any way that seems convenient. Your spreadsheet should be completely blank. If not, type:

**/CY**

to clear the sheet.

The hypothetical problem we initially proposed involved keeping track of profits generated by computer sales. Let's suppose the VisiCalc Computer Store has just received its first shipment of IBM PCjr computers, and they're selling like hotcakes. They're selling so fast, in fact, that the boss wants a complete report on the profits you've made this past month and has asked you to determine if adding another salesperson to the payroll will be worthwhile.

To tackle this problem, let's review some of the concepts you have already learned about using VisiCalc. Moving the cursor should still be fresh in your mind. But think back to the discussion of the three types of entries to a spreadsheet: labels, values, and formulas. We will be using all three in this example.

The sales data the boss just gave you includes the number of PCjr computers sold, the cost of each one (both to the customer and to the store), and the overhead incurred in selling PCjr's, which includes advertising, sales personnel and salaries—no commissions in this store—and so on. From this information you will have to calculate the gross income and the net profit from PCjr sales.

For starters, let's put the appropriate labels in the first column of the spreadsheet. Start in A1 by typing the label:

**NO. SOLD**

(We will use all capitals in this exercise, but VisiCalc treats capital and small letters identically, as you know.) Watch what happens on

the entry line (top line) and the edit line (third line) as you type. No data is entered permanently into a cell until one of two things happens—either you press [ENTER] or you move the cursor to the next cell with one of the cursor control keys. This second option makes data entry more convenient, as you will see.

Press the [↓] key and move the cursor into A2. Again, take note of the entry line, edit line, and now the label stored in A1. Enter the following labels in cells A2 through A5:

**COST PER, GROSS, DEAL COST, OVERHEAD**

Use the [↓] key to enter each label. Move down to cell A7 and enter the label

**PROFIT**

Then press [ENTER]. The screen should look like Figure 9-3. Take a look at the entry line now. It displays the address of the cell currently containing the cursor (A7) and the contents of the cell: PROFIT. It also says (L). What does this mean? This is nothing more than VisiCalc's notation that cell A7 contains a label—not a value or formula. So far, so good.

Now, move the cursor to cell B1 using the [>] key (Go To command). Enter the appropriate sales figures next to the labels you have just entered. In cells B1, B2, B4, and B5, type:

**NO. SOLD 80**  
**COST PER 1500**  
**DEAL COST 1000**  
**OVERHEAD 12500**

respectively. Did you leave B3 (GROSS) blank? You should have, because we will use VisiCalc to calculate that value. If you mistakenly entered the value for cell B4 in B3, return the cursor to B3 and type /B (the Blank command). Then reenter the correct figures in the appropriate cells.

Notice the entry line for a cell containing a value. It contains the VisiCalc message (V). Also notice that the labels you entered in column A were all aligned to the left, but the numbers in column B were all aligned to the right. This is standard operating procedure

A8								C
Label								65
%GROSS	A	B	C	D	E	F	G	H
1	NO. SOLD							
2	COST PER							
3	GROSS							
4	DEAL COST							
5	OVERHEAD							
6								
7	PROFIT							
8	%GROSS							
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								

**Figure 9-3** Labels entered on the spreadsheet.

for VisiCalc, but (as you will see later) it can be modified to suit your specific needs.

The preliminaries are now out of the way, so let's start calculating. We need a formula to determine our gross income from *PCjr* sales. This isn't too difficult. The gross income is derived by multiplying the number sold by the cost per unit. In this specific case, the value for the number sold is contained in cell B1, and the value for the cost per unit is contained in cell B2. So in VisiCalc shorthand, the gross income would equal B1 times B2. VisiCalc uses the same four mathematical operators that BASIC used—namely, + - \* / for add, subtract, multiply and divide. Go to cell B3 and type:

**B1\*B2**

and then press [ENTER].

Wait a minute. VisiCalc didn't calculate anything! It just put B1 \* B2 into cell B3 (see Figure 9-4). The entry line, however, tells the story. VisiCalc interpreted the entry as a label (L), not a formula.



B3								C
Label	A	B	C	D	E	F	G	H
B1*B2								65
1	NO. SOLD	80						
2	COST PER	1500						
3	GROSS	B1*B2						
4	DEAL COST							
5	OVERHEAD							
6								
7	PROFIT							
8	%GROSS							
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								

**Figure 9-4** Formula entered on the spreadsheet.

VisiCalc decides what type of entry it is about to receive on the basis of the *first* character of the entry. If the first character is a letter, then VisiCalc interprets the entire entry as a label. If the first character is a number, VisiCalc considers the entire entry to be a value—even if you planned it to be a label, like 1ST QTR SALES.

To explain to VisiCalc that an entry that starts with a number is really a label, begin the entry with a quotation mark: "1ST QTR SALES. The quotation mark will not appear in the cell, but it will show up on the entry line. To use quotation marks in a cell, you must double the first quotation marks: as ""QUOTE."

Now that we know how to make a number into a label, how can we make a label into a formula—our original problem? Begin the entry with a plus sign, and VisiCalc will consider the letters and numbers that follow to be the values of the cells they specify. Go back to cell B3 and type:

**+ B1\*B2**

then press [ENTER]. VisiCalc should now have calculated the gross income and displayed the value in cell B3. Notice, however, that

	A	B	C	D	E	F	G	H
1	NO. SOLD	80						
2	COST PER	1500						
3	GROSS	120000						
4	DEAL COST							
5	OVERHEAD							
6								
7	PROFIT							
8	%GROSS							
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								

**Figure 9-5** Calculation performed by VisiCalc.

the entry line still displays the formula from which that value was derived (Figure 9-5).

You can now calculate the profit made for the month. In this example the profit would equal the gross income (B3) minus the dealer's cost for all the computers sold (B1 \* B4) minus the overhead (B5). Enter the formula

$$+ B3 - (B1 * B4) - B5$$

into cell B7. B7 should now display the total profit made this month from PCjr computer sales (\$27,500).

Let's go one step farther and calculate the profit as a percentage of the gross income. This figure is obtained by dividing profit (B7) by gross income (B3) and multiplying by 100 (to get percent). Type the label

**% GROSS**

in cell A8. In cell B8 enter the formula

$$+ B7 / B3 * 100$$

The value 22.91667 should now be displayed in B8. That's a very precise value—far more precise than you're ever likely to need. In this case, the appearance of the spreadsheet would be enhanced if you displayed the percentage with only two decimal places, instead of five.

Changing the decimal places will involve our first in-depth look at VisiCalc commands. Before we go into that, let's quickly review.

1. VisiCalc recognizes three types of cell entries: labels, values, and formulas. It bases recognition of these types on the first character of the label. If the first character would lead to an erroneous assignment, special characters are added to designate the type of cell entry. To fix an entry as a label, precede the first character with a quotation mark. To fix an entry as a formula, precede the first character with a plus sign. Values must be numbers. Unless otherwise specified, labels are aligned to the left and values are aligned to the right within the cell.

2. The entry line will indicate whether a cell contains a label, value or formula. Even if the calculated result is displayed inside the cell, the entry line will always display the actual formula from which the value was derived.

3. Formulas can specify the address of the cell(s) that contains the desired value(s) and will calculate the result based on the current value of the designated cell(s). The arithmetical operators used in VisiCalc formulas are + - \* / (add, subtract, multiply, divide). Parentheses may be used within a formula to specify the order of calculation.

### More Work For VisiCalc

Now, let's address ourselves to the problem of adjusting cell B8 to display fewer decimal places. The cursor should be in cell B8, and the edit line should be clear.

We already have been formally introduced to one VisiCalc command, the Go To command. This command was invoked by pressing the [>] key. Most VisiCalc commands are invoked by pressing the [/] (slash) key. Strike the [/] key now. The prompt line displays a list of the available commands. They are listed in alphabetical order (by their first initial) in Figure 9-6. To invoke a specific com-

	A	B	C	D	E	F	G	H
1	NO. SOLD	80						
2	COST PER	1500						
3	GROSS	120000						
4	DEAL COST	1000						
5	OVERHEAD	12500						
6								
7	PROFIT	27500						
8	%GROSS	22.91667						
9								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								

Figure 9-6 VisiCalc commands listed on the screen.

mand, you need only type [/] followed by the first letter of the command name. The accompanying checklist provides a listing of all the VisiCalc commands available in this mode.



## VisiCalc Commands

Command	Action
B	Blank—Deletes contents of highlighted portions
C	Clear—Deletes contents of spreadsheet
D	Delete—Erases row or column
E	Edit—Permits editing
F	Format—Establishes local formats
G	Global—Modifies entire sheet

(continued)

---

<b>Command</b>	<b>Action</b>
I	Insert—Adds row or column
M	Move—Transfers row or column
P	Print—Sets up printing
R	Replicate—Allows repetition
S	Storage—Interacts with files
T	Titles—Establishes the boundaries and parameters of columns and rows
V	Version—Shows copyright, version, and serial number
W	Window—Divides screen for multiple viewing

---

Because we wish to change the format (appearance) of cell B8, we will select F for the Format command. Type:

**F**

and see the Format command options displayed on the prompt line—again, abbreviated by a single letter or symbol. Currently, the entire spreadsheet is in General format, which calls for maximum precision. Thus the screen will display as many decimal places as will fit in the cell. Another alternative would be Integer format, in which all decimal places are dropped from the display. (Have no fear, VisiCalc internally keeps track of all decimal places—at least eleven significant figures—regardless of how many decimal places appear on the screen.)

The format option we desire is to display just two decimal places. Because this is the way monetary values most typically are displayed, VisiCalc abbreviates this format with the symbol \$. Press the [F\$] key (remember to shift) and watch what happens to cell B8. The result is displayed as 29.92, just as we wanted it. The entry line now shows /F\$ next to the B8, indicating that the cell has been formatted to a special setting.

You've done a lot of work to get to this point, so let's save this spreadsheet onto a diskette. Remove your VisiCalc program diskette from the disk drive and replace it with a *formatted*, blank diskette. Type:

**/S**

for the Storage command. The storage options are now displayed on the prompt line. The storage option you need is the Save option, S. Type:

**S**

and the prompt line will ask File for Saving, and the cursor will be in the edit line. Type:

**VCEX1**

and press [ENTER] to record the new file name. The entire spreadsheet will now be saved onto your diskette.

The filename conventions for VisiCalc are similar to those for DOS but slightly more restricted. Names can be from one to eight characters long and can be composed of letters or numbers (no other characters). The first character of every VisiCalc filename must be a letter. Filename extensions cannot be specified, as VisiCalc automatically assigns the extension .VC to every file.

Now that the file is saved, we can feel free to do some "what if" calculations. The original statement of the problem included the hypothetical question, "Would it be worthwhile to add another salesperson?"

Imagine that the new salesperson will receive a monthly salary of \$2,000, which will add to our overhead. If he or she increases PCjr sales by an additional twenty computers per month, how will that affect profits?

Currently, our spreadsheet analysis indicates that we are making \$25,500 profit per month, or a healthy 21.25 percent of our gross income. Go to B1 and enter the value 100, which is twenty more PCjr's sold than we originally calculated.

Now go to B5 and increase overhead by \$2,000 to \$14,500. We could just reenter the number 14500, but because we really need to change only one digit (the 2 to a 4), reentering the entire number seems a shame. VisiCalc has an Edit command that will allow us to change only a part of the contents of a cell, without affecting the remainder. Type:

**/E**

to invoke the Edit command. The contents of cell B5 now appear on the edit line, along with the edit cursor. Move the cursor to the 5. Now, use the [DEL] key to delete the 2; insert a 4 in its place. Press [ENTER] to leave the edit mode and to enter the new value into the spreadsheet.

If you make an error while you are entering data into a new cell, you can invoke the Edit command in midentry by the combination of the [CTRL] and [E] keys ([CTRL E]).

Now check the profit figure in B7. The new figure for profit is \$35,500, or 23.67 percent of the gross income. You can confidently suggest that adding another salesperson at \$2,000 a month will be productive, if that person can sell twenty computers a month. In fact, with a little manipulating of the spreadsheet, you can find out exactly how many computers the new salesperson would have to sell to pay his or her way.

If you wish to send a copy of your "what if" calculations to the boss, type:

**/P**

to invoke the Print command, followed by P for Printer as the specified output device.

To return to the original spreadsheet, first clear the screen with the /C command. The prompt line will ask you if you are sure by saying,

Clear: Type Y to Confirm

Type:

**Y**

for Yes. The spreadsheet should now be blank. You will have to retrieve the other spreadsheet from the diskette on which you stored it. Use the /S command again, but this time select the L for Load option. The prompt line will ask you for the name of the file to load. Here, you can type in the name of the saved file (VCEX1, remember?). Or you can let VisiCalc do the work for you. If you're like us, you'll probably take the latter option. Press the [→] key once, and watch the edit line. The name of the file should come

into view. (If you did not save the file onto a blank diskette, another filename may have appeared. Keep pressing the [→] key until the desired filename appears.) Now, simply press [ENTER] and VisiCalc will load the spreadsheet from the diskette file back into memory and onto the screen. The cursor should still be in the cell where you left it (B8).

Let's review briefly what we've just covered.

1. Most VisiCalc commands are invoked with the combination of the [/] (slash) key and the first initial of the command. Some commands have additional options, which also are specified by the first initial. The Format command, /F, is used to specify the number of decimal places displayed on the screen for a given cell. Regardless of the number of decimal places displayed, VisiCalc always keeps track of at least eleven significant figures in memory, and uses the full eleven figures in all calculations.

The Format suboptions include General, which displays all the decimal places that will fit into the cell; Integer, in which no decimal places are displayed; and \$ (currency), in which two decimal places are displayed.

2. The Storage command, /S, handles storage and retrieval of spreadsheet files onto diskettes. The S for Save option creates a file on diskette, and the L for Load option retrieves it. Filenames can be up to eight characters in length and can be letters or numbers, with the first character always a letter. No filename extensions are permitted, as VisiCalc assigns the extension .VC to all files.

3. "What if" calculations can be done on a spreadsheet simply by manipulating a few values and watching the results. If there is an error in a cell, the contents of that cell can be retyped or changed with the Edit command, /E. A cell that has not yet been entered but contains an error can be edited with the [CTRL E] key combination.

4. The entire spreadsheet can be printed on the printer using the Print to Printer command, /PP.

## Parting Thoughts

This brief exposure to a hypothetical problem gives you an idea of what working with a spreadsheet is like—in this case VisiCalc. Of course, we've only given you a brief sampling of some of the com-



mands you can work with to move figures and manipulate formulas. Electronic spreadsheets give you a wide variety of mathematical capabilities. For example, using the same problem we just worked through, we could have gone much farther. Suppose your boss had liked your PCjr sales report so much that he asked you to expand it to include other computers sold in the store. Using VisiCalc, you could have made major additions to your spreadsheet with little effort.

Almost every profession, from teaching to medicine, has its own particular set of tables, productivity analysis, or planning tools that will easily lend themselves to a spreadsheet. If you keep your eyes and mind open, you will probably save far more than the price of your PCjr and your spreadsheet program in the first six months of application.

The texts we have included in Further Reading not only will give you hints but will give you actual model spreadsheets for a variety of professional and even some home applications. Be sure to take advantage of this free consulting.

There's plenty more to know about spreadsheets, but none of it is any more difficult than anything you've already mastered. And you can probably see, quite clearly now, the ways that a spreadsheet program like VisiCalc can increase *your* personal productivity, even if you don't own a computer store.

## Further Reading

*Doing Business with VisiCalc*, by Stanley Trost (Sybex, 1982). A review of VisiCalc with a number of business examples.

*The Power of MultiPlan*, by Robert Williams (Prentice-Hall, 1982). An excellent set of MultiPlan spreadsheet applications.

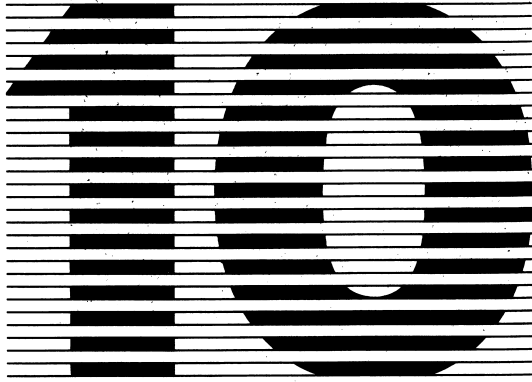
*The Power of VisiCalc*, by Robert Williams and Bruce Taylor (Prentice-Hall, 1982). Contains a series of worked VisiCalc spreadsheets. An excellent application reference.

*The Power of VisiCalc Real Estate*, by Patricia Hughes and Kaz Ochi (Prentice-Hall, 1982). Specifically designed for the use of VisiCalc in real estate applications.

*VisiCalc Guide* (VisiCorp). The manual for the VisiCalc program. It contains a five-lesson tutorial on the use of VisiCalc, as well as detailed references on each VisiCalc command.

*VisiCalc: Home and Office Companion*, by David M. Castlewitz and Lawrence Chisausky (Osborne/McGraw-Hill, 1982). Worked VisiCalc examples. Most are unrealistic but still serve as teaching aids.

*VisiCalc for Science and Engineering*, by Stanley R. Trost and Charles Pomernacki (Sybex, 1983). Application of VisiCalc to a number of physical science and engineering problems.



# Data Bases, Or How To Have Your Own Electronic File Cabinet

You already have had an introduction to two of the “big three” software applications—word processing and spreadsheets. In many ways the remaining member of the “big three”—data base management—is similar to the other two applications. Just as a spreadsheet made it easier to handle numbers and calculations, so can a data base management program simplify the manipulation of records and forms and make the job faster and more efficient. But because the data base management program manipulates words instead of numbers, it has features in common with the word processing program as well.

## What Is A Data Base?

A *data base*, in the broadest sense, is any collection of information (data) having a common thread. The data in the data base can be arranged in a variety of ways—by name, telephone number, address, job title, date, and so forth. A file cabinet full of folders brimming with paper is a data base. So is a stack of index cards bound together with a rubber band, or a drawerful of old newspaper clippings. It doesn't matter what kind of container holds the data base. All that counts is the information stored inside. This is true of a computerized data base. The exact form of the data stored in it is only of secondary importance.

In a data base, the data is stored on individual forms, which may be more than one page long. The collection of forms makes up a file. Once the form is set up, data is entered directly into fields

on the form. For example, a file composed of the names and addresses of various customers would be made up of individual forms with fields for name, street, apartment number, city, state, and zip code. Each customer's data would be entered onto his or her own form.

Individual items or the entire form can be retrieved from a file, on demand. For example, you may need to search your data base for the address of one customer whose name you already know, or you may want to find the names of all your customers who live in a certain city. Possibly, if you are selling renter's insurance to people who live in Los Angeles, you would search your data base for the names and addresses of people who live in Los Angeles *and* who have an entry listed for apartment number. So, a data base management program can use multiple retrieval specifications to search and retrieve forms.

Once the data has been retrieved, programs vary in what they can do with it. Some are capable of printing high-quality reports complete with headings, titles, and appropriate arithmetic calculations. Others do little more than display the requested data onto the monitor screen.

## **pfs:FILE And pfs:REPORT**

The data base management package available for use with the PCjr is actually two separate programs. One program is designed for entering data into the data base, and the other is used to prepare reports summarizing all or part of the data stored in the data base. The programs are called pfs:FILE and pfs:REPORT, names that summarize what each does. These two programs work hand in hand to accomplish more than a single program could do.

pfs:FILE and pfs:REPORT are marvels of simplicity. They demonstrate quite clearly that one can have powerful programming capability without making the program impossible to use. Both programs are highly menu oriented, which means that at almost every turn the computer displays to you all the available options on some sort of menu. To execute the option you desire, you simply type a number or press one of the function keys ([F1] through [F10]).

Both pfs:FILE and REPORT can operate easily on single disk drive systems. However, there is some loss of power. Specifically,

the Copy and Change Design commands cannot be used on single drive systems. The lack of the copy command is not significant, because the DOS Copy command usually can accomplish the same result. The loss of the Change Design command, however, imposes a slightly more severe limitation, because it means that you must design (from scratch) a new file every time you wish to make even a slight change in an old one. If you will be using pfs:FILE and REPORT heavily, this alone could justify the cost of an add-on disk drive, when they become available.

This chapter will walk you through the highlights of pfs:FILE and REPORT. We're going to use an example of setting up a data base of all the patients in a doctor's office. Working through this example will show the various steps of using a data base—designing a file, entering input on the file, retrieving data from the file, printing forms from the file, and generating reports. If you have pfs:FILE and REPORT, then follow along with the example as it unfolds. If you don't, you can still follow along and see the capabilities of these two programs, as well as their simplicity. This combination of virtues may just be enough to convince you that pfs:FILE and REPORT are programs you want to add to your software collection.

## Designing A Data File

If you own pfs:FILE and REPORT, prepare your programs to be self-booting by using the Install programs as described in the first chapter of each manual. You will also need a blank, formatted diskette. If you do not have one available, use DOS to format another. Our sample data base will contain ten patients. After you have followed the step-by-step instructions for entering the data on the first patient, you will be able to enter the remaining data yourself. (The data for all ten patients is found in Appendix A.)

Start by loading pfs:FILE (either by booting the diskette or—if you are already in DOS and have the A> prompt—by typing FILE). You will now see the first pfs:FILE menu, the Main Menu. This menu lists the seven primary functions of the pfs:FILE program (see Figure 10-1). You will see the Main Menu whenever you load pfs:FILE or press the [ESC] key. Once the Main Menu is displayed, you may remove pfs:FILE and replace it with the blank diskette you previously formatted.

PFS:FILE FUNCTION MENU  
-----

- |   |               |   |               |
|---|---------------|---|---------------|
| 1 | DESIGN FILE   | 5 | PRINT         |
| 2 | ADD           | 6 | REMOVE        |
| 3 | COPY          | 7 | EXIT PFS:FILE |
| 4 | SEARCH/UPDATE |   |               |

SELECTION NUMBER:

FILE NAME:

(C) 1983 Software Publishing Corporation

F10-Continue

**Figure 10-1** The pfs:FILE Main Menu, which lists the primary functions of pfs:FILE.

Our first order of business will be to design a file to hold all the medical information pertaining to Dr. Data's patients. In doing so, we will review the key elements of constructing a data file for any application.

Designing a form is simple with pfs:FILE. Each form is made up of one or more pages (this example will have two pages) that consist of item names and blank fields to accommodate those items. Examples of the kinds of items that go into a data base include names and addresses, social security numbers, recipe ingredients, and so on. Each item is named appropriately for easy recall, hence SSNO might be the item name for the field containing the social security number.

There's no inherent significance to the order in which items are placed within a form. There is, however, one important rule to keep in mind. When pfs:FILE performs a search, it always checks the first item on the form first. If you put the most frequently searched-for item in the first position, your searches will be as speedy as possible. Pay attention to this as you design your own forms, because search time can be considerable for a large file, and any savings will help. In our example Dr. Data most frequently needs to find the records of patients. Therefore, the patient's name is stored in the first field.

Every item name must end with a colon. This is a signal to pfs:FILE that the item name is finished and that every blank space thereafter is to be considered as the "field" into which data later will be entered. The field ends whenever the next item begins. More than one item name can be entered per line, or one item name can have several lines as its field. Plan ahead, though, so there is always enough room to contain the largest possible data entry.

As an example, if the item name is STATE: you'd need to determine in advance whether you will use the two-letter postal service abbreviation for the state name or the full name of the state. Obviously, if you choose to enter the whole name, you will have to leave enough room for names like California and Massachusetts. In addition, this same blank field is used when you specify what data you want retrieved from the form. This "retrieval specification" you use can actually be *longer* than your longest data entry, so always assign at least two blank spaces more than the largest entry will require.

Beyond these two principles, pfs:FILE is quite flexible in how you can design your form. Keep in mind that the form should be easy to type in from the data you already have. For instance, if you already have a stack of index cards with the name on the top line; street address on the second line; and city, state, and zip code on the third line, it would make sense to set up your form in the same way. That way, it's not only easiest to enter your data into the computer, but it's also easiest to compare the computerized form to each handwritten card to check for keyboarding errors.

Our example, from a doctor's office, includes some technical terms that you may never have heard of. Don't worry about them. Type everything into the computer exactly as it appears in this book, and remember that you can always correct any mistakes. What is important in this exercise is not to figure out what a *pheochromocytoma* is (or even how to pronounce it), but rather, that you see how all the elements of a data base are interrelated. As you go through this sample program, you will use all of pfs:FILE and REPORT's main functions. The data we have included for you to enter, and the manipulations you will perform on that data, are designed to be both educational and fun. So roll up your sleeves and prepare for adventures.

To actually design our file, option 1 on the Main Menu, Design File, seems to be just the ticket. So, with the cursor next to the highlighted area SELECTION NUMBER:, type:

1

At this point you need to know a few things about how pfs:FILE operates. The [F10] key has special significance to pfs:FILE. Basically, whenever you press [F10], you are signaling that you have completed everything you want to do with the present screen or menu. To remind you of this, pfs:FILE will keep a prompt message in the lower right-hand corner of the screen that says:

**F10-CONTINUE**

Right now, though, you have not yet completed the Main Menu requirements, because you haven't specified the FILE NAME: yet. The pfs:FILE cursor can be moved in two ways. One is the standard fashion, using the various cursor control keys (the four arrows) and the [ENTER] key. With these keys you can move the cursor to any position on the screen. When you are operating in a menu or in a previously designed file, there is a second option. The [TAB] key automatically advances the cursor to the next data entry point. After you have typed

1

in the SELECTION NUMBER: area, press the [TAB] key to move the cursor to the FILE NAME: position.

Type:

**DRDATA1**

for the filename. pfs:FILE filenames follow the same rules as DOS filenames, so they too can have a three-character filename extension. (If you need to brush up on the DOS filename conventions, return to Chapter 6 for a quick review.) Press [F10] to signal to pfs:FILE that you have finished using the Main Menu.

pfs:FILE will now display the Design File Menu, which offers two options: Create File or Change Design. The Change Design option is not available if you are using a single disk drive system (standard for the PCjr). Type:

1

to specify the Create File option, and press [F10].



If you specify the Create File option along with the name of a file you have already created, FILE will double check that you really intend to erase that file and begin again. If you do, press [F10]. If you entered the filename in error, and you do *not* want to redo your file design, use the [ESC] key to return to the Main Menu.

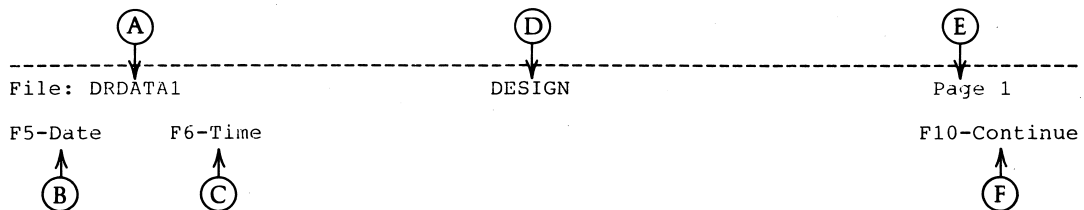
You will now have an almost blank screen with some information displayed along the last few lines (Figure 10-2). In the lower left-hand corner of the screen, you can see the name of the file being created ({A}), as well as the assigned meaning of two of the function keys—[F5] ({B}) and [F6] ({C})—which will insert today's date and time respectively into the file. In the center the current menu function, Design ({D}), is displayed. In the right-hand corner are the page number of the current file and the prompt for [F10] (indicating that you have finished the current function).

The cursor, however, should be above the line, in the home position (upper left-hand corner). It is in this area that you will design your data file. The file will be composed of item names. Each item name must end with a colon. The spaces that are left blank beyond the colon represent the area where the data eventually will be stored. Make sure you always leave sufficient blank space to accommodate the largest possible data entry in that item—then add a couple of extra spaces for good measure. These extra spaces will give you room to specify your data retrieval parameters, which can be longer than the longest data entry (more on this later).

Start in the home position and type:

**PT NAME:**

(patient name). The first item in the FILE data base is extremely important because of the fact that when FILE searches your data



**Figure 10-2** Screen display for creating a file.

file to retrieve information, it always starts with this first item. If the first item is the one you are most likely to be searching for, your searches will be as speedy as possible. In our example patient name is a common heading under which the file will be searched.

Use the [→] key to move the cursor to the middle of the first line and enter:

**DATE 1ST SEEN:**

Use the [ENTER] key or the combination of the [←] and [↓] keys to enter the remaining items until the first page of the file looks like Figure 10-3. If you make a mistake, use the [BACKSPACE] or [DEL] keys to remove errors or the [INS] key to insert a missing letter.

Now for page 2 of the data file. On this second sheet you will be listing the operations the patient has undergone, the dates, and the diagnoses. To begin entering these items, press [PGDN]. Enter the items exactly as they appear in Figure 10-4. When you have completed both pages, you may return to page 1 by using the [PGUP] key. Press [F10] to make the file design a permanent part of the

```

PT NAME:                                DATE 1ST SEEN:
ADDRESS:
CITY:                                    STATE:    ZIP:
BLUE CROSS #:                            REFERRING MD:
DIAGNOSIS:
OTHER MED PROBLEMS:

MEDICATIONS:

                                DATE NEXT APPT:

-----
File: DRDATA1                                DESIGN                                Page 1
F5-Date    F6-Time                                F10-Continue

```

**Figure 10-3** Entering information to create a file.

OPERATION:	DATE:	DX:
OPERATION:	DATE:	DX:
OPERATION:	DATE:	DX:
OPERATION:	DATE:	DX:
OPERATION:	DATE:	DX:
REMARKS/OTHER:		

```

-----
File: DRDATA1                DESIGN                Page 2
F5-Date    F6-Time                F10-Continue
  
```

**Figure 10-4** Creating the second page of the file.

diskette file. Until you press [F10], the file design will be wiped out if you press the [ESC] key.

## Entering Data On A File

After you press [F10], you will return to the Main Menu. You are now ready to enter the data to your data file. Enter selection 2, then press [F10] (the filename DRDATA1 should still be registered in the FILE NAME: area). The first page of the file you just designed will reappear, but in slightly different form, as you can see in Figure 10-5. The item names are now highlighted, and the cursor cannot be moved into these highlighted areas.

Note two subtle changes in the information displayed beneath the line at the bottom of the screen. In the center it now says FORM 1, to indicate that this is the first form of the current file for which data is being entered. In addition, the prompt "F2-Print Form" has appeared in the lower left. (If you had wanted to print out the blank form, you could have used the [F2] key to invoke this option.)

```

PT NAME:                                DATE 1ST SEEN:
ADDRESS:
CITY:                                   STATE:   ZIP:
BLUE CROSS #:                           REFERRING MD:
DIAGNOSIS:
OTHER MED PROBLEMS:

MEDICATIONS:

DATE NEXT APPT:
-----
File: DRDATA1                                FORM 1                                Page 1
F2-Print Form    F5-Date    F6-Time                                F10-Continue

```

**Figure 10-5** The file is now ready for the data to be entered.

The cursor should be next to the colon for the first item—patient name. Before you enter any data, press the [TAB] key several times and watch the cursor. The [TAB] key moves the cursor quickly from item to item in the file. To move the cursor backward one or more items, press the [SHIFT] and [TAB] keys simultaneously. If you use the [TAB] key to reach the last item on the page (in this case, DATE NEXT APPT:) and press the [TAB] key again, the cursor will move back to the first item (PT NAME:), *not* to the top of page 2. To get to page 2, use the [PGDN] key as you did before.

Return to the first item on page 1, and enter the first name:

**BROWN, KAY W.**

Press the [TAB] key and enter the date first seen:

**80-04-11**

Take special note of the format used by pfs:FILE for entering dates: YY-MM-DD. This format makes it easier to compare dates numer-

ically. (The dashes are simply ignored and the two dates are compared as if they were numbers.) For example, to determine which patient was seen first, 80-04-11 might be compared with 79-03-19. The number 790,319 is less than 800,411, therefore FILE knows that the second patient was seen before the first. Use the [TAB] key to move to the next item, and the next, until you have entered all the data as it appears in Figure 10-6. To leave an item blank (such as OTHER MED PROBLEMS:), simply [TAB] past it.

When page 1 is complete, use [PGDN] to move to page 2. Fill in the information for page 2 so that it appears identical to Figure 10-7. Now, press [PGDN] again. (Remember, you designed your file to have only two pages, so you will be moving to a page 3 that doesn't exist.) When you move to a page that has not been previously set up with the Design File option, pfs:FILE assumes you have an overflow of data and provides you with an "attachment" page. You can enter whatever additional data you desire on this extra page, and it will be appended to the end of the form.

In this case there is no additional data to be entered onto the attachment sheet, so use [PGUP] to return to page 2. Press [F10] to

```

PT NAME: BROWN, KAY W.          DATE 1ST SEEN: 80-04-11
ADDRESS: 63 DREW PLACE
CITY: NEWPORT BEACH           STATE: CA   ZIP: 93661
BLUE CROSS #: 1631267        REFERRING MD: BADASCI S
DIAGNOSIS: BREAST CYST
OTHER MED PROBLEMS:

MEDICATIONS:

```

DATE NEXT APPT: 84-02-03

-----  
File: DRDATA1

FORM 1

Page 1

F2-Print Form

F5-Date

F6-Time

F10-Continue

**Figure 10-6** All the data entered on the first page of the file.

OPERATION:	CYST ASPIRATION	DATE:	80-04-11	DX:	BENIGN
OPERATION:		DATE:		DX:	
OPERATION:		DATE:		DX:	
OPERATION:		DATE:		DX:	
OPERATION:		DATE:		DX:	
REMARKS/OTHER:					

-----  
File: DRDATA1

FORM 1

Page 2

F2-Print Form

F5-Date

F6-Time

F10-Continue

**Figure 10-7** All the data entered on the second page of the file.

store the first form onto the diskette. Remember, until you press [F10], all the data you have entered on the form can be wiped out by typing the [ESC] key.

Now, turn to the Appendix and enter the data on the remaining patients into the appropriate forms. Be sure to fill in both pages and to use the [F10] key to save each form onto the diskette. Items on the form that have no data are designated by an asterisk in the appendix. Do not enter the asterisk; merely [TAB] past that item. If the item says NONE, enter:

**NONE**

Use the [BACKSPACE], [DEL], and [INS] keys to correct any mistakes, as we explained earlier. When you reach form 11, press [ESC] to return to the Main Menu.

Well done so far. You've prepared your first pfs:FILE file. Now, let's take a quick review.

1. pfs:FILE is a menu-oriented data base management program that works hand in hand with pfs:REPORT to store data and pro-

duce reports. The FILE Main Menu is accessed by booting the program or by pressing the [ESC] key. Option 1—Design File—allows you to create new data files. (The Change Design option is not available on single disk drive systems like the enhanced version of the PCjr.) Selection 2—Add—is used to enter data into the completed file. Option 3—Copy—is unavailable on single disk drive systems. However, the DOS Copy command can be used to accomplish the same functions.

2. pfs:FILE uses special function keys for data entry and manipulation. The following keys have been introduced so far:

[ESC] returns you to the Main Menu. Whatever data you are entering on the current form will be lost when you press [ESC] to go to the Main Menu.

[F2] prints the form currently being displayed.

[F5] enters today's date in the format YY-MM-DD—76-07-04 for July 4, 1976, to use an example. All dates in FILE forms are entered in this format.

[F6] enters the current time in the format HH:MM, using twenty-four-hour time. For example, 2:15 a.m. would be 02:15 (2:15 is also acceptable) and 2:15 p.m. would be 14:15.

[F10] saves the current form onto the diskette file or activates the function that has been selected from a menu.

[TAB] moves the cursor to the next item in a form or menu.

[SHIFT TAB] moves the cursor back to the previous item in a form or menu.

[PGDN] moves the cursor to the next page of a multiple-page form. If the cursor is moved beyond the last page of a form, an attachment page (or pages) is added to the end of that particular form.

[PGUP] moves the cursor to the previous page of a multiple-page form.

3. Files are composed of forms, which are made up of pages of items. Each item must end with a colon and must be followed by a sufficient number of blank spaces to allow for data retrieval specifications (which may be several characters longer than the longest data element to be entered). When FILE searches a form, it begins with the first item in that form. For fastest possible data retrieval, make the first item in the form the one most likely to be searched for.





After you have finished with the current form, press [F10] to see the next form selected. If you wish to see every form of the file, don't enter any retrieval specifications at all—just press [F10]. For now, press [F10] and go through both pages of each of the ten forms to check for typographical errors (check the data against the Appendix). If you find any, merely [TAB] over to the erroneous entry and retype it correctly. When you have completely “proofread” your file, return to the Main Menu and again type:

## 4

Retrieval specifications provide for five possible “matches.” The first match is the “exact” or “full item” match. This means that FILE will retrieve only those forms that exactly match the retrieval specification at the designated location. FILE ignores extra blank spaces and doesn't distinguish between capital and lower-case letters, but otherwise any difference will disqualify a form from being selected.

For the item NAME:, the following will *match* using the “exact match” specification JIM DEAN:

```
JIM DEAN
jim dean
JIM DEAN
  JIM DEAN
```

The following, however, will *not* match:

```
James Dean
jimmy dean
jim
JIMDEAN
MR. JIM DEAN
```

Simple, right?

The next type of match is a “partial” match. Here, FILE will search for any form that contains the retrieval specification as part of the entry for the designated item. FILE uses two periods to represent the part of the entry that doesn't need to be an exact match. For example, if you worked in a sporting goods store and wanted a list of all the different types of balls you carried—tennis balls, basketballs, baseballs, and so forth—you would specify “..balls.” Now,

the "balls" must fulfill the criteria for an exact match, but anything (or nothing) preceding it is ignored by FILE. However, an entry like GOLF BALLS—YELLOW would not be a match because the "—YELLOW" after "BALLS" is not an exact match. To rectify this, you would use two sets of periods: "..BALLS..". Returning to the first example, you would find that "..JIM.." would match:

JIM DEAN  
 JIM  
 Diamond Jim Brady  
 Mr. Jim Smith  
 jimdean

but not:

James Dean  
 Mr. DEAN  
 J I M

If the "." specification is used without anything else, every form with any data entered for the specified item will be selected. So, if you were an insurance agent selling renter's insurance, you might want to select the forms for any person having an apartment number. The "." specification for the item APT NO. will ignore any form with no apartment number entered. Note, however, that if NONE was entered as the apartment number, this form *will* be selected by the "." specification.

The third type of match is the "numeric" match. This specification instructs FILE to search only the numbers in an item and to select whatever forms have the appropriate numeric match. If there are letters and numbers, as in the date JUNE 3, 1984, the letters and punctuation marks are ignored—FILE would match this item with the number 31984. The same is true for money (\$2,983.56 is considered to be the number 2983.56), time (12:13 p.m. is considered 1213), or dates (83-06-03 is considered 830603).

A numeric match has three different possibilities: The item can be *equal to*, *greater than*, or *less than* the specified number. You can select forms fulfilling any of these three possibilities by using the symbols: =, >, <.

For example, for the item DATE 1ST SEEN:, the retrieval specification <800101 will select the forms for all patients first seen

before New Year's Day of 1980. For the item SALARY:, the specification >15000 will select the forms of all employees making more than \$15,000 a year.

The numeric match can be extended into the "numeric range" match to include a range of values. The rules for numeric matches apply to the "numeric range" match, as well. The range is indicated by two periods. To select the forms of patients first seen in the month of January 1980, you could use the range retrieval specification =800101..=800131 (the "=" makes the range inclusive—that is, including the first and last values). To select the forms of all employees making between \$15,000 and \$24,999.99, use the retrieval specification = 15000..=25000.

The final type of match works to negate all the other matches. It is called the "not" match. The "not" match is represented with a slash. So, /JIM DEAN will select all forms that do *not* have the name Jim Dean in the specified item. The specification /..BALLS.. will select all forms that do *not* have the word balls in the item. The specification /<800101 will select only those forms *not* seen before January 1, 1980. The specification /= 15000..=25000 will select the forms of employees *not* making between \$15,000 and \$24,999.99 annually. The specification "/.." will select only those forms that are blank for the designated item, and /NONE will select only those forms that do *not* have a NONE entered for that item—two very important "not" matches.

With these five simple types of matches, you can retrieve any form or forms from your file. Now, let's actually carry out the search, using the DRDATA data file you have already set up. From the Main Menu, choose selection 4 (if you have not already done so). The blank first page of the form should be displayed. The first forms we will search for are those patients who will have appointments on January 4, 1984, so that Dr. Data can review the next day's schedule. The field for DATE NEXT APPT: is the last one on the first page, so use the [SHIFT TAB] combination to move directly to that item. Type:

**84-01-04**

at this location. Notice that you are asking FILE for an exact match—not a numeric match. Press [F10] to start the search.

The first form FILE retrieves should be form 8, patient Fredrick Trapletti, who is scheduled for an appointment on January 4, 1984. Dr. Data can now browse through this patient's record, noting the diagnosis and associated medical problems. The use of [PGDN] will give Dr. Data the opportunity to review the patient's surgical history as well. If desired, the [F2] key can be used to print out the form (more on this later).

To retrieve the next form, you merely press the [F10] ("continue") key. Form 3, Benjamin Crenshaw, should now be displayed. Again, the same options are available. Dr. Data would want to be particularly cognizant of the REMARKS/OTHER: entry on page 2—a warning that the patient may need surgery. Note that FILE retrieved the forms in the opposite order from which they were entered. Press [F10] again when you have reviewed this record. FILE tells you how many forms were retrieved in this search (two) and instructs you to press [F10] to return to the Main Menu.

Again, select the Search/Update option. Using the various matches you have learned about, perform the following four searches:

1. Find all patients seen for the first time during the year 1980.
2. Find all patients taking the medication cimetidine, either alone or with some other medication.
3. Find all patients whose diagnosis involves a breast problem (watch out, this one could be tricky!).
4. Find all patients who do *not* live in the state of California (CA).

The data retrieval specification is the foundation of both pfs:FILE and REPORT. Compare the results of your four searches with the data listed in the Appendix to be sure you are correct. (You should have found 3, 2, 2, and 1 forms in your four searches.) If you did not perform these searches correctly, go back and review this section until you are sure you understand the retrieval specification.

Let's take time out to review the pfs:FILE and REPORT retrieval specifications.

1. To retrieve data from pfs:FILE, select option 4 from the Main Menu—Search/Update. The blank form will appear on the screen with the message RETRIEVE SPEC. To review all the forms

contained in the file, simply press [F10]. To select for specific files based on one or more data entries (for example, all forms of patients with a diagnosis of breast problems, or all people who live in an apartment, or all the forms relating to a Jim Dean), use the [TAB] key to move to that entry and enter the retrieval specification.

2. The retrieval specification tells pfs:FILE which forms to select, based on five matching criteria:

a. An "exact" or "full item" match instructs pfs:FILE to select only those forms in which the entered data is an exact match to the retrieval specification (ignoring upper- or lower-case letters and extra blank spaces).

b. A "partial" match selects any forms containing the retrieval specification as part of the entry, even if the rest of the entry does not match. Two periods are used to specify the portion of the entry that does not have to match. For example, "..DEAN" matches Jim Dean, Mr. Dean, Dean, and JIMDEAN, but not DEANS LIST, or Mr. Dean Stevens. The "." specification used by itself retrieves all forms that have any data entered; that is, all forms except those left blank in that position.

c. The "numeric" match addresses only the numeric part of an entry, even if both numbers and letters are found in the same entry. Numeric matches use the "=", "<", and ">" symbols to specify the selection of forms containing exactly, less than, or greater than the specified numeric value.

d. The "numeric range" combines the rules for the numeric match with a range specification, indicated by two periods. Thus to select all forms containing an entry that has a value between 1 and 10, the specification = 1.. = 10 would be used.

e. The "not" match retrieves all the forms that would *not* be retrieved by the accompanying retrieval specification under one of the other four match criteria. So, there could be "not exact," "not partial," "not numeric," and "not numeric range" matches. A retrieval specification is made into a "not" specification by the addition of a slash, as in /JIM DEAN.

3. If more than one retrieval specification is given, pfs:FILE and REPORT will select only those forms fulfilling the criteria for *all* specifications.

## Printing Selected Forms

You may recall from the previous section that the Search/Update command will let you print the forms you have retrieved, using the [F2] key. However, this requires two separate operations—first the Search/Update and then the [F2] (Print Form) key. Furthermore, every form must be printed individually. pfs:FILE provides a faster way to print selected files. On the Main Menu this is selection 5: PRINT.

After you select option 5 and specify the appropriate filename in the Main Menu, pressing [F10] will reproduce the blank form and ask for the retrieval specifications. The rules for Print function retrieval specifications are identical to those for Search/Update. After you enter all the retrieval specifications, [F10] will generate a new screen asking you to fill in the PRINT OPTIONS (see Figure 10-9). There are four options, and pfs:FILE has already filled in “suggested” answers for each.

The first option says PRINT ITEM NAMES (Y/N);, and the default (suggested or standard) answer is Y for yes. A yes answer prints the form exactly as it appears on the screen, complete with the item names. A no answer will cause the form to be printed *without* the item names; that is, only the actual data will be entered.

PRINT TO, the second option, requests the DOS name of the device that will do the printing. The default option is LPT1, which is the DOS name for a parallel printer. The other options are either COM1 or AUX—to specify a serial printer, or a diskette filename to cause the file to be printed onto a different diskette file from the one it is stored on (occasionally this is necessary with single disk drive systems that cannot make use of the COPY function or to put a subdivision of data into a different file).

Next, the LINES PER PAGE option requests the number of lines you want used for each “page” of pfs:FILE output. A standard printer page has sixty-six lines, which is the default setting. Keeping this setting will cause each page of the pfs:FILE form to be printed on a separate page. A setting of thirty-three will cause two pages of the form to be printed on each printer page.

Finally, the NUMBER OF COPIES option asks for the number of copies of each form you wish printed. The default option is 1.

## PRINT OPTIONS

```
PRINT ITEM NAMES (Y/N): Y
```

```
PRINT TO: LPT1:
```

```
LINES PER PAGE: 66
```

```
NUMBER OF COPIES: 1
```

F10-Continue

**Figure 10-9** Print options of pfs:FILE.

To change any option, simply [TAB] over to the appropriate location and enter the new answer. Any time you change a print option, the changed value becomes the new default value for the remainder of the session, so that all subsequent printouts will look identical—unless you change the setting again.

After you enter the print options (or leave them unchanged), press [F10] to return the blank form to the screen. pfs:FILE is now asking for the Print Specifications. If you want the entire form printed, [F10] will activate the printer. However, if you want only selected items printed—like a mailing label that would contain only the NAME and ADDRESS—you can use the print specifications to identify the items to be printed.

There are two print specifications: X and +. The X specification entered for an item causes pfs:FILE to print that item and advance to the next line. The + specification causes pfs:FILE to print the item but *not* to advance the line (instead, it advances two spaces). This is particularly useful for items like STATE and ZIP CODE that are printed on the same line as the CITY.

There is one final print specification—the S for Sort option. This can be used once per printing. Whatever item the S is entered

in will be sorted alphabetically before printing. This is true even if the sorted item is not printed.

As an illustration of the PRINT function, let's return to Dr. Data's office records—stored in the file DRDATA1. Say Dr. Data is mailing bills to all patients and needs mailing labels for the envelopes, sorted by city. In addition, Dr. Data wants to make sure that no bills are sent to patients who have died.

Return to the Main Menu and select option 5 for file DRDATA1. Press [F10] to get the blank form for retrieval specifications. Dr. Data wants the name and address printed for every patient, except the ones who are dead. As you may remember, these patients were listed in the data base as "expired" under the REMARKS/OTHER: heading on page 2. Press [PGDN] to get to page 2, and use the [SHIFT TAB] combination to get to that heading. We are interested in *not* selecting any form where the word "expired" appears in the beginning of the entry. Use the "not partial match" /EXPIRED.. to select for printing only the living patients, then press [F10] to get to the print options.

Providing you have a parallel printer, change the print item names to N, because you do not want your mailing label to say NAME: and ADDRESS: and so on. Then change the lines-per-page option to 6. Most standard mailing labels accommodate 6 lines of typed material. Press [F10] to get to the print specifications.

The print specifications ask for the items to be printed (in this case, name, address, city, state, and zip code) and the sort priority (in this case, city). For the format of this mailing label, we will want name on one line, address on the next, and city, state, and zip code together on the third. So, using the [TAB] key as necessary, put an X next to NAME: and ADDRESS: and ZIP: (remember that ZIP: will be the last item on the line). Put a + next to CITY: and STATE:. But CITY: also needs an S to sort by this item (note that the S does not replace the need for a + or X specification).

Turn on your printer and press [F10] to begin printing the mailing labels. Note that you could also print these mailing labels onto another file on the diskette, naming it PTADRES1, so that whenever it was time to send bills you could print the entire label file without repeating all the searches and specifications. To do this, you would write the file onto the diskette *with* the item names (option 1 of the print options) but later print the labels without the item names.



## Removing Forms

Let's discuss the last two selections from the pfs:FILE Main Menu. Selection 6 is a simple one—REMOVE. Use this function to remove out-of-date forms from your file. The forms to be removed are selected by entering retrieval specifications just like those for the other search options. pfs:FILE always confirms that you really want to permanently erase forms from your file, and it asks you to type [F10] if you really want to go through with the removal. If you realize you are making a mistake, just press the [ESC] key to cancel the removal function. Remember that if you don't enter any retrieval specifications, you are telling pfs:FILE to erase every single form in the file. FILE will doublecheck with you first, but be careful! After completing the remove function, pfs:FILE will tell you how many forms it has erased.

When you are finished with pfs:FILE, return to the Main Menu and enter selection number 7—EXIT pfs:FILE. This will return you to the DOS prompt A>. You can now insert pfs:REPORT or any other program into the disk drive or shut off your computer.

It's time for another review session.

1. You can print selected forms from your file using the Print Option from the Main Menu (option 5). You then enter the retrieval specifications just as for the Search/Update function. The designated forms will be retrieved and printed according to the Print Options you specify.

2. The four Print Options already have standard, or default, settings. pfs:FILE will use these default settings unless you change them. You can specify: whether you want the item names printed along with the data entries, the device on which you want the output printed (parallel or serial printer or diskette file), the number of lines per page, and the number of copies you want printed.

3. You don't have to print every item from a file when you use the Print function. You can enter Print Specifications to select only the entries you want printed—such as name and address on mailing labels. There are three print specifications: X for print then move to the next line; + for print but do not advance a line; and S for sort the data by this item before printing.

4. Selected forms or an entire file can be deleted using option 6—Remove. Again, this function will accept retrieval specifications

to specify which forms are to be destroyed. If new retrieval specifications are entered, the entire file will be deleted. pfs:FILE always gives you a chance to change your mind before it permanently erases any forms.

5. Once you are finished with pfs:FILE, return to the Main Menu and select option 7—EXIT pfs:FILE—to return to the operating system.

## Generating Reports

You now have had a thorough introduction to pfs:FILE, and you can see how a computerized filing system can streamline office paperwork and record keeping. By combining forces with pfs:REPORT, you will be able to do even more. pfs:REPORT generates reports from pfs:FILE files. It can perform sorting, calculation, and retrieval functions to permit sophisticated manipulation of the data you have filed, allowing you to see unrecognized trends or to generate periodic summaries.

pfs:REPORT shares many of the features of pfs:FILE, including its menu-oriented design and use of retrieval specifications. You have already mastered three-quarters of pfs:REPORT by learning to use pfs:FILE. As a first step to using pfs:REPORT, install the DOS operating system on your diskette with the INSTALL program, as instructed by the manual.

When you load pfs:REPORT, you will see the pfs:REPORT Main Menu, which has four options (see Figure 10-10). The first option, PRINT A REPORT, will allow you to summarize data from your pfs:FILE files in tabular form. In this example you will use your DRDATA1 file to generate a report listing all the referring physicians of Dr. Data's patients.

Select option 1 and enter the filename:

**DRDATA1**

Then press [F10] to see the blank form asking for the retrieval specifications. For a report, as for a search in pfs:FILE, you can obtain data only from selected reports by entering the appropriate retrieval specifications. The rules for retrieval specifications in pfs:REPORT are identical to those for pfs:FILE.

PFS: R E P O R T  
-----

- 1 PRINT A REPORT
- 2 PRE-DEFINE A REPORT
- 3 SET NEW HEADINGS
- 4 EXIT PFS:REPORT

SELECTION NUMBER:

FILE NAME:

(C) 1983 Software Publishing Corporation

F10-continue

**Figure 10-10** pfs:REPORT—Main Menu.

In the current example Dr. Data needs a report summarizing all the patients' referring physicians. Therefore the computer would select the forms of all the patients who have a referring physician. Patients without a referring physician in the DRDATA1 data file are designated with the word NONE in the REFERRING MD: field. Accordingly, the only forms that should *not* be selected are those with the word NONE in that field. Type:

**/NONE**

(the "not" specification) in that field of the form, and press [F10].

The screen now shows the REPORT OPTIONS (Figure 10-11), with five options available. The first is the title of the report. Call this report:

**REFERRING PHYSICIANS**

Next, pfs:REPORT asks for a predefined report name. If a report is used repeatedly, you can store the parameters as their own named file. Then, whenever you need this report printed, you simply enter

## REPORT OPTIONS

TITLE:

PRE-DEFINED REPORT NAME:

OUTPUT TO: lpt1:

LINES PER PAGE: 66

PAGE WIDTH: 80

F10-continue

**Figure 10-11** pfs:REPORT—REPORT options.

the report name (different from the report title, which is the information printed at the top of the report) and pfs:REPORT knows exactly how to set up your report. You will see an example of this later on. For now, skip this entry, leaving it blank.

The last three entries have to do with your output device (the standard or default entry is LPT1 for the printer) and the size of your page (default is a standard page size of eighty columns and sixty-six lines). If you have a serial printer, change the output option to either COM1 or AUX; otherwise leave the options as is. Then, press [F10] to move to the next screen—the REPORT SPECIFICATIONS.

Just as you can choose which *forms* to use when producing your report, you can choose which *items* from the form to include in the report. In fact, every report is limited to a maximum of sixteen columns, with each item taking up one column. You must indicate which items you want included in your report, and number them in order from left to right (the left-hand column is number 1, the next column is number 2, and so on).

When the report is printed, pfs:REPORT will sort the data in the first column into alphabetical order and print the report that

way. When the first column contains two or more items that are the same (a common occurrence), these items will be sorted based on the second column. If for any reason you do not want the data sorted, begin with column 3 in the report specification and no sorting will be performed. This method, however, leaves you with two fewer columns in your report.

For this example you will want the first column (the sorted column) to be the name of the referring physician, followed by the patient name, the diagnosis, and the date of the next appointment. Therefore use the [TAB] key to move to the various fields of the form and number them 1, 2, 3, and 4, respectively. (You do not have to keyboard the numbers in that order. You can start by putting a 2 in the PT NAME: field, then go backward to DATE NEXT APPT: and enter a 4, and so on.) Now, pressing [F10] will cause your first pfs:REPORT report to be printed, so go ahead. The final report should look like Figure 10-12.

Notice that pfs:REPORT has sorted the patients by referring doctor in column 1, and for the same doctor has ordered the patient names alphabetically. Also notice that pfs:REPORT leaves a blank line between patients if the preceding referring physician had more than one patient. That is, there is a line between DOVER and LEVITT, because DOVER is the last patient of Dr. Badasci (three patients), but no line between LEVITT and TRAPLETTI, because LEVITT was the only patient of Dr. Carlson.

The final items to consider are the headings and column widths. The headings of each column are printed exactly as they appear on the pfs:FILE form. The columns are fashioned to be just wide enough to accommodate the widest data entry (for instance, RT INGUINAL HERNIA for DIAGNOSIS) or just wide enough to accommodate the heading if it is wider than the widest entry (such as DATE NEXT APPT).

Because of these rules for headings and column widths, the report does not always come out exactly as you would like. You can alter the headings, however, to fit the data columns, using selection 3 of the pfs:REPORT Main Menu.

Say that Dr. Data wants to modify the report, changing the heading PT NAME to PATIENT NAME and shortening the heading DATE NEXT APPT to NEXT APPT. Return to the Main Menu and enter a 3, then press [F10]. You will now see the blank form with the indication HEADINGS in the bottom center of the screen.

REFERRING PHYSICIANS			
REFERRING MD	PT NAME	DIAGNOSIS	DATE NEXT APPT
BADASCI S	BROWN, KAY W.	BREAST CYST	84-02-03
	CRENSHAW, BENJAMIN	CHOLELITHIASIS	84-01-04
	DOVER, BENNETT	COLON CA	84-03-14
CARLSON F	LEVITT, IRENE	ADRENAL ADENOMA	84-03-21
CHRISTIE W	TRAPLETTI, FREDRICK	PHEOCHROMOCYTOMA	84-01-04
ECONOMOU D	YOUNG, GENE E.	RT BREAST MASS	84-01-06
HANDLEY J	CONNERS, JASON	RT INGUINAL HERNIA	84-01-11
	GARDNER, ROGER	THYROID STORM	84-04-05
SPRULE B	MARTIN, SCOTT	GASTRIC POLYPS	

Page 1

**Figure 10-12** Your first pfs:REPORT should look like this.

To specify new headings, simply type the desired new heading next to the old heading. Type:

**PATIENT NAME**

next to PT NAME: and [TAB] back to DATE NEXT APPT and type:

**NEXT APPT**

Then press [F10] and return to the Main Menu. Reenter all the retrieval specifications and report options that you used to generate the report the first time around. Press [F10] again, and see your new, improved report, which should now look like Figure 10-13.

The last pfs:REPORT Main Menu option is selection 2, PRE-DEFINE A REPORT. This option allows you to store some of the data you use to generate a report, so that you can print frequently used reports quickly and easily. As an example, Dr. Data frequently

REFERRING PHYSICIANS			
REFERRING MD	PATIENT NAME	DIAGNOSIS	NEXT APPT
BADASCI S	BROWN, KAY W.	BREAST CYST	84-02-03
	CRENSHAW, BENJAMIN	CHOLELITHIASIS	84-01-04
	DOVER, BENNETT	COLON CA	84-03-14
CARLSON F	LEVITT, IRENE	ADRENAL ADENOMA	84-03-21
CHRISTIE W	TRAPLETTI, FREDRICK	PHEOCHROMOCYTOMA	84-01-04
ECONOMOU D	YOUNG, GENE E.	RT BREAST MASS	84-01-06
HANDLEY J	CONNERS, JASON	RT INGUINAL HERNIA	84-01-11
	GARDNER, ROGER	THYROID STORM	84-04-05
SPRULE B	MARTIN, SCOTT	GASTRIC POLYPS	

Page 1

**Figure 10-13** After revisions your pfs:REPORT should look like this.

needs an update of all the medications patients are taking. To help generate this report, you will predefine a report called MEDS.

Select option 2 from the Main Menu, and press [F10]. The screen will tell you the name of currently available predefined reports (it should say (None)) and ask for the name of the new report. Enter MEDS and press [F10]. This report will begin with the patient name in column 1, followed by the medications in column 2, then the diagnosis, other medical problems, and the date of the next appointment. Return to the Main Menu to enter the retrieval specifications—in this case Dr. Data would want all those patients *not* expired—and press [F10] to get the Report Options. Give this report the Title PATIENT MEDICATIONS, then [TAB] to the area for the PREDEFINED REPORT NAME:. Enter the name:

**MEDS**

and press [F10].

That easily, pfs:REPORT generates the patient medication report, and you never have to enter the report specifications again—just the predefined report name.

But wait, there seems to be a problem. pfs:REPORT is telling us that the report is too wide to fit on the paper. Press the [SPACEBAR] to have pfs:REPORT print out all that will fit, which should look like the report in Figure 10-14. Most of the associated medical problems got printed, but not quite all. The date of the next appointment never had a chance.

In a case like this, you have two choices. You can change the headings to shorter ones, in hopes of decreasing the column widths and fitting more columns on a page. In this report, however, all the column widths were determined by the longest entry, not by the heading length. Your other choice is to go back and modify your predefined report to include less information. In this case you could return to the PREDEFINE A REPORT screen and change MEDS so

MEDICATIONS			
PT NAME	MEDICATIONS	DIAGNOSIS	OTHER MED PR
BROWN, KAY W.		BREAST CYST	
CONNERS, JASON	HCTZ, INDERAL	RT INGUINAL HERNIA	HYPERTENSION, CAR
CRENSHAW, BENJAMIN		CHOLELITHIASIS	EMPHYSEMA
DOVER, BENNETT		COLON CA	CIRRHOISIS
GARDNER, ROGER	SYNTHROID	THYROID STORM	
LEVITT, IRENE	CIMETIDINE, PREDNISONE	ADRENAL ADENOMA	HYPERPARATHYROIDI
TRAPLETTI, FREDRICK		PHEOCHROMOCYTOMA	MEA-IIB
WEBB, JAMES	CIMETIDINE, VIT D	LARYNGOSPASM	CHRONIC RENAL FAI
YOUNG, GENE E.	QUIBRON (PRN)	RT BREAST MASS	ASTHMA, ALLERGY T

Page 1

Figure 10-14 Your final pfs:REPORT.



that OTHER MEDICAL PROBLEMS was dropped and DATE NEXT APPT was moved to column 4.

## Numbers And Calculations With pfs:REPORT

With pfs:REPORT, you can count, total, or average columns of numbers—much like with a spreadsheet program such as VisiCalc. In addition, you can get subcounts, subtotals, and subaverages every time the item in the first column is changed. To obtain these calculations, you simply add one of the following letters to the column number specified during the report specifications:

T for Total (totals the numbers in the column)

C for Count (counts the number of items in the column)

A for Average (divides the total by the count to obtain the average for the column)

ST, SC, and SA for Subtotal, Subcount, and Subaverage, respectively.

The calculated value is printed on a separate line below the last entry for the report (or item if it is a subvalue).

Along with these calculated values, you can include formulas—or “derived values”—in your pfs:REPORT report. Each derived value must have a heading, a formula, and a number indicating which column in the report this calculated value is to occupy (same as a report specification). You are permitted a maximum of three derived values per report.

To assign derived values, use the [F7] key while entering the report specifications (the notation “F7-Derived Columns” will be present on the report specifications screen to let you know this is available). The standard four mathematical operators (+, -, \*, and /, for add, subtract, multiply, and divide) are available. In addition, you can use parentheses to denote the order in which a complicated calculation is to be performed.

When you use a derived value, it will be printed out on the report just like any other column, with the heading you specify. If no heading is specified, pfs:REPORT prints the formula at the head of the column. For a more complete discussion (including examples) of the calculated and derived values capability of pfs:REPORT, consult your pfs:REPORT *User's Manual*.

The final option on the pfs:REPORT Main Menu is the EXIT pfs:REPORT option, which is selection 4. This returns you to DOS,

from which you can shut down your computer, switch over to pfs:FILE, or begin using any other PCjr programs.

You have now been introduced to pfs:REPORT as well as pfs:FILE. Let's review the material covered on pfs:REPORT.

1. pfs:REPORT is a report-generating program fully compatible with pfs:FILE. It retrieves forms from pfs:FILE files and prints tabular reports. pfs:REPORT uses the identical rules for retrieval specifications as does pfs:FILE and is also menu oriented.

2. To use option 1, Print a Report, enter retrieval specifications to determine which forms of the file are to be printed. Then enter Report Options, which give the report title, the name of a predefined report specification file (if used), and the output device (serial or parallel printer), page length, and page width. Finally, enter report specifications.

3. The pfs:REPORT report can have up to sixteen columns of data, designated by the Report Specification. With the Report Specification, each item to be included in the report is numbered in order of its appearance from left to right on the report. The report is sorted by the first column before printing. If the report is not to be sorted, begin numbering in column 3.

4. Special headings can be specified for each column with the Headings option (number 3). The width of any column of a report is determined by the width of the widest data entry or the width of the heading, whichever is wider. If a report is too wide to fit on the page, pfs:REPORT will tell you and will ask you if you want as much printed out as possible. Then, you can revise your report by eliminating columns or decreasing the column width by supplying new headings.

5. If a report is generated frequently, the report specifications can be predefined with option 2—Predefine a Report. Then, whenever you want to produce this report, you merely enter the retrieval specifications and the report options and your report is generated.

6. pfs:REPORT can calculate the average or total for a column or part of a column and print these values on the report. It can also keep count of the number of items in a column and print that value. In each report you can have three columns with derived data, specified by a formula. This brings some of the power of spreadsheet programming to your data base management needs.

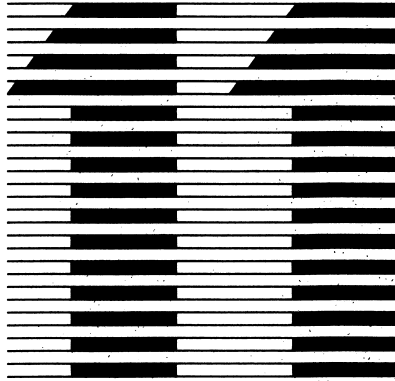
This chapter has covered the main aspects of data base management programming, but there is much more to be learned. Both *pfs:FILE* and *pfs:REPORT* have other features we did not have room to discuss. Still, the examples provided should clearly illustrate how data base management by computer works and how it works for you. A little experience will be all you need to fill in the gaps.

## Further Reading

*Data Base Management Systems*, by David Kruglenksi (Osborne/McGraw-Hill, 1983). A good introduction but not directly applicable to the PCjr.

*An Introduction to Database Systems*, by C. J. Date, 2 vols. (Addison-Wesley, 1982). The standard text on data base systems. These volumes probably contain more than you care to know about the subject, so look at them in the library before you decide to buy them.

*pfs:FILE and pfs:REPORT User's Manuals*. Well-written manuals that are consistent between the two application packages so that once you learn to use one package, you have a good understanding of the other. A worked-through example is included.



## Talking To Your PCjr: BASIC And LOGO

If you check the bookshelves of the top programmers—those professionals who have made valuable contributions to computer science (and a lot of money to boot)—one set of books is always among them: the series entitled *The Art of Computer Programming*, by Don Knuth. Knuth not only is a brilliant programmer, he is also a perceptive observer of the programming scene. It was no accident that he called programming an art. And that word—*art*—says a great deal about programs, programmers, and how to learn programming.

### The Art Of Programming

As “works of art,” programs have style—that elusive quality that makes them either attractive, graceful, and utilitarian, or crude, ugly, and awkward. In this chapter you will learn the essence of style and how to use it in creating your own programs.

Another point about pieces of art is that there are really no hard and fast rules about the best way to create them—the best way to draw, write, or act. That’s true for programming, too—there is no one “best” way to program. There are schools of thought, methods, and suggestions, but there is no perfect formula or (as in mathematics) “optimal solution.” Your way can be the best way for you—as long as it works. So feel free to experiment. Try something different, but also learn from others.

We will show you how to approach a programming problem, how to start the program, and, just as important, when to finish

the program. A famous artist is supposed to have said: "My work is never done until they take it away from me." Unfortunately, too many programmer-artists (especially younger ones) don't know when the job is "done" either.

Good art and good programs have a certain economy of style. They are straightforward, clean, and uncluttered. Computer scientists use the acronym KISS (*Keep It Simple Stupid*) when teaching new programmers how to design that first program. The expression is crude but the advice is sound. Concentrate on making sure your program can do the job efficiently. Don't worry about creating a masterpiece.

The use of files is one of the most important parts of programming in real-life situations. Most introductory chapters and books on programming avoid the word *file* as though it were a four-letter word. Here we'll not only talk about files and describe them, we'll give examples of their use. What's more, we'll show you that files are fun.

## Programming Languages For The PCjr

There are over 1,000 computer languages available today, but only a few are of any interest to the majority of computer users. A glance at the programming section in computer and book stores shows us which languages are most common. Titles having the word BASIC outnumber all others, followed by two other languages: LOGO and Pascal.

BASIC is the most popular computer language in the world today. It is taught in high school and college computer classes and is used in industry. Versions of BASIC can be found on computers ranging in size from the smallest home computer to the largest supercomputer. Most home and personal computers, like the PCjr, have a version of BASIC built into the ROM. The power of the language varies considerably between the different dialects, but the BASIC available with the PCjr is one of the best.

LOGO is quite different from BASIC. LOGO was pioneered by Seymour Papert at the Massachusetts Institute of Technology as a vehicle for teaching young children how to use the computer. Despite LOGO's apparent simplicity, it is an extremely powerful language with a number of features that are considered to be at the cutting

edge of computer language design. The LOGO system available for the PCjr is also one of the best on the market. In this chapter we will examine program construction using LOGO. After trying your hand at this interesting language, we think you will agree that LOGO is not just kid stuff.

Pascal, because it is available for the PC and for many other personal computers, probably will be available for the PCjr before long. Therefore it's worth taking a brief look at Pascal in this chapter. By the way, the word *Pascal* is not written in capitals as are most other languages. Pascal is named after a French mathematician, Blaise Pascal, whereas the other names are acronyms.

All three programming languages have an interesting common thread. Each was developed at a university—BASIC at Dartmouth, LOGO at MIT, and Pascal at Stanford and the University of California at San Diego—and all started as teaching languages. At a time when our educational system often is under fire, these outstanding achievements in computer education seem to have gone unnoticed.

You already have had a general introduction (Chapter 4) to Cassette BASIC, the BASIC that comes with the PCjr. In this chapter we will focus on the art of writing BASIC and LOGO programs. We will build on the material presented earlier, so you may wish to refresh yourself by quickly reviewing Chapter 4 before you continue.

You may ask: Why another chapter? Why not just lengthen the previous chapter to include this material? Our reasons are simple. First, this chapter deals with Cartridge BASIC using files and LOGO. Both of these programming languages must be purchased, which means an extra cost for you (although PCjr Cartridge BASIC and Disk BASIC on DOS 2.10 are two of the best software buys around). Second, it takes some time and experience to understand the ideas presented here. After you have wrestled with EDLIN, been tossed by DOS, and worked with less than perfect manuals, the observations and comments given here on programs and programming will have more impact and meaning.

This is a long chapter, but don't feel intimidated by it. Take your time, read it slowly and carefully, and make certain you really understand the concepts we talk about. Everything will start to make sense before you know it.

## Six Steps To Successful Programs

You would be startled if a construction crew just walked onto an empty lot and started to build. It is only reasonable to expect that the workers (regardless of their skills) would have detailed plans to describe every step of the building process. In fact, even before those plans were drawn, the property owner and the architect probably would have spent several days together discussing their goals and objectives. Clearly, the better the planning process, the better the end results.

The same thing holds true for programs. They, too, require a practical, no-nonsense approach. Computer science experts agree that there are six key steps in building successful programs. These are:

1. Be sure you understand exactly what you want the program to do.

2. Develop your programming approach before you start to write the program.

3. Code your program. (Translating your problem into a programming language is called coding.) Remember KISS. The simple and direct way is the best; don't try to be fancy.

4. Test the program thoroughly. Prepare your test data carefully and be sure you know the kind of answers you expect. (Don't scoff. Many people only *think* they know what to expect.)

(Errors in computer programs are called *bugs*; correcting errors is known as *debugging*. The connotation of *bug* as a problem actually originated in the computer world. Once, in the infancy of computers, there literally was a bug in the equipment, which caused the machine to malfunction. No one could detect the problem. After taking the entire computer apart, technicians found the culprit, a large moth crushed in the works. From then on any problem jokingly was blamed on a bug.)

5. Document your program. Put remarks and comments in the program itself and write down instructions about how the program works. Keep these written records for later reference. You'd be surprised how often a programmer returns to a program months after it was written and can't remember what he or she had been trying to do or how the program operates.

6. Maintain your program. When you make changes, find bugs,

correct errors, or add sections, remember to test the program again and update your records. Even a minor change can disrupt a smoothly working program, so don't try to shortcut the testing stages.

These six steps may sound like a lot of work, but many studies have shown they are actually time-savers. Most beginners believe that writing the program is the most time-consuming step in the programming cycle. The truth is that correcting errors and making changes to update the program (so that it will remain a useful tool) are the real time-takers. This six-step approach will make these tasks much faster and easier in the long run.

Now let's turn from philosophy to practice. We'll begin by building some BASIC programs. BASIC is a rich language. It offers many ways to accomplish the same goals. Often you can use three, or even four, different statements to do exactly the same thing. In discussing BASIC statements and programs we will try to follow the advice we gave you: Keep it simple. Therefore we will not attempt to cover every possible BASIC statement or even all the options for most of the statements. But we will present more than enough material to allow you to become a respectable programmer in a short time, if not one day.

## Variables In BASIC

Before you begin to write a program, you should define all the variables. As you may recall, a variable is a quantity in a program that can assume any value during a calculation. In BASIC variables must be named. Like filenames, variable names should convey a meaning about what the variable represents. Variable names can be composed of any letter or number and the decimal point, but they must begin with a letter. Unlike filenames (which are limited to eight characters), variable names can be up to forty characters in length, which allows more than enough capacity to create meaningful variable names.

Variables in programming languages can contain different types of data as well as different values. The two general types of data used in BASIC are numeric and text. In BASIC variables that contain text are called *string variables*. To indicate that a variable is a string variable, you must end its name with a dollar sign (\$). Strings can



contain up to a maximum of 256 characters. For example, the variable `TITLE$` can contain the words "The Complete Guide To Success With The IBM PCjr."

Numeric variables also can have a suffix. These suffixes tell BASIC how much storage space the variable will require in memory. For example, a one-digit integer would require much less storage space than a fifteen-decimal-place scientific measurement.

If you know that a variable will contain only integers (whole numbers), the percent sign (%) suffix will cause BASIC to allocate only two bytes of storage. This is a small savings over the four bytes normally allocated for a numeric variable without a suffix, but it can mount up.

In fact, the "standard" numeric variable can be written with a suffix to specify that it should have four bytes allocated. This suffix is the exclamation point (!). The numeric variable `REAL!` can contain real numbers (numbers with decimal fractions, such as 123.45) or integers because the exclamation point will cause it to be allocated four bytes of storage. This is the type of numeric variable you use regularly, and if you don't use any suffix, BASIC will assume you want this much storage. The formal name of this type of variable is "single precision, real numeric variable," and it can represent up to nine digits. It is called a numeric variable for short.

By using a pound sign (#) suffix, you can tell BASIC to set aside eight bytes for an extra-long, double-precision numeric variable. Double-precision variables sometimes are used in scientific calculations where extreme accuracy is needed.

## Arithmetic In BASIC

The very name *computer* means to calculate, and the computer's talent as a calculating machine is still one of its main strengths. The PCjr and BASIC make a winning combination for performing all sorts of complicated arithmetic operations. Writing arithmetic formulas in BASIC is easy and straightforward. All it takes is applying a few simple rules. We can illustrate these rules and refresh your memory about BASIC at the same time with a few examples.

A good place to start is where we left off in Chapter 4, on the subject of metric values. In the metric system temperature is mea-

sured in degrees centigrade. The arithmetic formula to convert our familiar Fahrenheit measurements to centigrade is:

$$\text{degrees centigrade} = 5/9 \times \text{degrees Fahrenheit} - 32$$

If the temperature is a balmy 78 degrees Fahrenheit, and you want to do the conversion calculations to centigrade yourself, you would apply the formula as follows:

$$\text{degrees centigrade} = 5/9 \times 78 - 32$$

Your result would be 25.6 degrees centigrade.

Although such formulas are easy for people to follow, a computer needs more structure. People understand the logic of formulas and recognize the proper order for calculating them just by looking at the formulas. A computer, however, must be told how to evaluate any arithmetic expression. As you've learned, nothing is "obvious" to the machine.

Therefore, the designers of BASIC built in a specific order for the computer to follow when it evaluates a formula. And, unless you tell the computer otherwise, that's the way the computer will perform the calculations. This built-in method is perfectly logical to the computer, but sometimes the logic is not so clear to you. Even sophisticated mathematicians can write programs that come up with the wrong answers because they don't follow the computer's logic.

If you remember the following seven simple rules for BASIC arithmetic, you will have no trouble:

1. Exponents (for example,  $b^2$ ) always are calculated first.
2. If there are two or more exponents, the calculation is done from right to left.
3. Multiplication and division always are performed next.
4. If there are two or more of these operations in the formula, the computer calculates from left to right.
5. Addition and subtraction always are performed last.
6. If there are two or more of these operations, the computer calculates from left to right.
7. If you must change BASIC's built-in order of calculation, place the changed part of the arithmetic expression in parentheses and the computer will evaluate the parenthetical expression first.

According to the seven rules of BASIC, we would calculate the temperature as follows:

1. Division first, or  $5/9 = 0.55555$
2. Multiplication next, or  $0.55555 \times 78 = 43.3333$
3. Subtraction last, or  $43.3333 - 32 = 11.33333$

In this case we didn't get the answer we wanted. To get the proper answer, we should have written the formula thus:

$$\text{centigrade} = 5/9 \times (\text{Fahrenheit} - 32)$$

Inserting the parentheses ensures that the subtraction is done first and that the computer gets the proper answer. With the parentheses in place, the calculations would be:

1. Contents of parentheses first, or  $78 - 32 = 46$
2. Division next, or  $5/9 = 0.5555$
3. Multiplication last, or  $0.5555 \times 46 = 25.555556$

Hard experience with the use of parentheses has taught us that when in doubt, *don't* leave them out. An additional set of parentheses doesn't hurt, but omitting necessary ones will give you the wrong answer.

## Logic In BASIC

When the computer first was invented, it was called a "logic machine." The truth is that the logical capabilities of the computer are quite limited. They consist of comparison operations performed by subtracting and sensing whether the results are positive, negative, or 0. With this ability, however, and simple programming on your part, the computer is capable of some amazing things. Let's take a look at logic in BASIC.

BASIC has six "relational" operators that compare two values. The results of the comparison are either *true* or *false*. The accompanying checklist shows the symbols for these relational operators and their meanings. You can connect these relational operators in BASIC expressions with the logical operators AND, OR, and NOT. If you use AND, *all* the comparisons must be true for the final result

to be true. If you use OR, the final result is true if *any* of the comparisons is true. If you use NOT, the results of the comparison first are evaluated and then reversed.



## BASIC Relational Operators

<i>Logic Symbols</i>	<i>Meaning</i>	<i>Example</i>	<i>Result</i>
A = B	A equals B	4 = 7	False
A <> B	A unequal to B	12 <> 9	True
or			
A >< B			
A < B	A less than B	15 < 49	True
A > B	A greater than B	67 > 22	True
A <= B	A less than or equal to B	44 <= 44	True
A >= B	A greater than or equal to B	90 >= 70	True

BASIC would evaluate the expression:

$$79 \leq 12 \text{ AND } 14 = 14$$

as false because  $79 \leq 12$  is false, even though  $14 = 14$  is true. The expression:

$$79 \leq 12 \text{ OR } 14 = 14$$

is true because one of the comparisons ( $14 = 14$ ) is true. The expression:

$$\text{NOT } 79 \leq 12 \text{ AND } 14 = 14$$

is true. Do you see why?  $\text{NOT } 79 \leq 12$  is true because 79 is more than 12, therefore the value of the comparison is false. The NOT operator reverses the false value and makes it true. Of course,  $14 = 14$  is true, so the final result is true.

All the logical expressions work with both numeric and string values. If you use the expression:

$$\text{CUSTOMER\$} = \text{"SMITH"}$$

the string value of the variable CUSTOMER\$ will be compared with the string constant "SMITH". The logical expression will be true if the string value in CUSTOMER\$ is "SMITH". Otherwise, it will be false. Note that BASIC requires string constants to be enclosed in quotation marks.

The logical expressions are used in the IF/THEN/ELSE statement to control the order of execution. The statement

```
200 IF ANSWER$ = "YES" THEN GOTO 700 ELSE 400
```

will send the program to statement number 700 if the value of the variable ANSWER\$ is the string "YES". Otherwise, it will send the program to statement 400.

## Displaying Output In BASIC

One of the strengths of BASIC is the variety of ways it allows you to display your output. In fact, BASIC is the best introductory language when it comes to data display. In this section we'll examine some of the statements that make BASIC so well suited for outputting results.

### The PRINT Statement

BASIC uses the PRINT statement, to which you were formally introduced in Chapter 4, to control output either to the screen or to the printer. The PRINT statement can include two types of elements: a list of one or more variable names to output data, and string constants to act as "labels." Labels explain the meaning of the data being printed. Each element in the PRINT statement must be separated from any other by a *cursor control character* (either a comma or a semicolon). The spacing on the display line is determined by these cursor control characters.

For example, if the value of the variable DEGREES is 26.6, the PRINT statement:

```
100 PRINT "THE TEMPERATURE IS";DEGREES; "CENTIGRADE"
```

will display

```
THE TEMPERATURE IS 26.6 CENTIGRADE
```

This example uses two labels: the strings "THE TEMPERATURE IS" and "CENTIGRADE". They are demarcated as strings by the use of quotation marks. The variable DEGREES is also part of the PRINT statement. Note that BASIC printed the value of the variable (that is, 26.6) and not the word DEGREES, because there were no quotation marks around the word DEGREES.

Each of the three elements of the PRINT statement (two labels and a variable name) is separated from one another by the cursor control character [;]—the semicolon. The semicolon causes BASIC to place the next item in the PRINT statement directly after the previous one.

Using the comma [,] between elements of the PRINT statement causes them to be displayed starting at column 1 and then starting at every fifteenth column thereafter.

If the string "JONES" is stored in the variable STUDENT\$ and GRADE has the value 89, the PRINT statements:

```
70 PRINT "NAME", "TEST SCORE"
80 PRINT
90 PRINT STUDENT$, GRADE
```

will display:

NAME	TEST SCORE
JONES	89

In this case the first item ("NAME") was displayed beginning at column 1 and "TEST SCORE" was started at column 15. PRINT statements containing no elements cause a blank line to be printed, in this case between the first line (NAME and TEST SCORE) and the third line (JONES and 89).

The PRINT statements:

```
900 PRINT "Budget", "Actual", "Year to Date"
910 PRINT "_____"
```

would give the following neatly spaced display:

Budget	Actual	Year to Date
<hr/>		

### The PRINT Statement With TAB And SPC

The normal display line is eighty columns wide. The screen can hold twenty-five lines and the printed page fifty-five lines. By using the PRINT statement and several other BASIC commands, you can place information anywhere on the screen or printed page.

The TAB function lets you specify the exact column you want the PRINT statement to begin printing. For example, TAB(1) means begin printing at column 1 and TAB(50) means begin printing at column 50. The TAB function, like any other element of the PRINT statement, is separated from the other elements by a cursor control key, usually the semicolon. Whatever follows the semicolon, whether it is a variable or a string, will be printed beginning in the location specified in the TAB function. The PRINT statements:

```
60 PRINT TAB(11);"ACCOUNTS RECEIVABLE AGING REPORT"
70 PRINT TAB(11);"_____"
80 PRINT TAB(5);"Customer";TAB(20);"Current Balance";
TAB(40);"Over 60 Days"
```

would display:

```
          ACCOUNTS RECEIVABLE AGING REPORT
          _____
Customer      Current Balance      Over 60 Days
```

The parameter in the TAB function can be a variable or an arithmetic expression, not just a number. This allows some interesting displays. The statements:

```
50 I = 10
60 PRINT TAB(I); "SUSAN"
70 I = I + 4
80 IF I < 26 THEN GOTO 60
```

would produce

```
  SUSAN
    SUSAN
      SUSAN
        SUSAN
```

As you can see, the TAB function is like the tab key on a typewriter. The SPC function is like the spacebar. The SPC function is similar to the TAB function except that it inserts the designated number of spaces into the display line. The statement:

```
80 PRINT SPC(10); "Print this here"
```

will begin the display in column 11 (ten spaces inserted at the start of the line). The advantage of this function over the TAB function is that the TAB function requires you to specify the exact position on the line where the PRINT statement is to go next (for instance, column 50), whereas the SPC function lets you designate spaces relative to the last item printed (that is, ten spaces from the beginning, or five spaces after the last letter, and so on).

### The LOCATE Statement

The LOCATE statement allows you to move the cursor to any place on the screen, unlike the TAB and SPC functions, which were restricted to the line currently being printed. The LOCATE statement is restricted to screen display, however; you cannot jump around the printed page with it. The parameters of the LOCATE statement are the line and column numbers of the cursor location. The statements:

```
420 LOCATE 12,37  
430 PRINT "CENTER"
```

will display the word CENTER in the middle of the screen (starting on line 12 and column 37).

Variables and arithmetic expressions can also be used as the parameters in the LOCATE statement. The BASIC program:

```
110 CLS  
120 PRINT "THIS DEMONSTRATES THE LOCATE STATEMENT"  
130 PRINT "ENTER ROW — A NUMBER FROM 1 TO 24";  
140 INPUT ROW  
150 PRINT "ENTER COLUMN — A NUMBER FROM 1 TO 80";  
160 INPUT COLUMN  
170 PRINT "THE ASTERISK WILL BE DISPLAYED IN THAT POSITION"
```



```

180 LOCATE ROW, COLUMN
190 PRINT "***"
200 END

```

illustrates the use of the LOCATE statement with variables.

### The PRINT USING Statement

The PRINT USING statement allows you to produce fancier reports and documents than could be printed with the combination of the PRINT statement, the TAB and SPC function, and the cursor control characters. The PRINT USING statement uses a *format string* to specify the way the elements in the statement are to be displayed.

For numeric data the format string controls the number of digits displayed and allows punctuation and labels to be inserted in the display line. With text data the format string establishes the size of the display field. If you ever wondered how the programmer wrote those programs that print names on class lists with only the first four letters of the first name, chances are this was done with a PRINT USING statement. The PRINT USING statement capability is fairly extensive, so we'll cover only part of it here. Once you have an idea of how it works, you can get the rest of the details from the BASIC manual.

Each pound sign (#) in the format string signifies that a digit of a numeric value is to be displayed. Decimal points, commas, and the dollar sign also can appear in the format string for a numeric value. Suppose you had a variable with a value of 2,000, but you wanted to print it as \$2,000.00. The format string "\$#,###.##" in the PRINT USING statement would do the job. Remember that because this is a string, it must be enclosed in quotation marks.

Here is another example. If  $A = 37.5$ ,  $B = 123.456$ , and  $C = 89,076.53$ , then:

```

90 PRINT USING "$#,###.## "; A, B, C

```

would display

```

$    37.50    $    123.46    $89,076.53

```

The values for each variable are modified to fit the format string—

that is, having two decimal places, a dollar sign at the beginning, and containing a comma if the value has four or more places to the left of the decimal point. The three blank spaces after the last # in the format string cause each value to be printed three spaces after the previous one.

Note that the PRINT USING statement rounds off decimal values (123.456 becomes 123.46). If there are not as many digits to the left of the decimal point in the value to be printed as are allocated by # symbols in the format string, that part of the field remains blank. On the other hand, to use the format string effectively, you must know the maximum number of digits in any value. Otherwise, some of the digits will not be shown or the extra digit can override the dollar sign.

For printing bank checks, \$ 37.50 is not satisfactory, as it might allow an unscrupulous individual to type in a few extra numbers, fattening the pocketbook. The PRINT USING statement allows a "floating" dollar sign, simply by placing back-to-back dollar signs in the format string. The statement:

**90 PRINT USING "\$\$##,###.##; A, B, C**

would yield:

\$37.50    \$123.46    \$89,076.53

As you can see, the dollar sign "floats" to the front of the number so that there is no blank space.

You can also use labels in the PRINT USING statement. The following statement illustrates this usage:

**120 PRINT USING "THE TOTAL COST IS \$\$##,###.##"; A**

(where the value of variable A is the same as that given earlier) would display:

THE TOTAL COST IS \$37.50

Just as we specified in the format string how many digits of a number are to be printed, we can specify how many characters of



```
140 F$ = "\  \ " + SPACES$(3) + "\  \ " + SPACES$(3) + "\  \ "
```

Here, there are three fields with backslash specifications separated by two strings of three blanks (five separate fields in all). The plus signs *concatenate* (link together) the separate fields to produce a single format string. Both versions of statement 140 will provide identical results, but the version with the SPACES\$ function is a little easier to read.

### Hardcopy Output

To get printed output, just use the letter L as the first letter of any of the PRINT statements. That is, LPRINT or LPRINT USING sends the results to the printer. Normally, the standard PRINT statement displays only on the monitor screen.

## Loops In BASIC

A *loop* is a series of statements that are repeated in the program. Loops of one size or another are found in almost every program more than a few statements long. In addition to the IF/THEN/ELSE statement you have already used, BASIC has two more sets of statements that make it easy to write loops. They are the WHILE/WEND statements and the FOR/NEXT statements.

When you program a loop you must:

1. Indicate the BASIC statements to be repeated. These statements are known as the "range" of the loop.
2. Establish a way to test when the loops should stop repeating. Otherwise the computer will go on doing the same sequence of statements over and over—forever.

The WHILE/WEND statement combination offers a number of advantages over the IF/THEN/ELSE choice. It makes the range of the loop much easier to see, and you don't have to keep track of statement numbers as you must with the IF/THEN/ELSE statement.

The WHILE statement is quite natural to use, because its operation is much like the instructions you might give a person to perform a task. You could say: "While the soufflé is in the oven, don't stomp around the kitchen." The WHILE/WEND statement works the same way. If the WHILE condition is true, the loop is

repeated. When the condition is false, the loop is broken and the statement after the WEND is executed.

The following program segment illustrates the WHILE/WEND statement:

```

40 I = 1
50 PRINT "NUMBER", "SQUARE", "CUBE"
60 PRINT "-----"
70 WHILE I <= 12
80 SQUARE = I * I
90 CUBE = SQUARE * I
100 PRINT I, SQUARE, CUBE
110 I = I + 1
120 WEND
130 PRINT "-----"

```

Loops have their own set of terminology. The logical expression in the WHILE statement is called the *condition*, and the statements between WHILE and WEND are called the *body of the loop*.

Statements 80 through 110 (the body of the loop) will be repeated until I becomes equal to 13. Then statement 130 will be executed. The results will be a table of squares and cubes of the first twelve integers.

The FOR/NEXT statement combination can be used effectively for loops that are repeated a specific number of times. The FOR statement does a lot of work for you. It sets up a "counter" that keeps track of how many times the loop has been performed and keeps adding to the counter each time through the loop. It also knows to stop the loop when the specified count has been reached. The FOR statement format is:

FOR index variable = start TO limit STEP increment

A typical FOR statement is:

```

200 FOR I = 1 TO 100 STEP 3

```

where I is the index variable (counter), 1 the start value, 100 the limit for I (ending count), and 3 the increment value (amount to increase the count each time through the loop). If the STEP specification is omitted from the FOR statement, the increment value automatically is set to 1.

The NEXT statement indicates the end of the loop. The format of the NEXT statement is:

NEXT index variable

The NEXT statement for the preceding FOR statement is:

**NEXT I**

When the program reaches the NEXT statement, the value of the index variable is examined. If the value is greater than the limit, the loop is completed and the statement after the NEXT statement is executed. Otherwise, the index variable is incremented and the loop is repeated. To add flexibility to the statement, the start, limit, and increment all can be variables or arithmetic expressions, as well as constants.

A ride on a few loops will clarify all this computerese. Here's a problem to solve with the use of FOR/NEXT statements. If you take a sheet of paper that is 1/1,000th of an inch thick and fold it twenty times, how thick will the resultant piece of paper be?

```
10 PAPER = 0.001
20 LIMIT = 20
30 FOR I = 1 TO LIMIT
40 PAPER = PAPER * 2
50 NEXT I
60 PRINT "The thickness after"; LIMIT ;"folds is "; PAPER;"inch(es)"
999 END
```

The results are surprising. Try the program yourself. Be careful that you input the program *exactly* as we have shown it here. Accuracy is extremely important in programming. An extra space or a missing symbol sometimes can keep a program from running correctly.

FOR statements can be *nested* inside one another. When this is done, the inner FOR loop runs to completion each time the outer FOR loop repeats. Imagine, if you will, two cogs in a giant machine. The cogs are set up so that the smaller cog must make one complete revolution to move the larger cog one notch. This is exactly what happens in a nested loop. For example, in:

```
10 PRINT "I", "J"
20 PRINT "-", "-"
```

```

30 FOR J = 1 TO 4
40 FOR I = 1 TO 3
50 PRINT I, J
60 NEXT I
70 NEXT J
999 END

```

the inner loop, statements 40 through 60, will run to completion and then begin again as the outer loop, statements 30 through 70, repeats. The results are:

I	J
1	1
2	1
3	1
1	2
2	2
3	2
1	3
2	3
3	3
1	4
2	4
3	4

## Subroutines In BASIC

A *subroutine* is a different kind of loop in a program. It is a set of statements that can be used repeatedly in the program without being redone each time. For example, in a program that prints a name and address at several different places, a printing subroutine can be a handy feature. The same subroutine can be used, or "called," by the program as many times as necessary. Therefore it is not necessary to write subroutines more than once, no matter how many times they are used in a program.

Subroutines are easy to use in BASIC. The subroutine is called into action with the GOSUB statement, which indicates the line number of the subroutine. The statement:

```
120 GOSUB 250
```

will send the program to the subroutine that starts at line 250.

Subroutines are terminated with a RETURN statement. The RETURN statement returns the program to the statement immediately following the GOSUB statement.

A program may have a "branch" point where several possible subroutines may be selected. BASIC handles this situation with the ON ... GOSUB statement. The format of this statement is:

ON index GOSUB line number, line number, ..., line number

The statement uses the value of the index to determine which subroutine to call. For example, the statement:

**200 ON I GOSUB 400, 500, 600, 700**

would call the subroutine at 400 if I was 1, the subroutine at 500 if I was 2, and so on. If the value of I is either 0 or greater than the number of subroutines in the statement, the ON ... GOSUB statement is ignored and the next statement in the program is executed. The index cannot be negative or greater than 255.

Some examples of the use of subroutines are shown in the following program:

```

10 PRINT "This program will hang some stars on your name"
20 PRINT : PRINT "Please enter your name";
30 INPUT NM$
40 PRINT : PRINT "How many stars do you deserve";
50 INPUT STARS
60 IF STARS > 0 AND STARS < 5 THEN GOTO 85
70 PRINT "Sorry, only one to four stars are allowed"
80 GOTO 50
85 PRINT SPC(34);
90 PRINT USING "\          \"; NM$
100 PRINT : PRINT TAB(37);"YOU GET"
110 ON STARS GOSUB 200, 300, 400, 500
120 PRINT : PRINT TAB(20); "CONGRATULATIONS—KEEP UP THE
GOOD WORK"
130 PRINT
140 GOTO 999
200 PRINT SPC(34);"** ONE STAR **"
210 RETURN
300 PRINT SPC(32);"*** TWO STARS ***"
310 RETURN
400 PRINT SPC(31);"**** THREE STARS ****"

```



```
410 RETURN
500 PRINT SPC(30);"**** FOUR STARS ****"
510 RETURN
999 END
```

## Placing Comments And Remarks In The Program

Even small programs can get complicated, and large ones can be quite difficult to follow. Therefore every programming language worth learning allows you to insert comments and remarks. BASIC offers two possible ways of doing this. The REM statement allows a remark or comment to be placed anywhere in the program. The apostrophe (') also allows the use of remarks and comments. For example:

```
100 REM THIS STATEMENT ALLOWS COMMENTS TO BE INSERTED
    IN THE PROGRAM
110 'THE APOSTROPHE ALSO ALLOWS COMMENTS
120 I = I + 1 'The apostrophe allows a comment right after a
    statement
```

These comments and remarks are ignored by the computer during the execution of a program. They just act as internal documentation. You should make liberal use of comments and remarks in your programs. They can be effective memory aids, and they are even more important if someone other than the programmer needs to make changes or corrections in a program.

If comments are so valuable, why did we wait so long to introduce the topic? There were two reasons: So far the programs have been straightforward, and we wanted you to get a good look at the way BASIC statements are used. From now on, though, look for lots of comments in the programs.

## Storing And Using Information With The DATA And READ Statements

The DATA statement in BASIC lets you store information within the program, and the READ statement allows you to use that stored information. The READ statement is similar in operation to the

INPUT statement (which we discussed in Chapter 4) in that it accepts values from the DATA statement, in order (one value for each variable listed in the READ statement). The DATA statement must follow its companion READ statement, but other than that it may be placed anywhere in the program.

An example will help illustrate the use of READ and DATA statements:

```

10 PRINT SPC(10);"NAME"; TAB(25): "AGE";SPC(5); "NUMBER
DEPENDENTS"
15 COUNT = 0
20 READ NM$, AGE, DEPENDENTS
30 IF NM$ = "XXXX" THEN GOTO 70
35 COUNT = COUNT + 1
40 F$ = "\          \ " + SPACES$(6) + "##"
+ SPACES$(11) + "##"
50 PRINT USING F$; NM$, AGE, DEPENDENTS
60 GOTO 20
70 PRINT
80 PRINT TAB(40);"A TOTAL OF"; COUNT; "RECORDS
PROCESSED"
90 GOTO 999
200 DATA "SLESLY JOHN", 35, 4
210 DATA "ETENER PAUL", 21, 0
220 DATA "O'BRIEN MARYANNE", 53, 3
230 DATA "GARCIA JUAN", 28, 3
235 DATA "WONG LEE", 29, 2
240 DATA "XXXX" , 0, 0
999 END

```

In this program the READ statement will read the list of names, ages, and number of dependents from the DATA statements and enter the values into the variables NM\$, AGE, and DEPENDENTS. The display will continue until the string "XXXX" is read. The COUNT variable acts as a counter to keep track of the number of records processed and is displayed at the end of the report.

The "XXXX" in the name field of the last data statement stops the processing and has a special name, *trip record*. Trip records are used with large data files to avoid the necessity of knowing the exact number of records in the data area when you write the program.

The computer uses a data pointer to keep track of the items

read from the DATA statements. When the data pointer reaches the end of the data in the DATA statements, and you attempt to perform another READ, an out-of-data error will occur. You can reset the data pointer to the first DATA statement at any time during the program by using the RESTORE statement. You can see the RESTORE statement in action in the following program:

```

10 'This program assists buyers in finding software for the PCjr
15 CLS 'clear screen
20 PRINT " SOFTWARE SELECTION NUMBERS "
30 PRINT " ***** "
40 PRINT "*** GAMES ..... 1 ***"
50 PRINT "*** WORD PROCESSING ..... 2 ***"
60 PRINT "*** SPREADSHEETS ..... 3 ***"
70 PRINT "*** DATA BASE SYSTEMS ..... 4 ***"
80 PRINT " ***** "
90 PRINT : PRINT "ENTER NUMBER ";
100 INPUT SELECTION
110 IF SELECTION >= 1 OR SELECTION <= 4 THEN GOTO 140
'check selection number
120 PRINT "THE SELECTION NUMBER SHOULD BE A 1, 2, 3, OR
4 - PLEASE ENTER ONE OF THESE";
130 GOTO 100 'return to selection entry
140 GOSUB 400 'transfer to selection processing
160 PRINT : PRINT "TO MAKE ANOTHER SELECTION ENTER Y,
OTHERWISE ENTER Q TO QUIT";
170 INPUT REPEAT$
180 IF REPEAT$ = "Y" OR REPEAT$ = "y" THEN GOTO 15 ELSE
GOTO 9999
400 'Selection Processing routine
410 RESTORE
420 READ TYPE, TITLE$
430 IF TYPE <> SELECTION THEN GOTO 420 'loop to get to
proper group
440 WHILE TYPE = SELECTION 'loop to complete listing
450 PRINT TITLE$
460 READ TYPE, TITLE$
470 WEND
480 PRINT : PRINT "          LIST COMPLETED"
490 PRINT
500 RETURN
1000 'Software name file

```

```
1010 DATA 1, "KEYBOARD ADVENTURE"  
1020 DATA 1, "SCUBADVENTURE"  
1040 DATA 1, "MOUSER"  
1050 DATA 1, "FLIGHT SIMULATOR"  
1060 DATA 2, "EASYWRITER"  
1070 DATA 2, "HOMEWORD"  
1075 DATA 2, "PEACHTEXT"  
1080 DATA 3, "VISICALC"  
1090 DATA 3, "MULTIPLAN"  
1100 DATA 4, "PFS: FILE"  
1110 DATA 4, "PFS: REPORT"  
8888 DATA 9, "XXXX"  
9999 END
```

## Arrays In BASIC

Thus far we have used independent variables or variables whose values are represented by a single number or string. Typically, in data processing, however, variables are related to each other—as fields in a record (like those we examined in the section on files) or mathematically (as part of a table or matrix).

BASIC lets you use the *array* structure to handle related variables. An array contains a number of related variables that share the *same* array name but still can be processed on an individual basis by the use of a subscript. The DIM statement declares an array name and sets aside storage space for it. The statement:

```
30 DIM MONTHLYSALES (12), STORENAME$ (3), BUDGETACTUAL  
(12,2)
```

declares that the variable MONTHLYSALES is to be composed of twelve separate but related numeric items, the variable STORENAME\$ is to be composed of three separate but related strings, and the variable BUDGETACTUAL is to be composed of twenty-four separate but related items structured as twelve rows of two columns each.

Each element in an array is accessed by using the array name and a subscript to indicate the specific element. The subscript is given (in parentheses) after the array name. The statements:

```

40 STORENAME$ (1) = "MARCYS"
50 STORENAME$ (2) = "WIDEWAY"
60 STORENAME$ (3) = "JUNES"

```

place the strings shown in each item of the array STORENAME\$. Subscripts can contain variables and arithmetic expressions, as well as constants. The statements:

```

70 FOR I = 1 TO 3
80 PRINT "THE NAME OF THE STORE IS ";STORENAME$ (I)
90 NEXT I

```

will display a list of the three values in the array STORENAME\$. The program segment:

```

90 FOR MONTH = 1 TO 12
100 PRINT "Enter the projected budget amount for the"; MONTH;
"month";
110 INPUT BUDGET
120 BUDGETACTUAL (MONTH,1) = BUDGET
130 PRINT
140 PRINT "Enter the actual amount spent for the month "; MONTH
150 INPUT DOLLARS
160 BUDGETACTUAL (MONTH,2) = DOLLARS
170 PRINT
180 NEXT MONTH

```

will accept entries from the keyboard to establish the contents of the array BUDGETACTUAL.

Statement 120 enters the value into the first of the two elements, whereas statement 150 enters the value into the second element. Each time the loop is executed, the index MONTH is incremented by 1 and the next set of the twelve pairs of items in the array BUDGETACTUAL can be entered.

The following program shows the use of arrays and DATA statements to create a budget control program.

#### ARRAY USAGE ILLUSTRATION

```

5 CLS : WIDTH 80 'clear screen and set width to 80 columns
10 PRINT " **THIS PROGRAM ILLUSTRATES THE USE OF ARRAYS**"
20 DIM STORES(3), MONTH$(12), ACTUALSALES(12,3), PROJECTEDSALES(12,3)
30 FOR I = 1 TO 3
40   READ STORES(I) 'places name of store in each array location
50 NEXT I
60 FOR I = 1 TO 12

```

```

70     READ MONTH$(I) 'places name of month in each array location
80 NEXT I
90 FOR I = 1 TO 3
100    FOR J = 1 TO 12
110      READ PROJECTEDSALES(J,I) 'places projected sales for store I and month J in array
120    NEXT J
130 NEXT I
135 PRINT : PRINT "DATA ENTRY SECTION" : PRINT
140 FOR I = 1 TO 3
150    FOR J = 1 TO 12
160      PRINT "ENTER SALES FOR "; STORES(I); " FOR MONTH "; MONTH$(J);
170      INPUT ACTUALSALES(J,I) 'sales for store I for month J
180    NEXT J
185    PRINT: IF I<3 THEN PRINT"PRESS ENTER TO CONTINUE": INPUT AN$: CLS
190 NEXT I
200 REM **PRINTOUT SALES AND PROJECTIONS TABLE FOR EACH STORE**
210 CLS: PRINT TAB(20); " SALES AND PROJECTIONS TABLE"
220 FOR I = 1 TO 3
230   PRINT:PRINT TAB(34); STORES(I)
240   PRINT:PRINT "MONTH", "ACTUAL SALES", "PROJECTED SALES"
250   PRINT
260     FOR J = 1 TO 12
270       PRINT MONTH$(J), ACTUALSALES(J,I), PROJECTEDSALES(J,I)
280     NEXT J
285   PRINT: IF I<3 THEN PRINT"PRESS ENTER TO CONTINUE": INPUT AN$: CLS
290 NEXT I
300 GOTO 999
400 REM **STORE NAMES**
410 DATA "MARCYS"
420 DATA "WIDEWAY"
440 DATA "JUNES"
500 REM **MONTHS**
510 DATA "JANUARY"
520 DATA "FEBRUARY"
530 DATA "MARCH"
540 DATA "APRIL"
560 DATA "MAY"
570 DATA "JUNE"
580 DATA "JULY"
590 DATA "AUGUST"
600 DATA "SEPTEMBER"
610 DATA "OCTOBER"
620 DATA "NOVEMBER"
630 DATA "DECEMBER"
700 REM**PROJECTED SALES DATA**
710 DATA 120000, 130000, 140000, 150000, 160000, 170000, 180000, 190000, 200000, 210000, 220000, 230000
720 DATA 40000, 60000, 80000, 100000, 100000, 110000, 110000, 110000, 110000, 120000, 140000, 160000, 180000
730 DATA 11100, 22200, 33300, 44400, 55500, 66600, 77700, 88800, 99900, 100000, 110000, 120000
999 END

```

## Files In BASIC

The DATA statement is valuable for storing information that remains constant between computer runs. However, if the information varies, the programmer must key any new data statements directly into the program. This arrangement is awkward, because it turns the programmer into a keyboard operator—a poor use of his or her time and expertise.

The file facilities in BASIC allow you to create and use files as a means of external storage for data, rather than relying only on the DATA statement to store data inside the program. Remember, a file is a collection of related data. Files are easy to use once you understand a few simple conventions and rules. Many of the file processing statements are similar to ones you have already used for display purposes.

Typical of BASIC, there are many different ways to obtain the same results with files. In this section we will focus on what is

considered the simplest, most understandable way. Two types of DOS files can be created and accessed by BASIC: sequential and random files.

Records on sequential files are stored in consecutive order—one after the other. To process any particular record, the computer must read each record from the beginning until the required record is reached. For example, if you want to read the 520th record, the computer must read through all 519 records before it. In addition, the fields or records on sequential files cannot be modified unless you rewrite the entire file.

This system is perfectly acceptable if you intend to process every record each time you use the file, such as when you are preparing paychecks, updating mailing lists, and so forth. However, if you want to treat records individually, without having to examine every preceding record on the file, you can use the random access method. Random access files allow you to go directly to any record on the file. Also, individual records can be modified without changing the entire file.

Obviously, the random access method can be much more convenient for many types of jobs. But because sequential files are easier to create and use in BASIC, let's start with them first.

## Opening And Closing Files

All files, both sequential and random, must be opened by an OPEN statement before you can use them and must be shut by a CLOSE statement when you are finished using them. Any files opened in the program must be closed before a program is terminated, or the data may be lost.

Opening a file in the computer is something like retrieving a paper file folder from a file cabinet. Closing the file is akin to returning the file folder and locking the cabinet. In BASIC files are referenced by a number. The OPEN statement sets up the connections between your DOS filename and the file number used to designate a file. The CLOSE statement severs these connections and ensures that all the data is sent back to the file.

When sequential files are opened, you must indicate whether they are going to be used for input or output. The OPEN statement:

**90 OPEN "ADDRESSES" FOR INPUT AS #1**

opens a DOS file named ADRESSES and allows the program to read data from this file. Any statements that use "ADRESSES" for input must now refer to it as file #1.

The statement:

```
200 OPEN "ADRESSES" FOR OUTPUT AS #3
```

opens "ADRESSES" to receive data from the program. Again, any statements that use "ADRESSES" must refer to it as file #3. A string variable can be used as the filename, and numeric variables or expressions can be used as the file number in the OPEN statement. For example,

```
60 F$ = "ADRESSES"  
70 FILENUMBER = 3  
80 OPEN F$ FOR OUTPUT AS FILENUMBER
```

will open "ADRESSES" as file #3.

The CLOSE statement is much simpler, and it can close one or more files. All it consists of is the keyword CLOSE # and the file number or numbers separated by commas. Again, a numeric variable or expression can be used to designate the file number(s). The statements:

```
390 TEXT = 4  
400 CLOSE #3, TEXT, 5
```

will close three files: numbers 3, 4, and 5.

The statement:

```
600 CLOSE
```

will close all the files opened in the program.

## Writing And Reading Data On Sequential Files

Three statements can be used to output data to sequential files: the WRITE # statement, the PRINT # and PRINT USING # statement. Two statements—the INPUT # statement and the LINE INPUT # statement—are used to input data from a file.



The WRITE # statement outputs data to a file in the format needed by the INPUT # statement. That is, commas are placed between numeric values and strings are enclosed with quotes. Use the WRITE # to place data on a file that will be read later as input by the INPUT # statement. Once the data is read in, you can process it as though it came from the ordinary INPUT statement. In other words, it makes no difference to the program whether the data came from the keyboard via an INPUT statement or from a DOS file on a diskette.

The following programs illustrate the use of the WRITE # and INPUT # statements.

```

10 'This program creates a sequential file
20 'of foods and their caloric values for an average serving (3 oz.)
30 OPEN "CALORIES" FOR OUTPUT AS #1
40 READ FOOD$, CALORIES
50 WHILE FOOD$ <> "XXXX"
60 WRITE #1, FOOD$, CALORIES 'writes data to file
70 READ FOOD$, CALORIES 'obtain values of data from DATA
statements
80 WEND
90 WRITE #1, "XXXX", 0 'puts trip record on file
100 CLOSE #1
110 GOTO 999
100 DATA "APPLES", 58
110 DATA "APRICOTS", 38
120 DATA "CHICKEN", 198
130 DATA "CHERRIES", 48
140 DATA "PEACHES", 31
150 DATA "CHOCOLATE", 520
160 DATA "BROWNIES", 420
170 DATA "XXXX", 0
999 END

```

This program reads and prints the contents of the CALORIES file.

```

10 'This program reads and prints the contents of the CALORIES
file
20 OPEN "CALORIES" FOR INPUT AS #1
30 INPUT #1, FOOD$, CALORIES 'read data
40 IF FOOD$ = "XXXX" THEN PRINT "NO DATA":GOTO 130

```

```

50 'check for trip record and empty file
60 PRINT "FOOD", "CALORIES"
70 PRINT "-----"
80 PRINT
90 WHILE FOOD$ <> "XXXX"
100 PRINT FOOD$, CALORIES
110 INPUT #1, FOOD$, CALORIES
120 WEND
130 CLOSE #1
140 PRINT
150 PRINT "File processing complete"
999 END

```

In these programs we used a trip record to detect the end of the data. That extra record is not absolutely necessary with files. BASIC provides an EOF function that signals when the end of the file is reached. The EOF(n) (where n is the number of the file) value is true when the end of file is reached. Otherwise, the end of file value is false. For example, we could use the statement:

```
50 WHILE <> EOF(1)
```

to continue the loop in a file processing program until the end of file is reached for file number 1.

The PRINT # and PRINT USING # statements operate in the same way as the PRINT and PRINT USING statements, except the data is sent to the designated file rather than to the screen. Normally, these two statements are used only to output data of text files that may have embedded commas or quotes.

The data from these files is read with the LINE INPUT # statement. The LINE INPUT # statement ignores commas and quotes and reads an entire line of data at one time. (A line is defined as all the text up to the Enter key symbol.) Each line can contain a maximum of 256 characters. The LINE INPUT statement, without a file number, can be used to obtain an entire line of data from the keyboard.

This example creates and displays a text file and illustrates the use of the LINE INPUT and PRINT # statements.

```

10 'This program creates, prints, and stores a memo
20 'Memos are at most 12 lines long

```

```

30 'Memos are named MEMOn, where n is the number of the memo
40 PRINT "ENTER MEMO NUMBER";
50 INPUT NUMBER$
60 FILENAME$ = "MEMO" + NUMBER$ 'concatenates number
to MEMO to create filename
70 OPEN FILENAME$ FOR OUTPUT AS #1
80 PRINT : PRINT "YOU MAY NOW ENTER A MEMO UP TO 12
LINES IN LENGTH"
90 PRINT "EVERY TIME YOU PRESS THE ENTER KEY COUNTS
AS A LINE"
100 PRINT "HOW MANY LINES DO YOU WANT";
110 INPUT LINES
120 IF LINES <= 12 THEN GOTO 150
130 PRINT "ONLY 12-LINE OR LESS MEMOS"
140 GOTO 100
150 PRINT "READY --- PLEASE ENTER MEMO"
160 FOR TEXTLINE = 1 TO LINES
170 LINE INPUT MEMO$ 'accept line from keyboard
180 PRINT #1, MEMO$ 'write line to file
190 NEXT TEXTLINE
200 CLOSE
210 PRINT : PRINT "WOULD YOU LIKE TO REVIEW THE MEMO? "
220 PRINT "IF SO ENTER Y FOR YES, TO QUIT ENTER Q";
230 INPUT ANSWER$
240 IF ANSWER$ <> "Y" AND ANSWER$ <> "y" THEN GOTO 999
250 OPEN FILENAME$ FOR INPUT AS #1 'now set up to read file
260 FOR TEXTLINE = 1 TO LINES
270 LINE INPUT #1, MEMO$ 'accept line from file
280 PRINT MEMO$ 'display line on screen
290 NEXT TEXTLINE
300 CLOSE
310 PRINT : PRINT "WOULD YOU LIKE TO WRITE ANOTHER MEMO?"
320 PRINT "IF SO ENTER Y FOR YES, TO QUIT ENTER Q";
330 INPUT ANSWER$
340 IF ANSWER$ <> "Y" AND ANSWER$ <> "y" THEN GOTO 999
350 CLS 'clear the screen
360 GOTO 40 'start again
999 END

```

## Adding Data To Sequential Files

A word of caution is needed here. You cannot simply open a sequential file and add data to it. When a sequential file is opened for

output, the computer assumes the file is either empty or filled with old, meaningless data. When the new output comes along, the existing contents of the file are destroyed. To add data without deleting the contents of the file, you must use the word APPEND, instead of INPUT or OUTPUT. For example, the OPEN statement to add data to our DOS file CALORIES is:

### **20 OPEN "CALORIES" FOR APPEND AS #1**

With this statement the file CALORIES is opened at the end of the current data and new data can be added without destroying the original data.

## **Error Handling**

When BASIC cannot find the filename you indicate in your program, an error condition results and the program is terminated. Sometimes, there is a good reason why the file is not present, especially in a single disk system like the PCjr. To avoid this error termination, use the ON ERROR statement. The ON ERROR statement lets you "trap" the error—that is, execute a subroutine when an error occurs. In this error-handling routine you can give the user instructions on any corrective action needed.

Normally, the ON ERROR statement is one of the first statements in the program, because BASIC requires that it be executed *before* the possible error can occur. The ON ERROR statement can handle many errors besides your forgetting to have the proper file present. The BASIC manual lists all the errors that BASIC will detect. See it for the details on errors and error messages. An example of the use of the ON ERROR statement is shown in the next program, Electronic Address Book. See if you can find it.

The Electronic Address Book program uses many of the statements we have covered in this section on BASIC. The program is longer than the example illustrated here, but not much more complicated. It uses sequential files to maintain the names, addresses, and telephone numbers. By now, you should be able to understand how this program was constructed and what each of the BASIC statements means. In fact, you should be able to write a program like this yourself. Are you game to try?

Keyboard the following program and run it. See if you can improve it or change it to suit your needs.

ELECTRONIC ADDRESS BOOK

```

5 'ELECTRONIC ADDRESS BOOK
10 'address and phone number program
15 'this program will handle 50 names and addresses
16 'name and address information is stored in a sequential file with 8
17 'string fields:      firstname, lastname, number & street, city, state,
18 '                   zip code, area code, phone number
20 'load addresses from file into memory
25 ON ERROR GOTO 3100 'enable error trapping to create file the first time
26 'the program is run
30 DIM RECS(50,8) 'array for storing names and addresses
40 OPEN "ADDRESS.DAT" FOR INPUT AS 1 'open file of addresses
50 LET NUM = 0 'field counter
60 WHILE NOT EOF(1) 'check for end of file
70   LET NUM = NUM + 1
80   FOR I = 1 TO 8
90     INPUT#1, RECS(NUM,I) 'read in record
100  NEXT I
110 WEND 'end of while loop
120 CLOSE #1 'close file
130 'main program loop
135 WHILE COMMAND$ <> "Q"
140 CLS 'clear screen
145 PRINT "*****"
150 PRINT "** ADDRESS AND PHONE NUMBER PROGRAM **"
155 PRINT "*****"
160 PRINT : PRINT
170 PRINT " |----- MENU -----| "
180 PRINT " | A for ADDING a new name | "
190 PRINT " | F for FINDING an address | "
200 PRINT " | P for PRINTING the whole list | "
210 PRINT " | D for DISPLAYING the whole list | "
220 PRINT " | Q for QUITTING when done | "
225 PRINT " |-----"
230 PRINT
235 INPUT "enter option:", COMMAND$
240 IF COMMAND$ = "F" THEN GOSUB 350 'find routine
250 IF COMMAND$ = "A" THEN GOSUB 680 'add new name routine
260 IF COMMAND$ = "P" THEN OPEN "LPT1:" FOR OUTPUT AS 1: GOSUB 1000 'output rtn
270 IF COMMAND$ = "D" THEN OPEN "SCRN:" FOR OUTPUT AS 1: GOSUB 1000 'output rtn
280 IF COMMAND$ <> "F" AND COMMAND$ <> "A" AND COMMAND$ <> "P" AND COMMAND$ <> "D" AND COMMAND$ <> "Q" THEN
PRINT "please type F, A, P, D, or Q":GOTO 235
284 WEND 'end of main program loop
285 'quit command routine
286 'replace address file on disk with new address file from memory
290 OPEN "ADDRESS.DAT" FOR OUTPUT AS 1 'construct new file
300 FOR I = 1 TO NUM
305   FOR J = 1 TO 8
310     WRITE#1, RECS(I,J) 'write 8 fields to file
315   NEXT J
320 NEXT I
330 CLOSE #1
340 GOTO 9999 'end program
350 'start of find routine
360 CLS
370 PRINT " Name finding routine" : PRINT
380 OPEN "SCRN:" FOR OUTPUT AS #1
390 FOUND = 0
400 NAMEKEY$(1) = ""
410 NAMEKEY$(2) = ""
420 INPUT "first name";NAMEKEY$(1)
430 INPUT " last name";NAMEKEY$(2)
440 IF NAMEKEY$(1) = "" AND NAMEKEY$(2) = "" THEN GOTO 560
450 IF NAMEKEY$(1) <> "" AND NAMEKEY$(2) <> "" THEN GOTO 520
460 IF NAMEKEY$(1) <> "" THEN FLD = 1
470 IF NAMEKEY$(2) <> "" THEN FLD = 2
480 FOR I = 1 TO NUM
490   IF NAMEKEY$(FLD) = RECS(I, FLD) THEN FOUND = 1: GOSUB 3000
500 NEXT I
510 GOTO 550
520 FOR I = 1 TO NUM
530   IF NAMEKEY$(1) = RECS(I,1) AND NAMEKEY$(2) = RECS(I,2) THEN FOUND = 1:GOSUB 3000
540 NEXT I
550 IF FOUND = 0 THEN PRINT "- name not on file"
560 CLOSE 1
565 PRINT : INPUT "To continue please press Enter", CONTINUE$
570 RETURN
580 'end of find routine

```

```

680 'start of add routine
685 CLS
690 PRINT " Name adding routine"
700 LET NUM = NUM + 1
710 INPUT "      first name";REC$(NUM,1)
720 INPUT "      last name";REC$(NUM,2)
730 INPUT "number and street";REC$(NUM,3)
740 INPUT "      city";REC$(NUM,4)
750 INPUT "      state";REC$(NUM,5)
760 INPUT "      zip code";REC$(NUM,6)
770 INPUT "      area code";REC$(NUM,7)
780 INPUT "      phone number";REC$(NUM,8)
790 'bubble sort - key is last name
800 FOR J = (NUM - 1) TO 1 STEP -1
810   LET F = 0
820   FOR I = 1 TO J
830     IF REC$(I,2) < REC$(I+1,2) THEN 880
840     FOR K = 1 TO 8
850       SWAP REC$(I,K), REC$(I+1,K)
860     NEXT K
870     LET F = 1
880   NEXT I
890   IF F = 0 THEN 910
900 NEXT J
910 RETURN
1000 'start output routine
1010 CLS
1020 PRINT "LIST OF ALL NAMES, ADDRESSES, AND PHONE NUMBERS"
1030 PRINT "*****"
1040 PRINT
1050 FOR I = 1 TO NUM
1060   GOSUB 3000           'print record
1070 NEXT I
1080 CLOSE 1
1085 PRINT : INPUT "To continue please press Enter key",CONTINUES
1090 RETURN
1100 'end output routine
2000 'check command entry
2010 IF COMMAND$ <> "A" AND COMMAND$ <> "F" AND COMMAND$ <> "P" AND COMMAND$ <> "D" AND COMMAND$ <> "E"
THEN PRINT "??": GOTO 230
2020 GOTO 235 ' return to command input
3000 'print record
3010 PRINT #1,USING "\\          \";REC$(I,1);           :PRINT#1," ";
3020 PRINT #1,USING "\\          \";REC$(I,2);           :PRINT#1," ";
3030 PRINT #1,USING "\\          \";REC$(I,3);           :PRINT#1," ";
3040 PRINT #1,USING "\\          \";REC$(I,4);           :PRINT#1," ";
3050 PRINT #1,USING "\\          \";REC$(I,5);           :PRINT#1," ";
3060 PRINT #1,USING "\\          \";REC$(I,6);           :PRINT#1," ";
3070 PRINT #1,USING "\\          \";REC$(I,7);           :PRINT#1," ";
3080 PRINT #1,USING "\\          \";REC$(I,8)           :PRINT#1," ";
3090 RETURN
3100 'end print record routine
3200 'error handling subroutine to initially create the address file
3210 IF ERR=53 THEN GOTO 3230 'error #53 is 'file not found' error
3220   ON ERROR GOTO 0           'print out other errors and stop execution
3230 OPEN "address.dat" FOR OUTPUT AS 1           'create file
3240 CLOSE 1
3250 RESUME           'return to load addresses into memory
9999 END

```

## Writing Random File Programs

Random files are more complicated and require more program steps than sequential files, but they offer some advantages in terms of speed and storage. We will present the basics of random files here, but we recommend that you experiment with simple programming projects first before you dive into the more complex problems using random files. Sequential files have been the backbone of data processing for over a decade, and they will serve most of your programming needs well. So don't worry if this section is too much

for you; you may never need to use random files, but it will be good for you to be exposed to them.

Before you write a program using random files, you must carefully think out your exact record structure. Random files store all data as string values in binary form. This saves space on the diskette but means you must convert back and forth between string values for storage and numeric values for processing. Special "make" functions are provided to change the numeric values to strings; "convert" functions change them back again. The programming is not hard, but it does require attention to detail. Random file programs involve the following steps:

1. Open the file as a random file and specify the length of the record. If you do not include a record size, it will be set automatically to 128 bytes.
2. Establish the composition of the record with the FIELD statement. This statement requires that you indicate the number of variables (fields) in the record, as well as their sizes and names. It allocates space in a random buffer that holds the variables until the record is written to the file.
3. Transfer the data from your program into the random buffer with the RSET and LSET statements.
4. Place the data in the random buffer on the diskette with the PUT statement.
5. Obtain data for the program from the file with the GET statement.

Some examples will illustrate these steps.

```
10 OPEN "RANDOMFILE" AS #1 LEN = 18
```

opens a random file with a length of eighteen bytes.

```
20 FIELD #1, 2 AS INTEGERS$, 12 AS STRNG$, 4 AS REALS
```

sets up three fields: one of two bytes for a whole number (integer), one of twelve bytes for a string, and one of four bytes for a real number (single precision).

```
30 INTEGER% = 20
```

```
40 LSET INTEGERS$ = MKI$ (INTEGER%)
```

These statements set INTEGER% equal to twenty. The percent sign after the variable name establishes it as an integer variable. Statement 40 makes the numeric value into a two-byte string and places it in the random buffer. The LSET statement sets the value in to the left. Numbers are LSET, and strings are RSET, or right set. The next series of statements sets the buffer for the other two variables.

```
60 RSET STRNG$ = "NORMAN"
```

Note that NORMAN is not twelve bytes long. The RSET statement pads the field to twelve bytes or truncates (cuts off) the string at twelve bytes.

```
80 REAL! = 1234.567 'the ! indicates REAL is single precision  
90 LSET REAL$ = MKS$ (REAL!) 'the MKS statement sets single  
precision numbers
```

The buffer is now set. The PUT statement places the data on the file in a specific record.

```
100 PUT 1, 50
```

The buffer was transferred to file number 1 as record 50. Both the file number and the record number can be numeric variables or expressions in the PUT statement.

As you can see, this programming is detailed and tedious but intellectually not too difficult. Now let's go the other way. Assume we have record number 30 on the file and it contains:

```
INTEGER$ = "46" , STRNG$ = "STRING VALUE", REAL$ = "99.99"
```

```
110 GET 1, 30
```

loads the buffer from the file; the three string variables are now defined.

```
130 INTEGER% = CVI (INTEGER$)
```

converts the string back to an integer for processing.



**140 REAL! = CVS (REAL\$)**

converts the string value back to a real single precision number.

You now have had a round trip on the random file express. The next ride is for a program example. All aboard!

In this section we'll use the example of an inventory file to illustrate the principles of random file programming. The inventory file will have a record that consists of three fields: an inventory number field, which will contain an integer; an inventory name field of twenty characters; and a dollar value field, which will be a single precision number.

The next two programs illustrate the use of random files. The first program sets up or, to use the computer science terminology, *initializes* the file. The second program uses the file as a basis for an inventory control program.

PROGRAM TO CREATE RANDOM ACCESS FILE

```

10 CLS
20 PRINT " This program sets up the file INVNTORY for use in the INVENTORY CONTROL program"
30 PRINT " The file contains 3 fields:"
40 PRINT " NUMBER (integer), DESC (20 characters), and VALUE (single precision)"
50 PRINT:PRINT " The file contains a maximum of 100 records"
60 PRINT:PRINT " ** WARNING ** if the INVNTORY file already exists, this program"
70 PRINT " will destroy the data -- DO NOT USE IT"
80 PRINT:PRINT " To start program enter S for START -- to QUIT enter Q";
90 INPUT ANSWERS
100 IF ANSWERS <> "s" AND ANSWERS <> "S" THEN PRINT "CANCELLED" : GOTO 260
110 OPEN "INVNTORY.DAT" AS #1 LEN=26
120 'length - 2 bytes for the integer, 20 bytes for the string, and 4 bytes
130 'for the single precision
140 FIELD #1, 2 AS NUMBERS, 20 AS DESC$, 4 AS VALUES
150 'set up random buffer
160 'initialize file with description field containing all X's and value = 0
170 INDESC$ = STRING$(20,"X") : VALUE = 0 : INTGR% = 0
180 FOR NUMBER% = 1 TO 100
190 RSET NUMBERS = MKIS(INTGR%) 'integer value for index
200 LSET DESC$ = INDESC$
210 RSET VALUES = MKS$(VALUE)
220 PUT #1, NUMBER%
230 NEXT NUMBER%
240 PRINT "file initialized"
250 CLOSE 1
260 END

```

RANDOM ACCESS FILE PROGRAM

```

10 'This program sets up and retrieves inventory records from a random file
20 'The inventory record has 3 fields: NUMBER (integer)
30 '                                     DESC (20 characters)
40 '                                     VALUE (single precision)
50 'The file contains a maximum of 100 records
60 OPEN "INVNTORY.DAT" AS #1 LEN=26
70 'Two bytes for the integer, 20 bytes for the string, and 4 bytes for the
80 'single precision value make the record 26 bytes long
90 FIELD #1, 2 AS NUMBERS, 20 AS DESC$, 4 AS VALUES
100 'set up random buffer

```

```

110 CLS 'clear screen
120 PRINT "*****"
130 PRINT "*** INVENTORY PROGRAM ***"
140 PRINT "*** MENU ***"
150 PRINT "***"
160 PRINT "*** A TO ADD ITEM ***"
170 PRINT "*** D TO DISPLAY LIST ***"
180 PRINT "*** F TO FIND ITEM ***"
190 PRINT "*** C TO CHANGE ITEM ***"
200 PRINT "*** Q TO QUIT ***"
210 PRINT "*****"
220 PRINT
230 PRINT "ENTER SELECTION";
240 INPUT SELECTION$
250 IF SELECTION$ = "A" THEN GOSUB 330
260 IF SELECTION$ = "D" THEN GOSUB 490
270 IF SELECTION$ = "F" THEN GOSUB 560
280 IF SELECTION$ = "C" THEN GOSUB 640
290 IF SELECTION$ = "Q" THEN 1290 'quit routine
300 IF SELECTION$ <> "A" AND SELECTION$ <> "D" AND SELECTION$ <> "F" AND SELECTION$ <> "C" THEN
PRINT "please enter an A, D, F, C, or Q" : GOTO 220
310 PRINT : INPUT "press return to continue",CONTINUE$
320 GOTO 110
330 'add routine
340 INPUT "ENTER PART NUMBER";NUMBER% 'percent sign indicates integer
350 GOSUB 1090 'check if on file
360 IF RANGEERROR THEN PRINT "out of range" : GOTO 340
370 IF ONFILE THEN PRINT "record number is already on file" : RETURN
380 INPUT "ENTER PART DESCRIPTION";DESCRIPTION$
390 IF DESCRIPTION$ = "" THEN 380
400 IF LEN(DESCRIPTION$) > 20 THEN PRINT "must be 20 characters or less" : GOTO 380
410 INPUT "ENTER PART VALUE";VALUE
420 'place the three fields in the buffer as strings
430 RSET NUMBER$ = MK$(NUMBER%)
440 LSET DESC$ = DESCRIPTION$
450 RSET VALUE$ = MK$(VALUE)
460 PUT #1, NUMBER%
470 PRINT "record placed on file"
480 RETURN
490 'display list routine
500 GOSUB 1190 'print headings
510 FOR NUMBER% = 1 TO 100
530 IF ONFILE THEN GOSUB 1230 'print item
540 NEXT NUMBER%
550 RETURN
560 'find item routine
570 INPUT "ENTER PART NUMBER";NUMBER%
580 GOSUB 1090 'check if on file
590 IF RANGEERROR THEN PRINT "out of range" : GOTO 570
600 IF NOT ONFILE THEN PRINT "record is not on file" : RETURN
610 GOSUB 1190 'print heading
620 GOSUB 1230 'print record
630 RETURN
640 'change routine
650 CLS
660 PRINT "change routine"
670 GOSUB 560 'find routine
680 IF NOT ONFILE THEN RETURN
690 PRINT
700 PRINT " change menu"
710 PRINT "-----"
720 PRINT " N to change number"
730 PRINT " D to change description"
740 PRINT " V to change value"
750 PRINT " R to return to main menu"
760 INPUT "enter choice >";CHOICES
770 IF CHOICES = "N" THEN GOSUB 830
780 IF CHOICES = "D" THEN GOSUB 960
790 IF CHOICES = "V" THEN GOSUB 1030
800 IF CHOICES = "R" THEN RETURN
810 IF CHOICES <> "N" AND CHOICES <> "D" AND CHOICES <> "V" THEN PRINT "please enter N, D, or V" : GOTO 760
820 GOTO 700
830 'change part number
840 INPUT "ENTER NEW PART NUMBER";NUMBER%
850 HOLDDESC$ = DESCRIPTION$
860 HOLDVALUE = VALUE
870 GOSUB 1090 'check if on file
880 IF RANGEERROR THEN PRINT "out of range" : GOTO 840
890 IF ONFILE THEN PRINT "new part number is already on file" : RETURN 'set choice to "R" to return to main menu
900 RSET NUMBER$ = MK$(NUMBER%) 'new record number
910 LSET DESC$ = HOLDDESC$ 'place old record in buffer
920 RSET VALUE$ = MK$(HOLDVALUE)
930 PUT #1, NUMBER%
940 PRINT "record updated"
950 RETURN

```

```

960 'change description
970 INPUT "ENTER PART DESCRIPTION";DESCRIPTIONS
980 IF DESCRIPTIONS = "" THEN 970
990 LSET DESC$ = DESCRIPTIONS
1000 PUT #1, NUMBER%
1010 PRINT "record updated"
1020 RETURN
1030 'change value routine
1040 INPUT "ENTER PART VALUE";VALUE
1050 RSET VALUE$ = MK$(VALUE)
1060 PUT #1, NUMBER%
1070 PRINT "record updated"
1080 RETURN
1090 'check file subroutine
1100 RANGEERROR = 0
1110 IF NUMBER% < 1 OR NUMBER% > 100 THEN RANGEERROR = -1 : RETURN
1120 GET #1, NUMBER%
1130 IF DESC$ = STRINGS(20,"X") THEN ONFILE = 0 : RETURN
1140 ONFILE = -1
1150 'move strings in buffer to variables
1160 DESCRIPTIONS = DESC$
1170 VALUE = CVS(VALUE$)
1180 RETURN
1190 'print headings subroutine
1200 PRINT "part number";SPC(8);"part description";SPC(10);"part value"
1210 PRINT "-----";TAB(18);STRINGS(20,"-");SPC(8);"-----"
1220 RETURN
1230 'print item subroutine
1240 PRINT TAB(5);
1250 PRINT USING "###"; NUMBER%;
1260 PRINT SPC(10) DESCRIPTIONS SPC(8);
1270 PRINT USING "$##,##.##"; VALUE
1280 RETURN
1290 'quit routine
1300 CLOSE #1
1310 CLS
1320 PRINT "program end - file closed"
1330 END

```

## Games And Graphics In BASIC

No discussion of BASIC programming would be complete without a fancy game program, so here's ours. It's a game of Roulette, and it has exciting graphics—in color. The program comes complete with lots of explanatory remarks, so play along and learn.

ROULETTE uses some graphics statements that are unfamiliar, so let's discuss them briefly. First of all, graphics statements and graphics programs are much more difficult than the standard text and numeric programs, so prepare to concentrate. The colorful results and "oohs and aahs" from your friends will make it well worth the effort.

### SCREEN Statement

BASIC allows you to control the screen colors and resolution, so you can design professional-quality graphics. The SCREEN statement is the BASIC statement that lets you specify (by means of several different coded parameters) the screen colors, resolution quality, and the graphics page being displayed.

The color/graphics monitor and adapter of the IBM PCjr are very versatile. By using the SCREEN statement, you can control the mode of operation, the color, alternate images on the screen, and even regulate the amount of memory committed to graphics. ROULETTE, like any computer game worth its salt, makes use of a number of these powerful features.

There are seven possible modes of operation, each offering a different level of detail and color. Naturally, you will have more flexibility in color display if you have all seven modes at your disposal. Cartridge BASIC on the PCjr supports all seven modes.

The Accompanying SCREEN Statement Checklist lists the seven possible modes of operation and the colors they support.



### *SCREEN Statement Summary*

<i>Mode</i>	<i>Type</i>	<i>Columns</i>	<i>Page Size</i>	<i>Colors</i>
0	Text	40 or 80	2Kb or 4Kb	16
1	Medium-resolution	40	16 Kb	4
2	High-resolution	80	16 Kb	4
3	Low-resolution	20	16 Kb	16
4	Medium-resolution	40	16 Kb	4
5	Medium-resolution	40	32 Kb	16
6	High-resolution	80	32 Kb	4

### **COLOR Statement**

The COLOR statement describes the colors used in your program. There are sixteen different colors available (see the accompanying checklist), but you can't use them all in every mode. For example, in the text mode, you can choose one of sixteen colors for the characters in the foreground and one of eight for the background. You can even select from eight separate colors for the border (the

area of the screen not used to display text) to vary your effects. You can even blink the foreground characters to create some stunning visual effects. Notice how we use this blinking capability in the ROULETTE program.

This color business is really not nearly as complicated as it appears. Conquering the world of color graphics is more a matter of detail than difficulty. As you will soon see, the concepts themselves are easy to grasp. And creating visual excitement is its own reward.



### *Available Colors And Codes*

---

0 Black	8 Gray
1 Blue	9 Light Blue
2 Green	10 Light Green
3 Cyan	11 Light Cyan
4 Red	12 Light Red
5 Magenta	13 Light Magenta
6 Brown	14 Yellow
7 White	15 High-intensity White

To cause color to blink add 16 to value

### **Drawing Lines**

The numbers in game programs can be constructed with LINE statements, which are easy to use. Just specify the beginning and end of the line and the color.

### **Random Numbers**

Games of chance (and a great many other things in life) depend on random events. The RANDOMIZE statement and the RND function in BASIC combine to produce random numbers (a set of numbers between 0 and 1 in which any number has an equal probability of appearing). Actually, these numbers produced by the computer are not completely random. They are pseudorandom numbers, but they are satisfactory for all practical purposes.

When you are working with truly random events, almost anything is possible—which means that debugging your program could be nearly impossible. Therefore the RND function is designed to produce the same sequence of random numbers each time it is used. This facilitates testing. But of course it's much too predictable for games because the same random sequence would be repeated every time you ran the program. To avoid this problem, the RANDOMIZE statement accepts a "seed," which alters the sequence of the random numbers that are generated. As a result the random numbers can vary from run to run, provided a different seed is used. To make it easier to generate a different seed for each run, the time of day can be obtained from the computer's built-in clock and used as the seed. See if you can discover where the RANDOMIZE statement and the RND function are used in our ROULETTE program.

### String And Character Functions

Games and text processing programs often make use of BASIC's capability to extract parts of strings or convert a string value into a numeric value for calculations. The use of the TIME\$, VAL, and RIGHT\$ statements to produce a seed for the random number function in ROULETTE is a typical example (see statement 140). Here the TIME\$ statement produces the current time in the standard DOS format. The RIGHT\$ statement extracts the seconds (the rightmost two digits), and the VAL statement converts the string digits into a numeric value.

The CHR\$ statement is another handy BASIC statement used in game and other special-purpose programs. Remember in Chapter 2 when we discussed the use of a numeric code to represent the characters displayed by the PCjr? The CHR\$ statement takes a number and returns the PCjr character equivalent. This capability is needed because the keyboard does not permit the direct entry of many of the unusual characters you can use in building a game display. Appendix G of the BASIC manual gives a complete list of these characters. The following ROULETTE program uses this statement to display some special characters.

#### ROULETTE PROGRAM

```
10 'roulette game
20 KEY OFF 'turn off soft key display
30 FLAG = 1
40 SCREEN 0,,0 : WIDTH 80 ' set text mode and width to 80 columns
```

```

50 CLS          'clear screen
60 FIND = 1
70 TOTAL = 0
80 DIM ROULETTE(38,4) 'array to store order and color of roulette numbers
90 'load roulette wheel information into memory from data statements
100 FOR I = 1 TO 38
110 READ ROULETTE(I,1), ROULETTE(I,2), ROULETTE(I,3), ROULETTE(I,4)
120 NEXT
130 'use number from system time to reseed random number generator
140 RANDOMIZE VAL(RIGHT$(TIMES,2))
150 'introduction routine
160 GOSUB 1130      'build wheel
165 'display "ROULETTE" built from ASCII special character number 178
170 X = 10
180 FOR I = 1 TO 6
190 LOCATE X+I-1,21
200 ON I GOSUB 310,320,330,340,350,360      'print line
205 'change color of "ROULETTE"
210 NEXT
220 X = 10
230 FOR J = 1 TO 7
240 COLOR J,0
250 FOR I = 1 TO 6
260 LOCATE X+I-1,21
270 ON I GOSUB 310,320,330,340,350,360      'print line
280 NEXT
290 NEXT
300 GOTO 370
310 PRINT STRINGS(4,178);" ";STRINGS(4,178);" ";CHR$(178);" ";CHR$(178);" ";
CHR$(178);SPC(4);STRINGS(4,178);" ";STRINGS(4,178);" ";STRINGS(4,178);" ";STRINGS(4,178);
: RETURN
320 PRINT CHR$(178);" ";CHR$(178);" ";CHR$(178);" ";CHR$(178);" ";CHR$(178);" ";
CHR$(178);" ";CHR$(178);SPC(4);CHR$(178);SPC(4);CHR$(178);" ";CHR$(178); : RETURN
330 PRINT STRINGS(4,178);" ";CHR$(178);" ";CHR$(178);" ";CHR$(178);" ";CHR$(178);" ";
CHR$(178);SPC(4);CHR$(178);SPC(6);CHR$(178);" ";CHR$(178);" ";CHR$(178);" ";CHR$(178); : RETURN
340 PRINT STRINGS(2,178);SPC(3);CHR$(178);" ";CHR$(178);" ";CHR$(178);" ";
CHR$(178);" ";CHR$(178);SPC(4);STRINGS(3,178);SPC(4);CHR$(178);SPC(4);CHR$(178);" ";STRINGS(3,178); : RETURN
350 PRINT CHR$(178);" ";CHR$(178);" ";CHR$(178);" ";CHR$(178);" ";CHR$(178);" ";
CHR$(178);" ";CHR$(178);SPC(4);CHR$(178);SPC(6);CHR$(178);SPC(4);CHR$(178);" ";CHR$(178); : RETURN
360 PRINT CHR$(178);" ";CHR$(178);" ";STRINGS(4,178);" ";STRINGS(4,178);" ";
STRINGS(4,178);" ";STRINGS(4,178);SPC(3);CHR$(178);SPC(4);CHR$(178);" ";STRINGS(4,178); : RETURN
370 'menu routine
380 SCREEN ,,0,0
390 CLS
400 LOCATE 4,1
410 PRINT TAB(24) "-----"
420 PRINT TAB(24) "|          BETS          |"
430 PRINT TAB(24) "-----"
440 COLOR 0,7 : GOSUB 920 : COLOR 7,0 'print first choice in inverse
450 GOSUB 930
460 GOSUB 940
470 GOSUB 950
480 GOSUB 960
490 GOSUB 970
500 GOSUB 980
510 GOSUB 990
520 GOSUB 1000
530 GOSUB 1010
540 GOSUB 1040
550 LOCATE 15,7
560 PRINT STRINGS(33,"-");
570 LOCATE 15,45
580 PRINT STRINGS(9,"-");
590 LOCATE 15,60 : PRINT "-----";
600 LOCATE 16,20 : PRINT "1:1";
610 LOCATE 16,48 : PRINT "2:1";
620 LOCATE 16,61 : PRINT "35:1";
630 LOCATE 17,33 : PRINT "PAYOFF";
640 LOCATE 25,12 : PRINT "use space bar to select bet then press ENTER";
650 'get bet type from user
660 CHOICE = 1
670 CHARS$ = ""
680 WHILE CHARS$ <> CHR$(13) '13 is code for ENTER key
690 CHARS$ = INKEYS
700 IF CHARS$ <> CHR$(32) THEN 760 '32 is code for space bar
710 ON CHOICE GOSUB 920,930,940,950,960,970,980,990,1000,1010,1040
720 IF CHOICE = 11 THEN CHOICE = 1 ELSE CHOICE = CHOICE + 1
730 COLOR 0,7 'inverse
740 ON CHOICE GOSUB 920,930,940,950,960,970,980,990,1000,1010,1040
750 COLOR 7,0 'white on black
760 WEND
770 CLS
780 IF CHOICE = 11 THEN GOTO 2420 'end routine
790 LOCATE 10
800 IF CHOICE <> 10 THEN GOTO 870

```

```

810 PRINT TAB(10);INPUT "which number (0 thru 36, or 00)";BETNUM$
820 IF BETNUM$ = "00" OR (VAL(BETNUM$) >= VAL("0") AND VAL(BETNUM$) <= VAL("36")) THEN 860
830 'invalid entry
840 PRINT TAB(10) "please enter a number 0 thru 36 or double zero"
850 GOTO 810
860 PRINT
870 PRINT TAB(10) "how much do you want to bet";:INPUT BETAMOUNT
880 IF BETAMOUNT <= 0 THEN PRINT TAB(10) "** you must bet a positive number" : GOTO 870
890 CLS
900 GOTO 1050
910 'end of menu routine
920 LOCATE 11,7 : PRINT "REDS"; : RETURN
930 LOCATE 13,7 : PRINT "BLACKS"; : RETURN
940 LOCATE 11,17 : PRINT "ODDS"; : RETURN
950 LOCATE 13,17 : PRINT "EVENS"; : RETURN
960 LOCATE 11,27 : PRINT "1ST EIGHTEEN"; : RETURN
970 LOCATE 13,27 : PRINT "2ND EIGHTEEN"; : RETURN
980 LOCATE 10,45 : PRINT "1ST DOZEN"; : RETURN
990 LOCATE 12,45 : PRINT "2ND DOZEN"; : RETURN
1000 LOCATE 14,45 : PRINT "3RD DOZEN"; : RETURN
1010 LOCATE 11,60 : PRINT "SINGLE";
1020 LOCATE 12,60 : PRINT "NUMBER";
1030 LOCATE 13,60 : PRINT " BET "; : RETURN
1040 LOCATE 22,34 : PRINT "- exit -"; : RETURN
1050 'get random number in range of 0 to 37 (37 is for double zero)
1060 RNUMBER = INT(RND*38)
1070 'spin wheel all they way around then stop at rnumber on the second loop
1080 SCREEN ,,1,1
1090 IF FLAG = 1 THEN CLS : GOSUB 1130 : FLAG = 0 'build wheel first time
1100 IF FIND = 1 THEN FIND = 0 ELSE FIND = 1
1110 GOTO 1330 'spin wheel (when wheel stops, program branches to found rtn
1120 '
1130 'build wheel routine
1140 FOR J = 1 TO 38
1150 LOCATE ROULETTE(J,3), ROULETTE(J,4)
1160 COLOR 7, ROULETTE(J,2) 'set background to red,green, or black
1170 GOSUB 1210 'print box
1180 NEXT
1190 COLOR 7,0 'color white on black
1200 RETURN
1210 'print box routine
1220 X = CSRLIN
1230 Y = POS(0)
1240 PRINT CHR$(201);STRING$(2,205);CHR$(187);
1250 LOCATE X+1,Y
1260 PRINT CHR$(186);
1270 IF ROULETTE(J,1) = 100 THEN PRINT "00"; : GOTO 1290
1280 PRINT RIGHT$(STR$(ROULETTE(J,1)),2);
1290 PRINT CHR$(186);
1300 LOCATE X+2,Y
1310 PRINT CHR$(200);STRING$(2,205);CHR$(188);
1320 RETURN
1330 'spin wheel routine
1340 FOR J = 1 TO 38
1350 SOUND 37,2
1360 LOCATE ROULETTE(J,3),ROULETTE(J,4)
1370 IF FIND = 0 THEN 1430
1380 IF ROULETTE(J,1) <> RNUMBER AND NOT (ROULETTE(J,1) = 100 AND RNUMBER = 37) THEN 1430
1390 'wheel is on the winning number
1400 COLOR ROULETTE(J,2)+16,7 'set color to blinking
1410 GOSUB 1210 'print box
1420 GOTO 1500 'found routine
1430 COLOR ROULETTE(J,2),7 'colored foreground on white
1440 GOSUB 1210 'print box
1450 LOCATE ROULETTE(J,3),ROULETTE(J,4)
1460 COLOR 7,ROULETTE(J,2) 'white on colored background
1470 GOSUB 1210 'print box
1480 NEXT J
1490 GOTO 1100 'go back to set find flag and spin wheel again
1500 'found routine
1510 COLOR 7,0
1520 WIN = 0
1530 ON CHOICE GOSUB 1810,1830,1850,1870,1890,1910,1930,1950,1970,1990
1540 IF WIN = 1 THEN 1590
1550 LOCATE 10,33
1560 PRINT "sorry, you lose";
1570 TOTAL = TOTAL - BETAMOUNT
1580 GOTO 1640
1590 LOCATE 10,37
1600 PRINT "you win!";
1610 IF CHOICE >= 1 AND CHOICE <= 6 THEN TOTAL = TOTAL + BETAMOUNT
1620 IF CHOICE >= 7 AND CHOICE <= 9 THEN TOTAL = TOTAL + BETAMOUNT * 2
1630 IF CHOICE = 10 THEN TOTAL = TOTAL + BETAMOUNT * 35
1640 LOCATE 15,31
1650 IF TOTAL = 0 THEN PRINT "you are even"; : GOTO 1680

```



```

1660 IF TOTAL < 0 THEN PRINT "you have lost "; ELSE PRINT "you have won ";
1670 PRINT USING "$$##,###.###";ABS(TOTAL);
1680 LOCATE 25,31
1690 PRINT "press any key to continue";
1700 A$ = ""
1710 A$ = INKEYS : IF A$ = "" THEN 1710
1720 'clear text from center of wheel and turn off blinking box
1730 LOCATE 10,30 : PRINT SPC(20);
1740 LOCATE 15,30 : PRINT SPC(35);
1750 LOCATE 25,30 : PRINT SPC(30);
1760 LOCATE ROULETTE(J,3),ROULETTE(J,4)
1770 COLOR 7,ROULETTE(J,2)
1780 GOSUB 1210 'print box routine
1790 COLOR 7,0 'color white on black
1800 GOTO 370 'go back to menu for next bet and spin
1810 IF ROULETTE(J,2) = 4 THEN WIN = 1
1820 RETURN
1830 IF ROULETTE(J,2) = 0 THEN WIN = 1
1840 RETURN
1850 IF ROULETTE(J,2) <> 2 AND ROULETTE(J,1) MOD 2 <> 0 THEN WIN = 1
1860 RETURN
1870 IF ROULETTE(J,2) <> 2 AND ROULETTE(J,1) MOD 2 = 0 THEN WIN = 1
1880 RETURN
1890 IF ROULETTE(J,1) > 0 AND ROULETTE(J,1) <= 18 THEN WIN = 1
1900 RETURN
1910 IF ROULETTE(J,1) >= 19 AND ROULETTE(J,1) <= 36 THEN WIN = 1
1920 RETURN
1930 IF ROULETTE(J,1) > 0 AND ROULETTE(J,1) <=12 THEN WIN = 1
1940 RETURN
1950 IF ROULETTE(J,1) >= 13 AND ROULETTE(J,1) <= 24 THEN WIN = 1
1960 RETURN
1970 IF ROULETTE(J,1) >= 25 AND ROULETTE(J,1) <= 36 THEN WIN = 1
1980 RETURN
1990 IF (ROULETTE(J,1) = VAL(BETNUM$) OR (ROULETTE(J,1) = 100 AND BETNUM$ = "00")) THEN WIN = 1
2000 RETURN
2010 'roulette wheel information; each data statement contains:
2015 ' 1) the roulette box number
2018 ' 2) the IBM BASIC color code for black(0), green(2), or red(4)
2020 ' 3) the row coordinate for the screen location for that number
2025 ' 4) the column coordinate for the screen location for that number
2030 DATA 100,2,1,39
2040 'double zero box is '100' to differentiate it from the single zero box
2050 DATA 1,4,2,43
2060 DATA 13,0,3,47
2070 DATA 36,4,4,51
2080 DATA 24,0,5,55
2090 DATA 3,4,6,59
2100 DATA 15,0,7,63
2110 DATA 34,4,8,67
2120 DATA 22,0,9,71
2130 DATA 5,4,10,75
2140 DATA 17,0,13,75
2150 DATA 32,4,14,71
2160 DATA 20,0,15,67
2170 DATA 7, 4,16,63
2180 DATA 11,0,17,59
2190 DATA 30,4,18,55
2200 DATA 26,0,19,51
2210 DATA 9,4,20,47
2220 DATA 28,0,21,43
2230 DATA 0,2,22,39
2240 DATA 2,0,21,35
2250 DATA 14,4,20,31
2260 DATA 35,0,19,27
2270 DATA 23,4,18,23
2280 DATA 4,0,17,19
2290 DATA 16,4,16,15
2300 DATA 33,0,15,11
2310 DATA 21,4,14,7
2320 DATA 6,0,13,3
2330 DATA 18,4,10,3
2340 DATA 31,0,9,7
2350 DATA 19,4,8,11
2360 DATA 8,0,7,15
2370 DATA 12,4,6,19
2380 DATA 29,0,5,23
2390 DATA 25,4,4,27
2400 DATA 10,0,3,31
2410 DATA 27,4,2,35
2420 'end routine
2430 SCREEN ,,0,0
2440 KEY ON
2450 END

```

### **An Additional Note—Sound And Play Statements**

BASIC has some interesting and powerful statements to add the extra dimension of sound to your programs. A simple warning or attention-getting beep can be produced with the BEEP statement. For example,

```
40 BEEP
```

will sound a short beep when it is encountered in the program.

The SOUND statement, however, is much more powerful. You can select both the frequency and duration of a tone. For example, the statement:

```
40 SOUND 440, 27
```

produces the old musician's standard—A above middle C—for about one and a half seconds (long enough to tune your instrument). Your BASIC manual gives you the frequency values for the notes to use with the SOUND statement. The duration is measured in beats, with 1,092 beats per minute (18.2 per second). As usual, you can use a numeric variable or expression in the SOUND statement. In the ROULETTE program, the SOUND statement:

```
1350 SOUND 37,2
```

is used to simulate the clicking of the roulette ball.

With advanced BASIC (BASICA), you can use the PLAY statement. This statement lets you write an entire sequence of notes to play tunes with your PCjr. The statement even allows "subtunes" in the same way you use subroutines—for repeated phrases and motifs. Couple this with your external speaker connection on the PCjr and you can really have fun with the music supplied by your own programs. Refer to the *BASIC Reference Manual* for the details.

### **More Fun And Games—The DOS 2.10 Supplementary Programs**

An excellent library of BASIC programs is yours for free. It is the collection of BASIC programs on the DOS 2.10 Supplementary Pro-

grams diskette. Use them in two ways—for fun and games, definitely, but also as a reference library of programming techniques. These are programs written by experts especially to show off the features of your computer. They contain some clever and elegant BASIC programming. A number of years ago we interviewed one of the best and financially most successful programmers in the business. When we asked for advice on how to learn the art of programming, he replied, "It's easy, just read good programs and learn from them."

We have provided you with some good programs, and here on the Supplementary Programs diskette are several more. Use the programs. Especially examine the displays they create. Writing statements for displays is one of the most time-consuming parts of programming games. Then, list the programs and see what statements the programmer used. Break up the program and see how the different groups of statements work. Look up new and unfamiliar statements in the *BASIC Reference Manual*. In a relatively short time you can learn a tremendous amount about advanced BASIC programming. The Music program will give you a lesson in how the PLAY statement is used in a practical situation. The Art program illustrates some basic graphics programming concepts.

An easy way to take a "course" in Advanced BASIC is to load and run the Samples program on the diskette. Then take your choice of the goodies IBM offers you for free. Have fun and learn.

## **BASIC System Commands—A Summary**

System commands are an important part of the BASIC language. You already have used the most common commands. The accompanying checklists give a brief description of the entire set of BASIC system commands.

We have learned through hard experience that knowing a few commands well is far better than knowing a large number superficially. Try to master the commands you have already found useful. Then, and only then, enlarge your command vocabulary by experimenting with a few new commands each time you write a program. Eventually, you will be comfortable using all the commands on this list. In the meantime you'll feel more confident if you build up your command vocabulary slowly but surely.



## ***Summary Of BASIC Statements***

---

**BEEP** Sounds internal speaker

**CALL** Obtains machine-language subroutine

**CHAIN** Runs new program but maintains current variable values

**CHDIR** Changes directory

**CIRCLE** Creates circle, arc, ellipse, or wedge

**CLOSE** Closes connection with data or communication file(s)

**CLS** Clears monitor screen

**COLOR** Establishes foreground, background, and border colors

**COM** Turns data flow from communications adapter on or off

**COMMON** Passes variable values to a CHAINED program

**DATA** Stores numeric and string constants in a program

**DATE\$** Displays or sets date

**DEF FN** Establishes a user-defined function

**DEF SEG** Indicates a location in memory

**DEftype** Declares variable names beginning with a designated letter to be of the indicated type

**DEF USR** Sets the starting address for a machine-language subroutine

**DIM** Establishes size and storage for arrays

**DRAW** Creates an image as defined by a string

**END** Ceases program execution and closes files

**ERASE** Deletes arrays and returns storage to program

**ERROR** Creates error conditions

**FIELD** Sets up fields for random access files

**FOR** Establishes a counting loop

**GET** Obtains a record from a random access file or reads data from the screen into an array

**GOSUB** Transfers to subroutine

**GOTO** Transfers to specific line

**IF** Bases action on value of logical expression



## *Summary Of BASIC Statements (Continued)*

---

**INPUT** Accepts data from keyboard or file

**KEY** Sets function keys

**KEY [ON/OFF/STOP]** Sets up or terminates monitoring function or cursor control key activity

**LET** Assigns value to variable

**LINE** Produces a line or box on the screen

**LINE INPUT** Accepts an entire line at once

**LOCATE** Positions the cursor on the screen

**LPRINT** Sends output to printer

**LPRINT USING** Sends output to printer using format control

**LSET** Left-justifies data in random file buffer

**MID\$** Replaces part of a string with another string

**MKDIR** Sets up directory

**MOTOR** Turns cassette off or on

**NEXT** Terminates FOR loop

**ON [COM/KEY/PEN/STRIG] GOSUB** Transfers to subroutine when designated action occurs

**ON ERROR GOTO** Transfers to designated line when error occurs

**ON GOSUB** Transfers to subroutine as defined by value of variable or expression

**ON GOTO** Transfers to line as defined by value of variable or expression

**OPEN** Establishes connection to designated data or communication file

**OPTION BASE** Permits array subscripts to begin at either 0 or 1

**OUT** Transmits byte to designated output port

**PAINT** Covers designated area with specified color

**PEN [ON/OFF/STOP]** Accepts or disables input from light pen

**PLAY** Produces tune based on string of data

**PMAP** Relates physical and world coordinates



## *Summary Of BASIC Statements (Continued)*

---

- POKE** Places a designated byte value in a location in memory
- PRESET** Places a point in a designated location on the screen in a specified color or background color (see also PSET)
- PRINT** Places data on the screen or in a file
- PRINT USING** Places data on the screen or in a file using a format control string
- PSET** Places a point in a designated location on the screen in a specified color (see also PRESET)
- PUT** Places a record in a random access file or places a stored image on the screen
- RANDOMIZE** Initiates a new sequence of pseudorandom numbers
- READ** Obtains data from DATA statement
- REM** Inserts comment in program
- RESTORE** Initializes data pointer in DATA statement
- RESUME** Continues processing after an error is detected and corrected
- RETURN** Returns to main program after subroutine
- RMDIR** Removes directory
- RSET** Right-justifies data in random file buffer
- SCREEN** Selects mode for display
- SOUND** Produces tone from speaker
- STOP** Ceases program execution
- STRIG** [ON/OFF/STOP] Accepts or disables input from joystick buttons
- SWAP** Interchanges values of two variables
- TIME\$** Displays or sets time
- VIEW** Sets up viewports
- WAIT** Holds up execution pending input from a machine input port
- WEND** Terminates WHILE loop
- WHILE** Creates loop that continues while the given condition is true



### ***Summary Of BASIC Statements (Continued)***

---

**WIDTH** Establishes length of output line

**WINDOW** Sets size of screen

**WRITE** Sends data to file or screen



### ***Summary Of BASIC Commands***

---

**AUTO** Produces line numbers automatically

**BLOAD** Places machine-language program in memory

**BSAVE** Stores machine-language program on specified device

**CLEAR** Sets any numeric variables to zero and any string variables to null

**CONT** Resumes program execution

**DELETE** Erases lines of a program

**EDIT** Shows line of program for editing

**FILES** Displays all or designated files on diskette

**KILL** Erases a designated file from a diskette

**LIST** Displays designated lines of a program

**LOAD** Places designated program in memory

**MERGE** Inserts the BASIC statements on a file into the current program

**NAME** Renames a file

**NEW** Clears memory for a new program

**RENUM** Assigns a new set of line numbers and line references to a program

**RESET** Empties the system buffer and shuts all files

**RUN** Initiates program execution

**SAVE** Stores current program as file

**SYSTEM** Terminates BASIC and initiates to DOS

**TRON [TROFF]** Initiates (or terminates) program trace option



## ***Summary Of BASIC Functions***

---

### ***Built-In Arithmetic Functions***

- ABS(N) Absolute (positive) value of N  
ATN(N) Arc tangent of N in radians  
CDBL(N) Changes N to double precision  
CINT(N) Changes N to integer  
COS(A) Cosine of angle A where the angle is given in radians  
CSNG(N) Changes N to single precision  
EXP(N) e to the N power  
FIX(N) Forms integer by truncating N  
INT(N) Yields the nearest integer less than or equal to N  
LOG(N) Natural log of N  
RND(N) Produces pseudorandom number  
SIN(A) Sine of angle A where the angle is given in radians  
SNG(N) Sign of N  
SQR(N) Square root of the positive number N  
TAN(A) Tangent of angle A where the angle is given in radians

### ***String And Character Functions***

- ASC(S\$) ASCII numeric code value for the initial character of S\$  
CHR\$(N) Yields ASCII character with numeric code N  
CVI(N\$) [CVS(N\$), CVD(N\$)] Changes N\$ to numeric value of the specified precision  
INSTR(p, S\$, T\$) Indicates the location of the first occurrence of T\$ in S\$ starting from character number p  
LEFT\$(S\$,x) Yields a string composed of the x left-most characters of S\$  
LEN(S\$) Number of characters in S\$





## ***Summary Of BASIC Functions (Continued)***

---

**MID\$(S\$,x,y)** Yields the y characters of S\$ beginning at character number x

**MKI\$(N) [MKSS\$(N), MKD\$(N)]** Makes a string of specific length for the number n

**RIGHT\$(S\$,x)** Yields a string of the x right-most characters of S\$

**SPACE\$(N)** Produces a string of N blanks

**STRING\$(x,v) [STRING\$(x,V\$)]** Generates a string containing the character with the ASCII numeric code v [or the first character V\$], x times

**VAL(N\$)** Numeric value of N\$

### ***Other Functions***

**EOF(N)** End of file N

**INKEY\$** Accepts a character from the keyboard

**POS(C)** Yields column position of cursor

**SPC(X)** Outputs X spaces in PRINT statement

**TAB(X)** Moves cursor to position X in PRINT statement

## **The LOGO Language**

LOGO is a very different language from BASIC. The distinction between the two is fundamental: LOGO is a *procedural* language. That is, the program is divided into small, independent pieces called *procedures*, which you can write separately and link together to form the entire program.

LOGO programs allow *recursion*, which, to oversimplify, means to repeat again and again. Actually, recursion is a computer science term that requires higher mathematics for a more accurate definition. The important thing is that, in LOGO, recursion is relatively

easy to use. For now, just come along for the ride, and we will give some clarifying examples of recursion later.

LOGO variables can contain any value—a number or a string. In formal terms LOGO does not use *data typing*, where the variable NUMBER can contain a numeric value but not a string, as in BASIC. Instead, LOGO is list oriented. In data processing a *list* is a loosely related group of items, similar to a shopping list. A list is much less structured than the array in BASIC or the file with its records and fields.

LOGO emphasizes graphics by using *turtle geometry*. You can make interesting drawings quickly and easily with LOGO. In the next section we'll discuss using the turtle, LOGO's guide in creating graphics.

We think you will like LOGO, as we do. The LOGO available for the PCjr is outstanding because it is user friendly and is supported by excellent documentation. Here are a few programs to illustrate the language and its power.

### Training The Turtle

The LOGO diskette comes ready for action, so insert it in the disk drive and boot. (Don't forget to make a backup copy of LOGO the first time you use it, so that you can use the copy and protect the original.) Enter the date and time when requested, just as you did for DOS. You will get a message saying "WELCOME TO LOGO," with a "?" on the next line. The question mark is the LOGO prompt character. Next to the "?" you should see the square LOGO cursor—the signal that you are ready to meet the LOGO turtle.

The turtle is an on-screen symbol that LOGO uses to guide you around the screen when you create pictures and other graphic displays. The turtle is always there, but it usually hides. To get the turtle to show itself, type:

#### SHOWTURTLE

(one word) or its abbreviation ST, then press [ENTER]. The "turtle" should appear as a triangle pointing straight up in the center of the monitor screen. The direction the turtle is heading is represented by the direction the triangle is pointing, in this case up.

Note that the LOGO prompt symbol and cursor are now rel-

egated to the lower left-hand corner of the screen, to allow the turtle maximum area to create its pictures. When you are finished using the LOGO turtle, you can make it hide again by typing HT (or HIDE TURTLE) and pressing [ENTER]. There are four commands (or, as they are called in LOGO, "primitives") that will change the turtle's heading or position. First, let's make the turtle go forward or backward a specified distance. The FORWARD command—abbreviated FD—instructs the turtle to move forward a designated distance. The distance must be given as a number, separated by a blank space from the FD command, such as:

**FD 40**

followed by [ENTER]. Try that simple command, and watch the turtle go. The BACK command (BK) is identical to the FD command, but it works in reverse.

**BK 40**

will return the turtle to its starting point. Notice that the heading of the turtle hasn't changed. Also notice the visible trail that the turtle leaves wherever it goes. This trail is what allows you to create pictures with LOGO.

In addition to moving forward and backward, the turtle can also change course, either to the RIGHT (RT) or the LEFT (LT). With the RT and LT commands, you must specify the number of degrees you want the turtle to turn. So,

**RT 90**

would cause the turtle to make a right-hand turn ninety degrees, but the turtle itself would not move. Note that either RT 180 or LT 180 will cause the turtle to turn completely around and begin heading in the opposite direction.

Using these four commands, explore the screen with the turtle. With a little trial and error, you should be able to determine how far it is from one end of the screen to another in turtle units, both horizontally and vertically. Experiment by moving the turtle off the edges of the screen, and see what happens (it reappears at the opposite side).

Now that your screen is cluttered with turtle tracks, you'll appreciate the next command. Type:

**CS**

(for CLEARSCREEN) to remove all the tracks and return the turtle to the center of the screen, where it will be heading straight up. Notice that CS does not erase the commands you have been typing in the lower left-hand corner.

The turtle is a talented artist, but if it isn't controlled carefully, things can get messy. LOGO provides some commands that enable you to shut off or erase the turtle's track-writing so that you can move the turtle around without leaving tracks. This ability will be necessary in the program you are about to write.

To understand these commands, imagine that the turtle has a pen in its mouth. When it moves, the pen drags along the ground and leaves tracks. You can tell the turtle to lift the pen off the ground with the PU (PENUP) command and then tell it to put the pen back down with the PD (you guessed it, PENDOWN) command. There are two other commands. PE (PENERASE) has the turtle erase any line it moves along, and PX (PENREVERSE) erases a line if the turtle walks along a previously made track. If the turtle moves over blank territory, however, it will leave its own track.

Experiment with the combination of the pen and movement commands (use the accompanying checklist as a guide). Then you can clear the screen with the CS command when you are ready to begin some serious drawing.



### *Some Important LOGO Primitives*

<i>Primitive</i>	<i>Abbreviation</i>	<i>Action</i>
BACK	BK	Moves turtle backward a designated distance
CLEARSCREEN	CS	Clears screen and places turtle in center of screen



## *Some Important LOGO Primitives (Continued)*

---

CLEARTEXT	CT	Erases text and places cursor in upper left corner of the screen
EDIT	ED	Changes to edit mode
ERASE	ER	Deletes a procedure
FORWARD	FD	Moves turtle forward a designated distance
HIDETURTLE	HT	Causes turtle to disappear, but it can still draw
IF		Decision statement
LEFT	LT	Rotates turtle left (counterclockwise) a designated number of degrees
LIST		Generates a list of objects
MIXEDSCREEN	MS	Permits both text and graphics on screen
PENDOWN	PD	Permits turtle to draw
PENERASE	PE	Permits turtle to erase the line it passes over
PENREVERSE	PX	Permits turtle to reverse the line it passes over
PENUP	PU	Stops turtle from drawing
PRINT	PR	Outputs to screen or printer
REPEAT		Executes a list of instructions a designated number of times
RIGHT	RT	Rotates turtle right (clockwise) a designated number of degrees
SHOWTURTLE	ST	Causes turtle to appear
TO		Defines a procedure

### **PCjr Draws PCjr**

Let's write a LOGO program to write the word "PCjr" on the screen using the turtle. As a demonstration we will first create the letter "P" as it will be drawn in the program.

If you have not done so, type:

**CS**

to clear the screen and return the turtle to the center. Now make a sharp left turn with the command:

**LT 90**

You must move the turtle to the left side of the screen but without leaving tracks, so type:

**PU**

(Remember, of course, that each entry must be followed by the [ENTER] key.) Now, with the turtle heading due west, type:

**FD 120**

to reach almost to the end of the screen on the left. Take a ninety-degree right turn and put the pen down by typing:

**RT 90**

followed by

**PD**

The turtle is ready to start drawing the "P".

Go forward ninety turtle units to form the spine of the "P". Turn right ninety degrees and go forward sixty units to form the top of the "P". Now turn right again ninety degrees, and draw the right side of the "P" by moving forward (down) forty-five units. Turn right again, and move forward sixty units to complete the letter. Congratulations!

The only drawback to what you have just accomplished is that it cannot be saved into a permanent file on a diskette, or even for later use by LOGO. To rectify this, we must give your program a name. In LOGO this is called defining a procedure. When you define a LOGO procedure, you tell the computer how "TO" something. After the "?" prompt symbol, type:

### TO PCJR

to tell LOGO that you will be teaching it to draw the word "PCjr." The prompt symbol now changes to the ">" symbol, which means you are defining a procedure.

Enter the program as it is written below. If you make a mistake in a line, use the [BACKSPACE] key to erase it and retype the command. If you made a mistake on a previous line, you will have to wait until you are finished and then use the EDIT command to make the necessary changes.

#### LOGO PROCEDURES

##### TRAINING THE TURTLE PROGRAM

```

TO PCJR
CS
LT 90
PU
FD 120
RT 90
PD
FD 90
RT 90
FD 60
RT 90
FD 45
RT 90
FD 60
LT 90
PU
FD 45
LT 90
FD 90
LT 90
PD
FD 90
RT 90
FD 60
RT 90
PU
FD 90
RT 90
PD
FD 60
RT 180
PU
FD 90
LT 90
PD
FD 45
PU
FD 10
PD
FD 10
RT 180
PU
FD 65

```

```

PD
FD 40
RT 90
FD 30
PU
BK 30
RT 90
FD 40
RT 90
FD 30
LT 90
PD
FD 45
BK 10
RT 45
FD 12
RT 45
FD 20
PU
END

```

Notice that when you are defining a procedure, the turtle does not carry out the commands as it did when you were typing them in directly. However, once the procedure is fully defined, all you need to do for the turtle to carry out a command is to type in the name of the procedure.

Every procedure must end with the statement `END`. After LOGO receives that statement, it will tell you that the procedure has now been defined. Once `"PCJR"` is defined, until you shut off the power, your computer will know how to `"PCJR"`.

To save this procedure onto a diskette, remove the LOGO diskette and replace it with a blank, formatted diskette. Type:

### **SAVE "PCJRPIC**

then press `[ENTER]`. LOGO filenames follow standard DOS conventions. However, they cannot have a filename extension, because LOGO automatically assigns them the extension `".LF"`

But wait a minute. You haven't seen the result of your labors. To see your creation, type:

### **PCJR**

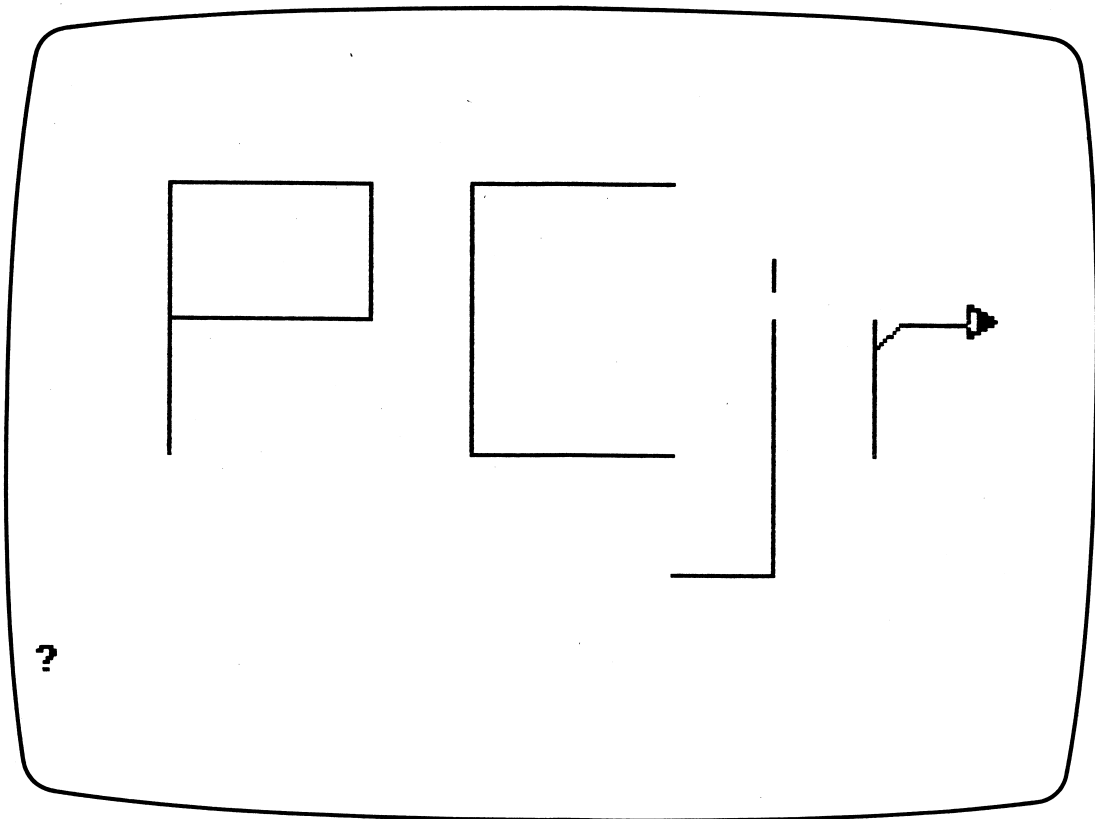
after the LOGO prompt (`"?"`) and press `[ENTER]`. If your screen doesn't look like Figure 11-1, you must have erred in entering the commands.

Mistakes are easily corrected with the LOGO editor. To get the editor, just type `EDIT`, followed by the name of your procedure, which must begin with a quotation mark. So, typing:

### **EDIT "PCJR**

followed, naturally, by the `[ENTER]` key will get you into the editing





*Figure 11-1* Using LOGO to draw PCjr.

mode. From there simply use the cursor control arrows, the [ENTER] key, and the [DEL] key to correct any errors. When you have made all the necessary corrections, press [ESC] to return to the standard LOGO screen with the turtle. Now type:

**PCJR**

again to execute the corrected version.

Be aware of the fact that the diskette version of the procedure has not been changed by the edit session. To save the edited version, you must repeat the SAVE command. However, LOGO will not let you save the same file more than once. Therefore you must either give the procedure a new name—such as PCJRPIC2—or erase the

first version with the ERASEFILE command. In this case you would type:

```
ERASEFILE "PCJRPIC
```

Once you have done this, you can save PCJRPIC again, this time with the corrections.

### **Recursive Programs**

LOGO is particularly well suited to an unusual type of programming called recursive programming. *Recursive* simply means "repeating," and a recursive program is one in which the same basic function is repeated over and over—changing only minutely—to form a totally new result. Sound complicated? Actually, the sample recursive program you will be running is far simpler than the exercise you just completed.

Start by defining the procedure TRI, a simple program that produces an equilateral triangle (all three sides of equal length) 100 turtle units long. The procedure goes like this:

```
TO TRI  
PU  
BK 30  
LT 30  
PD  
FD 100  
RT 120  
FD 100  
RT 120  
FD 100  
RT 150  
PU  
FD 30  
END
```

If you make any mistakes, use EDIT "TRI to correct them. Otherwise, save the file under the name TRI1. Now type:

```
TRI
```

and press [ENTER] to see what you have created.

LOGO has some powerful features. First, you can use a procedure inside another procedure. Second, you can repeat that inner procedure multiple times, varying the execution ever so slightly each time. This combination, called the REPEAT command, proves valuable, as you will see.

The REPEAT command needs two inputs. The first input on the line, separated from the word REPEAT by a space, is the number of times that the command is to be repeated. The second input, entered in brackets, is the command to be performed.

In this sample recursive program we will repeat the TRI procedure thirty-six times, but each time we will start the turtle off ten degrees farther to the right than the previous time. Type:

```
TO RECURSV
CS
REPEAT 36 [RT 10 TRI]
END
```

Your first recursive program is just that simple. Save this under the name RECURSV1. Now, type:

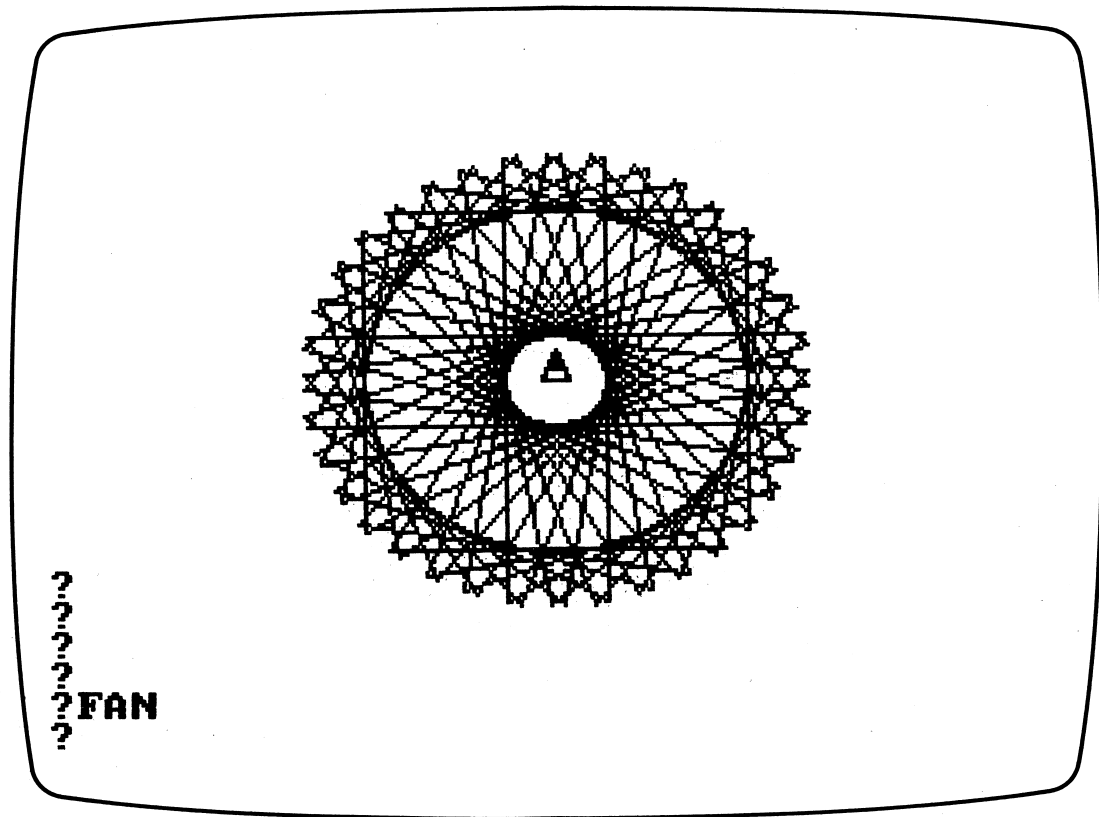
```
RECURSV
```

and watch the fun LOGO has. You should have created a beautiful design like the one in Figure 11-2. Alter the number of iterations (repetitions) and the number of degrees of offset in the REPEAT statement by using the EDIT "RECURSV command. See what you can create. What was so special about the combination of thirty-six repetitions and ten degrees of offset per repetition? (Hint: How many degrees are in a circle?)

## LOGO—Not Just Another Pretty Picture

The graphics and recursive capabilities of LOGO are among the most spectacular of any programming language. But LOGO can do a lot more than just draw pictures. Let's put it to work as a teacher.

We'll write a program that is really a math learning game. It was designed to make learning division with decimals fun. In this program we will take advantage of some of LOGO's text manipulation features to custom design a math quiz that keeps asking new and different questions.



*Figure 11-2* Creating a design with LOGO.

Here are two sample questions from this quiz game:

JANE HAS A PET TURTLE  
IT EATS 3 POUNDS OF FOOD EVERY 2 DAYS  
HOW MUCH FOOD DOES IT EAT PER DAY?  
2  
SORRY, THE CORRECT ANSWER IS 1.5  
FRED HAS A PET DOG  
IT EATS 3 POUNDS EVERY 4 DAYS  
HOW MUCH FOOD DOES IT EAT PER DAY?  
.75  
CORRECT

Each time, LOGO will substitute a different name, animal, and mathematical problem. Does this look like an impossibly difficult program to write? Not in LOGO. It is quite simple and consists of just two procedures—one to pick the names and the pets and the other to ask the questions.

This program will rely heavily on an important LOGO function, the `RANDOM n` statement. This statement requires a number (`n`) to determine how many possible random numbers the `RANDOM` statement can come up with. `RANDOM` will select a random number from 0 to 1 less than `n`. So, for `RANDOM 2` the number will be either a 0 or 1. You can write a coin-flip program using `RANDOM 2` by designating 0 "heads" and 1 "tails." `RANDOM 10` will select a number from 0 to 9, `RANDOM 100` a number from 0 to 99, and so on for every possible `n`.

The random numbers generated in this program will have two distinct uses. One is to supply the numbers for the math question. The other is to select the problem element from a list of people and pets.

This program also will use the LOGO `PRINT` statement, abbreviated `PR`. LOGO prints sentences when the abbreviation `SE` is used in the `PRINT` statement. Both variables and literal words can be used in the `PRINT` statement, as you will see.

LOGO variables are identified with the `MAKE` statement. There are several `MAKE` statements in this program, including the following two examples:

```
MAKE "PET [CAT DOG FISH TURTLE]
MAKE "LB ( RANDOM 6 ) + 1
```

The first `MAKE` statement translates as: Make the variable `PET` one of the choices `CAT`, `DOG`, `FISH`, or `TURTLE`. The second one is: Make the variable `LB` one more than a random number between 0 and 5 (in essence, a random number between 1 and 6). The quotation mark in front of the variable indicates that this is the name of the variable. If the value of the variable is needed, the name is preceded by a colon instead, as in the `PRINT` statement:

```
PR ( SE [IT EATS] :LB [POUNDS OF FOOD EVERY] :DYS "DAYS )
```

Here, the value of the variables `LB` and `DYS` is inserted into the

sentence `IT EATS POUNDS OF FOOD EVERY DAYS`, which then is printed. Note that before you can use a variable's *value* in a statement, you must first define the variable with a `MAKE` statement.

The words that are to be printed as part of the `PRINT` statement must either be set off by brackets or a quotation mark. In the `PRINT` statement above, `IT EATS` and `POUNDS OF FOOD EVERY` are enclosed in brackets. Brackets usually are used to enclose multiple words, although they can be used to enclose only one word. The quotation mark, however, can designate only a single word. As soon as a space is encountered, LOGO will assume that the space signals the beginning of a new variable or command.

Now for the actual programming. The core of the program involves two procedures. The first we will call the `PICK` procedure; it will pick the names for the first part of the question. `PICK` requires that you define a variable, called `LIST`, that contains the names from which the choices will be made. `PICK` then `COUNTs` the number of elements in the list (`COUNT` is a predefined LOGO function) and randomly selects a number from 1 to the number of elements in the list (in this case 4). `PICK` then chooses the `ITEM` (also a predefined function) from our list. The choice is dictated by the random number (the third item if the random number is 3, the first item if it is 1, and so on). `PICK` then `OUTPUTs` the `ITEM` to the main program. Simple, right? Try it. Enter the `PICK` procedure:

```
TO PICK :LIST
  OUTPUT ITEM (1 + RANDOM COUNT :LIST) :LIST
END
```

As a check, let's run a sample list and see what happens. Type:

```
MAKE "TEST [ONE TWO THREE FOUR]
PR (PICK :TEST)
```

You already have seen most of the coding for the rest of the program. Call the main procedure `QUIZ`.

```
TO QUIZ
  MAKE "NAME [WALT FRED JANE MARY]
  MAKE "PETS [CAT DOG FISH TURTLE]
```

```

PR ( SE PICK :NAME [HAS A PET] PICK :PETS)
MAKE "LB ( RANDOM 6 ) + 1
MAKE "DYS ( RANDOM 3 ) + 1
PR ( SE [IT EATS] :LB [POUNDS OF FOOD EVERY] :DYS "DAYS )
PR SE [HOW MUCH FOOD DOES IT EAT PER DAY?]
MAKE "ANSWER :LB / :DYS
TEST :ANSWER = READWORD
IFTRUE [PR "CORRECT]
IFFALSE [PR SE [SORRY, THE CORRECT ANSWER IS] :ANSWER]
QUIZ
END

```

The last part of the quiz program inputs the student's answer and tests its accuracy. To do this, LOGO must already know the answer. So the variable ANSWER is defined as LB divided by DYS. The READWORD operation tells LOGO to accept a single word from the keyboard as direct input (similar to the BASIC INPUT statement). The "word" being inputted in this case is actually a number, but this makes no difference to LOGO. The program TESTs the inputted number to see if it is the same as the answer it has already calculated. Every TEST command has an IFTRUE statement and an IFFALSE statement associated with it. In this case, if the answer given by the student is correct, the TEST is TRUE, and IFTRUE prints "CORRECT." If the answer is not an exact match (watch what happens with numbers like 4/3), then TEST is FALSE, and IFFALSE prints the correct answer.

The last line of the program before the END statement is QUIZ, the name of the procedure. This sets up an endless loop, repeating the QUIZ questions forever. Well, almost forever. You can stop the quiz by turning off the computer (the drastic solution) or by pressing the combination of the [CTRL] and [BREAK] keys. This key combination interrupts any LOGO procedure and returns you to the LOGO prompt symbol "?".

If necessary, EDIT your program with the EDIT "QUIZ or EDIT "PICK commands. When the program is properly entered, save it under the name QUIZ1. Then, to see your quiz program in action, type:

**QUIZ**

Voilà! You have written your first educational program.

## Pascal Revisited

The Pascal language was developed in 1971 in response to a controversy in computer science about—of all things—the GOTO statement. The GOTO statement was considered a problem because it allowed programmers too much freedom. They could transfer into any part of a program, with the result that trying to follow some of their poorly constructed programs was like trying to unravel a plate of spaghetti. Pascal was designed to teach programming without allowing the student to use the troublesome GOTO statement and end up in a tangled mess.

The designer, Niklaus Wirth, had other requirements for the language as well. He wanted Pascal to have more flexible data types than were available with the popular languages of the day, like BASIC, COBOL, or FORTRAN. He also wanted to be sure Pascal was efficient for use on small computers so that it could be used to teach programming classes without tying up a large computer.

Wirth accomplished these goals by making Pascal "strongly structured." In fact, before you can even begin to write a line of instructions, you must work out all the details of your program, including the name and type of each variable. Professional programmers are expert enough to lay out their plans at this level of detail, but novices often find it difficult to anticipate all the program details in advance.

Unfortunately, getting rid of the GOTO proved more difficult than anticipated, so Pascal does include a GOTO statement. However, Pascal is a procedural language, like LOGO, which means that the need to use the GOTO is minimal.

The University of California at San Diego subsequently developed an excellent Pascal package for microcomputers, and the language became popular for introductory computer science programming courses. Business schools, however, continued to use BASIC as their primary teaching language.

Probably the most important contribution Pascal has made in data processing is to increase our awareness of the need to preplan programs. The structured approach does not require a strongly structured language. Consequently, a number of the best features of Pascal already have found their way into BASIC and other languages. As we said earlier, expect a Pascal version suitable for the PCjr soon.



## Further Reading

### BASIC

Your IBM BASIC manuals are among the best in the industry. If you have any questions about the action or format of a statement, they are your best source. Not only do they include the formal definitions of the BASIC language components, but many of the statements are illustrated by examples. Some of the IBM BASIC manuals are: *BASIC Made Easy for the IBM PCjr*, *Hands-On BASIC for the IBM PCjr*, *IBM PCjr BASIC Reference Manual*, and *BASIC Reference Manual*. The IBM Personal Education Series offers an additional resource: *Learning to Program in BASIC*.

A great number of programming books have the word BASIC in the title, but most of them don't give you a good grasp of the powerful features available with your PCjr BASIC. PCjr BASIC, as we said, is one of the best around. One BASIC book worth reading is *Programming the IBM Personal Computer: BASIC*, by Neill Graham (Holt, Rinehart & Winston, 1982). Another book, with an emphasis on business applications, is *Using BASIC on the Cyber*, by Norman Sondak and Richard Hatch (Science Research Associates, 1982). The BASIC is similar to the PCjr version, and the book offers many practical examples and worked programs. *BASIC Computer Programs for the Home*, by Charles Sternberg (Hayden, 1980) suggests a number of applications for the home. Its BASIC is not the PCjr version, but the ideas are still useful.

If you want a detailed review of all the BASIC statements, try *Handbook of BASIC for the IBM PC*, by David Schnieder (Brady, 1983).

### LOGO

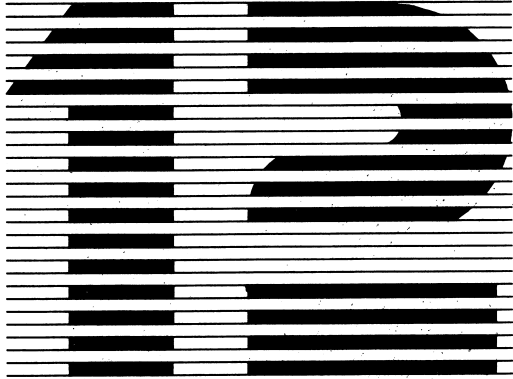
Your major reference on LOGO is *LOGO Programming with Turtle Graphics* (IBM Corporation). The documentation is typical IBM—very good—and there is an excellent user's guide. The book's only major shortcoming is that it doesn't do justice to the enormous power of LOGO.

To learn more about LOGO and its philosophy, read *Mindstorms: Children, Computers and Powerful Ideas*, by Seymour Papert (Basic Books, 1982). On the topic of recursion, you might want to see Douglas R. Hofstadter's superb book, *Godel, Escher, Bach: An Eternal Golden Braid* (Random House, 1980).

Several books on LOGO for the Apple computer generally are applicable to the PCjr version of LOGO. Two of the better ones are: *LOGO For the Apple II*, by H. Abelson (Byte Books, 1982) and *Discovering Apple LOGO*, by David Thornburg (Addison-Wesley, 1983).

### **Pascal**

If you really become interested in Pascal, Peter Brown's *Pascal from BASIC* is a fine transition book (Addison-Wesley, 1982). The basic reference on Pascal is *Pascal: User Manual and Report, 2d ed.*, by K. Jensen and N. Wirth (Springer-Verlag, 1978), but this is not easy to read and we cannot recommend it for beginners.



# Telecommunications— The PCjr's Window On The World

Ten or fifteen years ago if you had predicted that a computer as powerful and sophisticated as the IBM PCjr would be widely available and affordable, you would have been a laughingstock. But now home computers are firmly entrenched in hundreds of thousands of households, and we are taking for granted capabilities that were considered impossible to achieve (let alone mass-produce) just a short while ago. The computer age has come a long way in a brief time, but we believe that the near future holds even more astounding possibilities for owners of home computers like the PCjr—and many other experts agree. Some of these forthcoming innovations seem as farfetched to today's computer users as the blue-sky forecasts of the dreamers of a decade ago.

## The Future Is Now

Right now, in the city of Los Angeles, you can pick up your touch-tone telephone, check your bank account balance (and if necessary, move some money from your savings to your checking account), then hang up and call a local supermarket and do your grocery shopping for the week—all without leaving your favorite chair. In less than five years (a *very* conservative estimate) even this scenario will be obsolete—as extinct as the dinosaur.

How, you ask? By the year 1989, you will simply use your PCjr and its internal modem to execute your SHOPPING program, key in a few special orders at your keyboard, and your groceries will be

selected, packaged, and paid for without your ever lifting a telephone receiver or speaking to another human being. Your trusty computer will handle everything—aided by the myriad of computers on the other end of these transactions.

When you finish “shopping,” you will ask your PCjr to pay your credit card charges, your monthly gas, electricity, and telephone bills, and to compare the payments to your preprogrammed budget estimates for the year to see if you are staying within your spending limits. Next, you will use your computer to check on the status of your entire stock portfolio, and perhaps you will sell a few shares of one stock and buy some others.

Finally, exhausted by the sheer strain of all this, you will call up your TVGUIDE program for a listing of the day's fare—a program preset to notify you when your favorite sit-com is on and whether your hometown football team is playing, so you won't miss anything.

If this sounds farfetched, take heart. So did all the other predictions that came true to bring computer power not just into *your* hands but into your children's! To us, the only thing farfetched about this prediction is the time scale. We will be quite surprised if all these fantasies take that long to become a reality.

You see, the essential mechanical elements of the picture we have just painted already exist and are available for your PCjr right now. In fact, a good deal of the informational elements exist too. Bank accounts are totally computerized and readily accessed by telephone or by card-activated teller machines. Stock market quotations can be accessed by computer/telephone linkage. And, as we said, grocery shopping by phone is now more than a reality—it's a growing industry. The only hypothetical part of this prediction is the availability of a home computer market large enough to support this and make it a paying proposition. However, with home computers selling like hotcakes (IBM alone has sold hundreds of thousands), the market forces are present. It's just a matter of time before the prediction becomes a day-to-day fact.

*Telecommunications* have been conducted via personal computers for over a decade. The electronic device that makes telecommunications possible is the *modem*. Initially, modems were placed outside the system unit. These external modems were connected to the computer by a serial interface and to the telephone by an acoustic coupler. The acoustic coupler modem converts sound from

the telephone instrument to electronic pulses for the computer and vice versa.

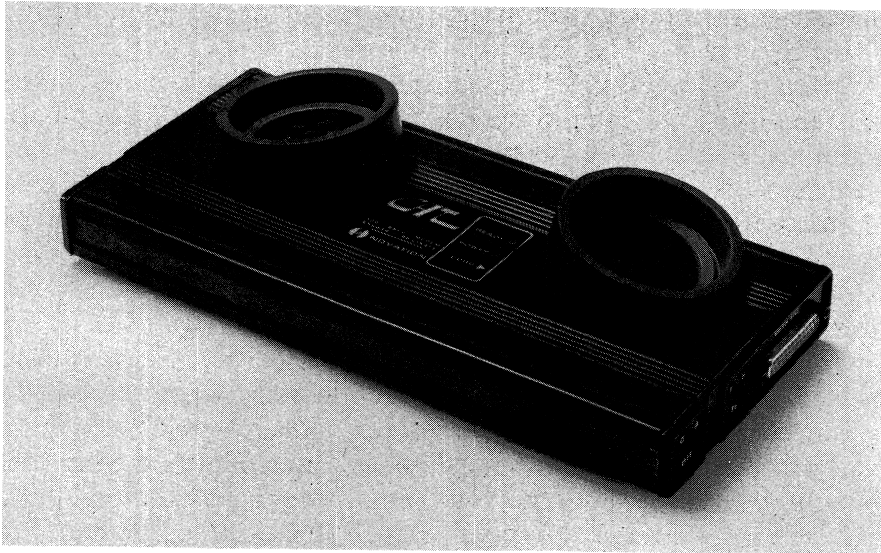
Today, two types of modems are available: external and internal. The *internal modem* represents the highest level of sophistication ever achieved for connecting a computer to the unlimited wealth of resources available by telephone. Unlike previous generations of modems, the internal modem requires no telephone instrument. The modular cord that normally plugs into your telephone is connected directly to the back of your PCjr, and the computer does the rest—including the dialing.

You already have a serial interface as part of the hardware package for your PCjr, and you can buy a low-cost *external modem* to connect with it. These modern external modems no longer require acoustic couplers, but they're still not as convenient as the internal version, because they tie up your serial interface (which you may have already committed to your printer). But external modems are less than two-thirds the price of their internal counterparts—quite a savings, especially if communications is not a big factor in your processing. And they are as reliable as internal modems. You should consider using an external modem if you're not planning to do extensive communicating or if you have a parallel printer (so the serial port is available for the modem).

We have used both internal and external modems on our personal computers. Once the system is set up, there is no functional difference between the two. A number of low-cost external modems are available, such as those manufactured by USI Corporation and Novation. In fact, we have used a Novation modem for over ten years without a single problem or repair. By the way, the IBM internal modem for the PCjr is manufactured by Novation.

To support this powerful piece of hardware, IBM has provided a dazzling collection of communications software. This array of software will make it as easy for you to use the PCjr for intercomputer communications as it is for the PCjr to handle the many communications tasks you design for it. Once you master the communications programs required to perform these applications, the world will truly be at your fingertips. And when all the forthcoming developments we described are operational, you will be ready, willing, and able to take full advantage of every one.

The communications capability of the PCjr is the most excit-



An external modem. The telephone handset is placed on top, and information is transmitted and received over the telephone lines.

ing and potentially rewarding aspect of your new home computer. This chapter is your introduction to this vast potential. It will prepare you to take full advantage of this vital resource for your own expanding needs. Welcome to the future of home computing—right here and now.

## **Telecommunications Fundamentals**

Before you launch into the twenty-first century, there are some basic facts and specialized terms you should know about telecommunications. Moving data from place to place is nothing new for the computer. Internally, the computer transfers data from the disk drives to the central processing unit and from memory to the printer or the display monitor all the time. However, moving data from one computer to another presents a different set of problems.

Three components are needed to move data outside of the system and across telephone wires to another machine. In order to accomplish this feat of engineering, you must have a modem, data communications software, and the necessary communications

linkages to join the two computers. Right now, you should have two of these components (the internal modem and the telephone system). Later in the chapter we will discuss the third element (the data communications software), because there are several options open to you in this area.

Once you have an open channel for communications, you can either talk to people, communicate with other computers, or interrogate commercial information services or public access data bases. The latter are popping up in tremendous numbers. We will explore these new data bases in a separate section. For now, however, let's concentrate on the technical information that will save you a good deal of time and trouble when you begin reading the directions *not* so clearly printed on the labels of most of these new systems.

The following material on communications parameters is not easy to grasp, but understanding it is essential if you want to be self-sufficient in handling data communications. Otherwise, you will have to depend on the local computer guru to bail you out each time you want to establish communications with a new location. But take heart, with a little concentration and fortitude, you will get the hang of it. Thousands of other novices already are tuned into the world of data communications. Read this portion carefully—more than once, if necessary—and you soon will be the communications guru in your neighborhood.

As you already know, all the data inside the computer is processed, stored, and transferred as binary digits or bits. Within the digital computer these bits are represented as voltage pulses, either "on" (1) or "off" (0). The job of your communications package is to move these bits reliably (without losing or distorting any data) from your computer to the remote location, and vice versa.

Within the computer, data can be represented in various ways, but data sent across telephone wires normally is encoded in the American Standard Code for Information Interchange, or ASCII (pronounced as-key) for short. To make matters a little more complicated, there are two versions of the ASCII code in use: the standard seven-bit code, which can display 128 characters, and an extended eight-bit version used in your PCjr, which can display 256 characters. Each character is transmitted serially, one bit at a time, across the telephone line.

While data is in transit across the wires, it undergoes a considerable transformation, although this process is invisible to you.

The bits within your computer must be converted to electronic signals that represent a range of tones, because, as you well know, the telephone system was designed to transmit sounds, not digital data. When these tones reach their destination, the process must be reversed. The tones have to be converted back to the digital pulses used by the computer. The job of performing these transformations is relegated to the modem. The word *modem* is actually a contraction of two words—*modulate* and *demodulate*—because the modem handles both the conversion (or modulation) of bits on the originating end and the translation (or demodulation) of tones back to bits at the receiving end. Because the modem is really doing two things at once, it follows certain conventions (or *protocols* as they are known in data communications) to avoid muddling the messages.

The modem that originates the call uses the frequency 2,225 Hz (hertz, or cycles per second) for a 1 and 2,025 Hz for a 0, whereas the modem that answers the call uses the frequency 1,270 Hz for a 1 and 1,070 Hz for a 0. These protocols permit the two modems to talk across the line without getting mixed up. Figure 12-1 illustrates this concept.

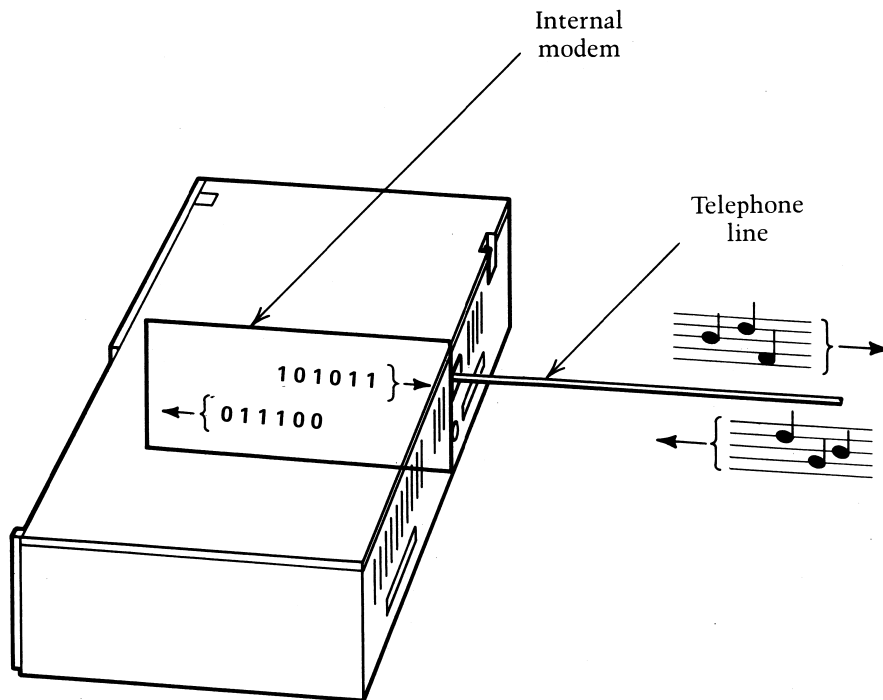
How can the modem at the receiving end be sure where one character ends and the next begins, since each character is sent out serially? The answer is a technique called *asynchronous transmission*.

Asynchronous transmission uses a zero bit, called a *start bit*, at the beginning of each character and a one bit, called the *stop bit*, at the end of the character. With these start and stop signals to *frame* the data bits that compose each character, the receiving modem knows exactly when a new character begins and ends. Some communications systems even require two stop bits instead of one, for extra security.

Did you ever play the parlor game Telephone? Then you know how garbled a message can get from the time the first person receives it to the time the last person in line has to deliver it. But you don't need a game to convince you that the human communications link is unreliable. Your personal experiences must have included numerous "mix-ups" because of communications breakdowns.

Your telecommunications equipment is a lot less error-prone than people, but the possibility of a minor miscommunication still





**Figure 12-1** The internal modem changes the voltage pulses to tones for transmission over the telephone network.

exists, even in the most sophisticated systems. However, the designers of these modern communications systems have developed a technique to detect the most common errors. The method is simple. It involves adding an extra bit to every data item as a check bit. The extra bit, known as a *parity bit*, is selected on the basis of the number of one bits in the data.

This error-detection system can use either odd or even parity. With odd parity the total number of one bits (including the parity bit) must add up to an odd number. For example, to send the character S with the seven-bit ASCII character code 1010011, you would add another one bit to generate odd parity or,

<i>Data</i>	<i>Parity</i>
1010011	1
S	odd

Notice that the total number of one bits in this example is four data bits plus one for the parity bit, or five bits in all—an odd number.

If a miscommunication changes the S to an R (which has the ASCII code 1010010), our parity scheme would detect the error right off the bat, because the total number of one bits would be even instead of odd.

#### ERROR IN TRANSMISSION CHANGES S 1010011 TO R 1010010

<i>Transmitted</i>		<i>Received</i>	
<i>Data</i>	<i>Parity</i>	<i>Data</i>	<i>Parity</i>
1010011	1	1010010	1
S	odd	R	FAILS PARITY CHECK— EVEN BIT TOTAL

The even parity system works the same way, except that the parity bit would make that total number of one bits an even number.

While either odd or even parity easily can detect single-bit errors, two-bit errors can still go undetected. Fortunately, even single-bit errors are such a rarity in modern data communications networks that many communications links do not require the use of parity at all. And double errors are so unlikely that they usually are not worth the expense and effort to detect.

The transmission rate of a communications line is measured in terms of *bauds*, the number of signal events per second. In data communications the signal event would be the transmission of a bit. Hence the baud rate is the number of bits sent over the communications line in one second. The standard asynchronous transmission rates over a telephone line are 300 and 1200 baud.

If you're operating at 300 baud, with a seven-bit ASCII character code, and transmitting ten bits per character, you can expect to send or receive about 300 words a minute. This rate produces the equivalent of one double-spaced page of type in approximately three-quarters of a minute. A 1200-baud line operates four times as fast.

When the modem is both sending and receiving data at the same time, the mode of operation is known as *full duplex* (FDX).

Full duplex is the fastest and most common mode of operation for data communications systems. It offers another advantage besides speed. The receiving computer can return or *echo* back the transmitted data to the sending computer, which then displays each character being sent. This gives the sender an opportunity to make a visual check of the accuracy of the transmission. At one time the FDX mode was even called the echo-plex mode of operation.

As long as you realize that the characters you see displayed on the screen are reflected back from the receiving computer, you can use that data to verify that all is well with your transmission. But, should you type:

**Deposit value \$220.00**

and see

Deposit value \$2260.00

you can be sure that something has gone haywire in the transmission. If you see even one small error, stop everything. Check all your transmitted data in detail. With a communications error similar to the one just depicted, you can be sure you are seeing only the tip of the iceberg. Such an error is one of the major reasons computers occasionally issue those outlandish checks for \$1,000,000,000.00 dollars when the amount was supposed to be \$1,000. Such things can happen when the user or operator fails to detect a data transmission error.

The other choice of operation modes is *half duplex* (HDX). With HDX operation, no echo is transmitted. Each character is displayed directly on the screen as it is typed. In this case what you see is what you type, not what is being transmitted. You have no way of knowing what is received at the other end.

Occasionally, a discrepancy occurs between the two modems: The originating modem is set to HDX and the receiving modem is set to FDX operation. This kind of mistake is easy to recognize because it causes each character you type to appear in duplicate, thus:

```
TTHHISS IISS WWHHAATT AA HHAALLFF DDUUPPLLEEXX,,
FFUULLLL DDUUPPLLEEXX MMIISSMMAATTCCHH LLOOOKKSS
LLIIKKEE...
```

What is happening is that the transmitted data is being displayed directly on the screen in one mode and echoed back in another. This problem has a simple solution—just change the setting on your modem to full duplex.

By the way, if the modems are reversed, with full duplex on the originating end and half duplex on the answering side, you will see nothing at all on the screen. Again, changing the setting (this time to half duplex) will correct the problem.

## Setting Up Your Equipment—The First Step

To use telecommunications, you will need to set up your equipment so that your PCjr can “talk to” the other computer or data base. If the two systems don’t “speak” the same language, there can be no communications. You should know the proper values for these settings before you start a communications session. However, if you don’t, try the settings given in the checklist as a starting point.



### Telecommunications Settings Checklist

1. ASCII code: Seven or eight bits (seven bits is typical)
2. Start and stop bits: Not mandatory, but they are normally used
3. Parity: Odd, even, or none. (If in doubt, try none as a first guess)
4. Full or half duplex: Use Full Duplex. If it is incorrect, you'll know it soon enough
5. Baud rate: 300 is usual

## Communications Software—Your Window On The World

We have examined two of the essential ingredients in your communications system—the modem and the telephone system that

links your computer to the world—but the key to making it work is, as usual, in the software. Communications, like every other computer application, could never be accomplished without programmed instructions from software packages designed specifically for the task of communicating—with other systems, information services, bulletin boards, and data bases.

Many different communications software packages are available, but we'll be concentrating on two that are available for use with the family of IBM personal computers (including your PCjr): Personal Communications Manager and CROSSTALK. Both of these communications packages are outstanding examples of personal computer communications software, and both contain the features necessary to make your computer a window on the world.

What exactly can you do with your communications capability? For one thing, you can use CROSSTALK and the Personal Communications Manager to design an electronic mail system. For another, you can use these powerful packages to convert your PCjr into an "intelligent terminal."

You know by now that computers can do nothing by themselves—except follow instructions—so how can a computer or a computer terminal be intelligent? Perhaps "intelligent" is a misleading description of these devices, but compared with ordinary terminals, they are extremely capable. Intelligent terminals have computational and storage capabilities of their own, whereas their "dumb" counterparts (with no central processors) are dependent on a remote computer to control their operation. Dumb terminals can let you keyboard data into a remote computer and receive data back, but that is all these terminals can do. Every other action must be controlled by the computer. For the most part this arrangement is fine, especially if you are only making simple data entries that require an immediate response.

Such terminals are just the ticket for that airline reservations clerk who only has to book your ticket or check your seat assignment. But for composing a letter, writing a program, or doing anything that requires editing, review, and study—or even simple calculations—dumb terminals can be a drag. You want a terminal with some storage and arithmetic capabilities of its own so that you can do the preliminary editing and program testing before you connect with the remote facility. You soon will see how significant the difference between an intelligent and a dumb terminal can be in

data communications.

With an intelligent terminal you will be able to hook up with a remote computer and tap other available information services or data bases—something you'll be tuning in to more and more with the explosion of resources. You also will be able to use your computer as an electronic mailbox to send, receive, and store messages. You even will have the option of creating your own personal computer center—accessible from anywhere in the world (via the telephone)—thanks to your intelligent terminal.

Dumb terminals can perform these tasks for you too, but at a higher price. You will have to pay the expensive connect-time charges while you labor through the typing and editing functions necessary to create a finished product, instead of taking care of this work on your own computer (and your own time) before you make the connection. By using the intelligent terminal capability, you can do the preliminary work in advance and save literally hundreds of dollars that would go down the drain if you bounced back and forth between a dumb terminal and the expensive remote computer. We've known several experienced computer users who discovered this after they had spent a lot of money on an information service.

The intelligent terminal has other advantages as well. If you're making an inquiry into a data base on one of the information services, you can have the file "down-line loaded." With this option the entire file would be sent to your computer at high data transfer speed (not at your typical reading speed) so that you could study it in detail and even extract portions of it for your own reports at your own convenience—and without those costly connect charges.

The saving in money is a big bonus, but if you have limited access time on the remote computer you're using—a typical problem for high school or college students doing computer assignments—an intelligent terminal can offer another important advantage. It allows you to "up-line load." That is, you can do all your typing, editing, and preliminary testing on the local system (your home computer) before you connect with the remote computer. Then you can send your completed work to the remote computer at high speed for processing.

If you make efficient use of the down-line and up-line loading capabilities available with your personal computer, you will save money and get a lot more mileage out of your data communications system.

Electronic mail, an exciting use for your intelligent terminal, allows you to transmit letters and messages to and from your computer via the telephone network. The extra advantage inherent in this computerized system is that your mail can be stored, cataloged, and filed—indefinitely—in a compact electronic form. And, of course, you can extract portions of the letters and documents, just as you can with word processing systems, without having to retype a single word.

Now, let's take a closer look at the "brains" behind these communications marvels—the software packages.

### **Personal Communications Manager**

The Personal Communications Manager (PCM) is a state-of-the-art example of a communications software package designed to make your PCjr perform as an intelligent terminal and to manage an electronic mail system—two telecommunications tasks that are essential for personal computers.

At first glance, PCM can seem quite formidable—for two reasons. First of all, PCM is a large and comprehensive package that supports the entire IBM PC line as well as a number of different communications options. All these variations on a basic communications theme can be confusing even to an experienced data processing professional, let alone a novice. The second reason is that the documentation is a far cry from the usual IBM standards. It is poorly organized and more confusing than enlightening in some places. Its approach smacks of the old cliché: Just get off one stop before I do. Obviously, this kind of "help" is not very useful, and you'll find yourself going around in circles quite a bit before you catch on.

Fortunately, if you are using an internal IBM modem (as you probably are), then you'll avoid dealing with some of the most difficult parts of the PCM package. However, once PCM is *configured* (tailored for use on your particular system) and the proper communications settings have been established, PCM is simple to use. Poor directions for its setup are the biggest barrier to your communications with PCM.

To assist you in handling some of the more baffling aspects of the setup and configuration process, we have developed a straightforward and easy-to-follow PCM Checklist. Because it would be

impossible to take you through the details of the process (we would have to rewrite the manual to do that), you should be prepared to consult the PCM documentation for the nitty-gritties. Our checklist will give you a perspective on the process, if you use it in conjunction with the manual.



### ***Personal Communications Manager Checklist***

---

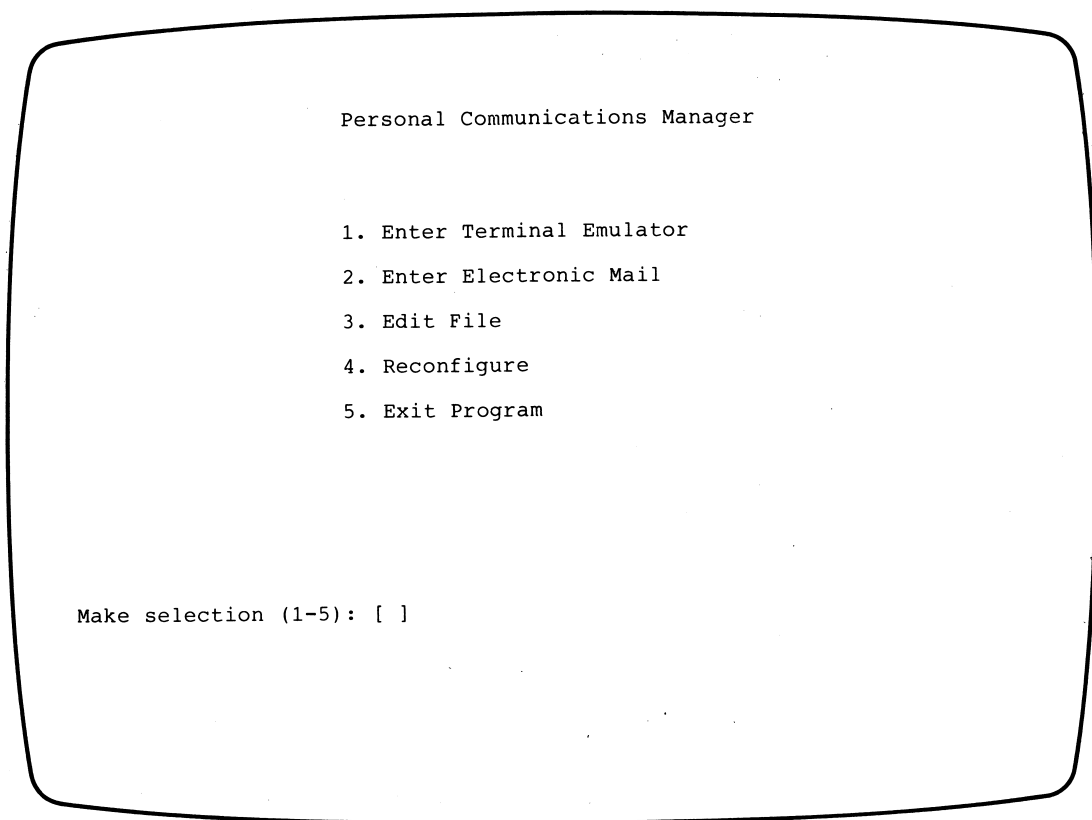
This checklist focuses on the hows and whats of putting the Personal Communications Manager (PCM) software package to work for you without getting bogged down in extraneous detail. To use this checklist properly, you need *five* things: an installed internal IBM modem, the PCM manual, and the number, communications settings, and log-on procedures for an information service. See the section on public access systems for possible free access numbers you can use to test your communications setup.

1. Begin your reading of the PCM manual with Appendix B, "Telecommunications Overview." This appendix explains some of the communications terminology used in the manual. Then read the manual from the beginning. Don't spend time with the technical jargon at the front. Generally, everything works the way you would expect it to from your experience with DOS.
2. To use PCM you first must set up and then configure the program. Doing this is easier than it appears from the manual. Have a blank diskette ready and use:

#### **SETUP 3 80**

as the setup parameter. If the computer seems to be taking forever to finish, this doesn't mean you did anything wrong. The setup routine is slow, so be patient. Be sure to label the diskette: PCM Work Diskette. From now on use the Work Diskette and put the original PCM program diskette in a safe place.

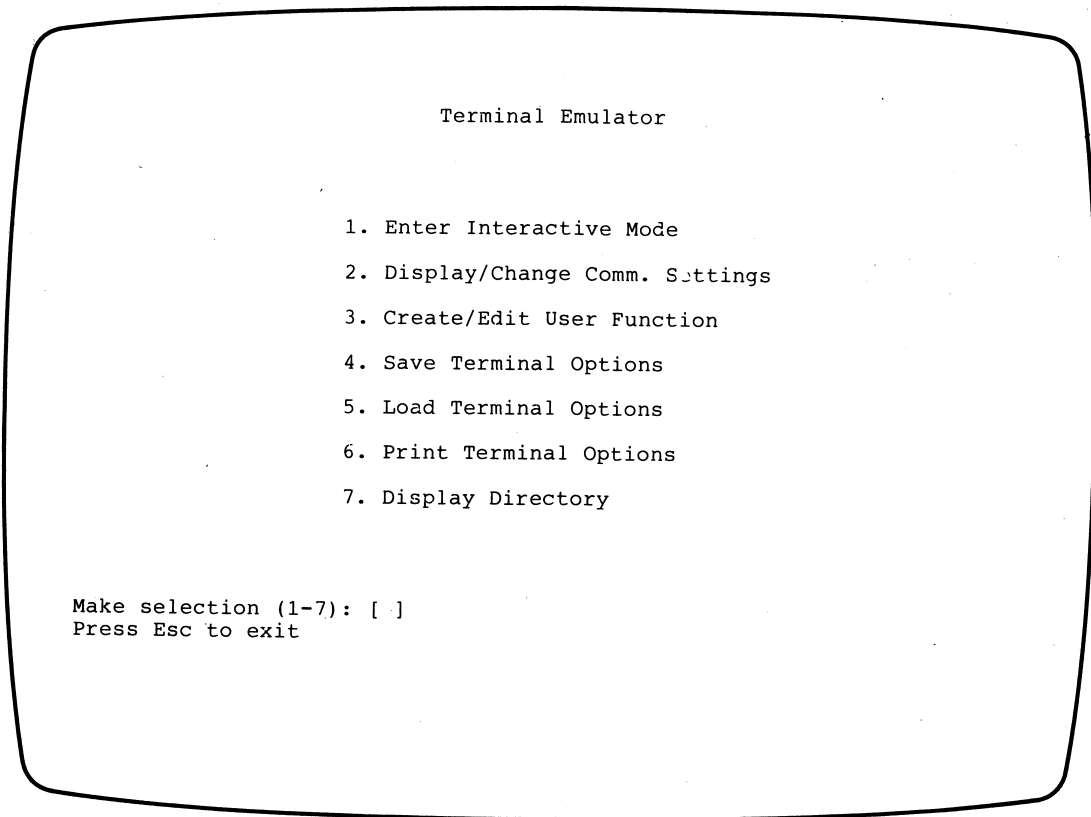




**Figure 12-2** Personal Communications Manager (PCM) Main Menu.

3. Load PCM from the Work Diskette. PCM will ask you a number of questions in order to configure the program. Several of these can be confusing even after you've read the PCM manual, but you can use the default answers (the answers already there) for most of them. Don't be concerned about the large amount of space PCM allows for answers. Short replies are fine. Figure 12-2 illustrates the PCM Main Menu.

The first two questions that cover your name and phone number are easy enough. The next asks about the editor you will be using and recommends an answer of EDLIN. Now you have a chance to put that work you did in Chapter 4 to some



**Figure 12-3** PCM Terminal Emulator Mode—preparing to go on-line.

practical use. But after you answer EDLIN.COM, you will get the disconcerting response:

FILE NOT FOUND

Don't worry about this error message, but be sure to copy EDLIN.COM to the working diskette as soon as the configuration is completed.

We recommend that you use the default settings for the next set of questions (paginated printouts, number of drives for receiving, modem type, default dialing mode, and directory log). Then indicate if you have a color or black-and-white

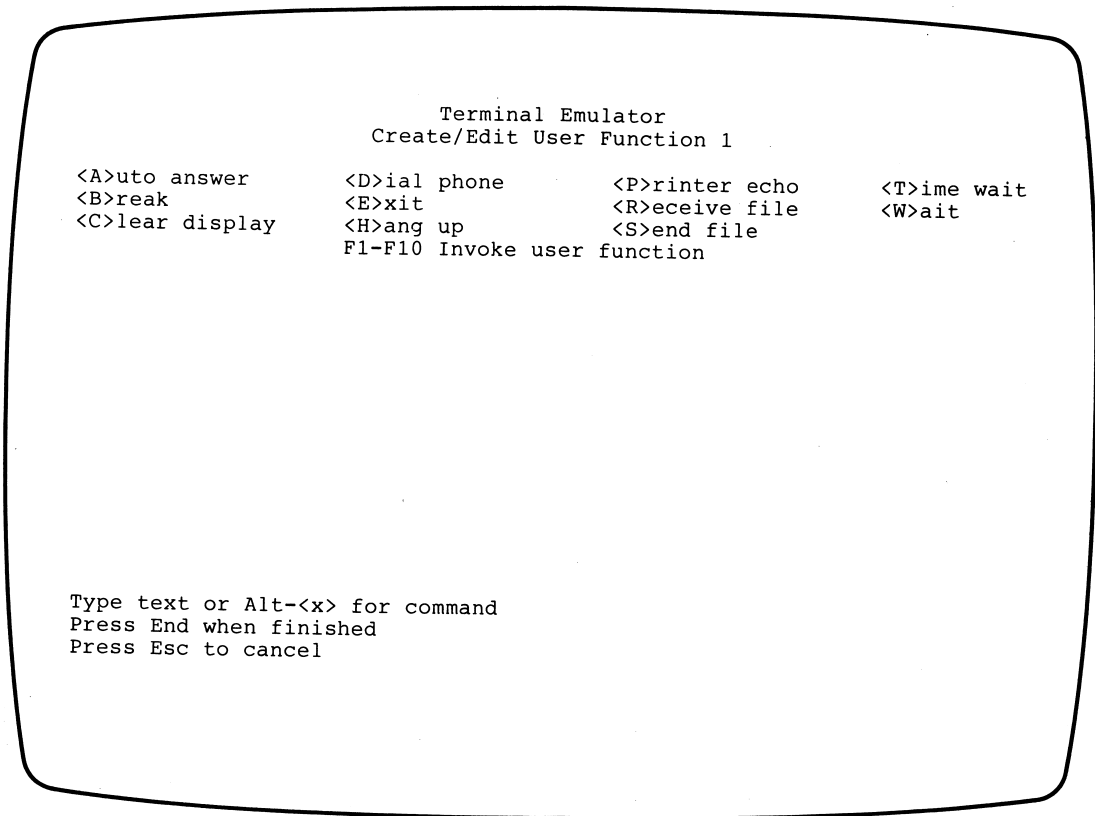
```
Terminal Emulator
Interactive Mode

ALT-A...Auto answer
ALT-B...Send break
ALT-C...Clear display
ALT-D...Dial telephone
ALT-E...Exit interactive mode
ALT-H...Hang up telephone
ALT-P...Printer on/off
ALT-Q...Help
ALT-R...Receive file control
ALT-S...Send file control
F1-F10..User functions

NO Connect  07:58:01  Printer OFF      Snd      Rcv      ALT Q--Help
```

**Figure 12-4** PCM Terminal Emulator—on-line.

- monitor. Finish by answering yes to the last question: "Is this correct (y/n)?" You are almost ready to go on-line with your computer.
4. For now, concentrate on the intelligent terminal mode emulation. Go through the example on "Connect to an Information Service" in Chapter 4 of the manual.
  5. Now you are ready to go on-line. Put PCM in the Terminal Emulation mode (Figure 12-3). If you have a subscription to Dow Jones, The Source, or CompuServe (one of the three services with predefined option files), load that file. To do this, make selection 5 (Load Terminal Options). Then enter the Interactive Mode (selection 1), and make the connection by pressing [ALT-D] (to dial the number) (Figure 12-4).



**Figure 12-5** PCM Create/Edit User Function.

Then log on. Good luck. You have just entered the electronic universe!

6. To create a personalized user function, enter the Terminal Emulator mode and choose selection 3 (Create/Edit User Function). Figure 12-5 illustrates your screen display. Turn to Chapter 6 in the PCM manual. Using the communications parameters you obtained from the service, create the function. Use a twenty-second wait between your commands as a starting value. The slashed circle symbol (Ø) is created automatically when you press [ENTER]. You do not have to enter it separately.

## CROSSTALK

PCM is flexible enough to handle an external modem, but CROSSTALK by Microstuf is easier to use. This communications software package has an excellent tutorial that lets you get started quickly and then build gradually to the finer points. One of its strongest features is that CROSSTALK uses a combination of menus and narrative descriptions to guide you through many of the complex communications setups required to link your computer with the outside world. The screen displays in Figure 12-6 illustrate CROSSTALK. If you are using an external modem or a non-IBM modem, you will be better off with CROSSTALK. Even if you do have an IBM internal modem, you should seriously consider CROSSTALK. But there is another software terminal emulator that is actually available free (as part of your operating system), so read on.

## COMM.BAS

The COMM program is part of the DOS 2.10 Supplementary Programs diskette. It was originally written in BASIC for the IBM PC, but as far as fancy communications packages go, this one is strictly plain vanilla. However, it will allow you to use your serial port, an external modem, and manual dial-up to convert your PCjr into a dumb terminal. This arrangement will mean using an acoustic coupler and your telephone or a Bell data set (their name for a modem) with dial-up capabilities to connect with the remote computer.

The COMM program offers a menu of seven choices and includes the communications settings for two information services: Dow Jones and The Source. (You must supply the communications parameters in order to use any other service.)

This free communications program can come in handy in a number of situations. For a college student COMM may be all the communications power necessary to log on to the college or university computer system. Because acoustic couplers are now out of fashion (internal modems are the current vogue), many good used ones are available at low prices. An enterprising student can add communications power to the PCjr for hundreds of dollars less than the price of a new internal modem and communications software.

```

      Novation / CROSSTALK Status Screen      Off line
Name   CROSSTALK default settings           Loaded  A:STD.XTK
NUmber                               Capture Off

----- Communications parameters -----
Speed 1200  Parity None  Duplex Full
Data 8     Stop 1      EMulate None
Port 1     Mode Call   INfilter On

----- Filter settings -----
Debug Off  LFauto Off
TABex Off  BLankex Off
INfilter On  Outfiltr On

----- Key settings -----
Atten Esc  Command ETX (^C)
Switch Home BBreak End

----- SEND control settings -----
CWait None
LWait None

----- Available command files -----
1) BBS-CA      2) BBS-FL      3) BBS-IL      4) BBS-NY      5) BBS-TX
6) CAT        7) CAT2       8) COMPUSER   9) DOW        10) NEWUSER
11) SDSU      12) SETUP     13) STD      14) UNION

Enter number for file to use ( 1 - 14 ): _

```

```

      CROSSTALK - XVI Status Screen      Off line
Name   CROSSTALK defaults (Novation modem)  Loaded  A:STD.XTK
NUmber                               Capture Off

----- Communications parameters -----
Speed 300   Parity None  Duplex Full
Data 8     Stop 1      EMulate None
Port 2     Mode Call   INfilter On

----- Filter settings -----
Debug Off  LFauto Off
TABex Off  BLankex Off
INfilter On  Outfiltr On

----- Key settings -----
Atten Esc  Command ETX (^C)
Switch Home BBreak End

----- SEND control settings -----
CWait None
LWait None

----- Available command files -----
1) NEWUSER      2) SET1200B    3) SETUP      4) STD      5) UNION

Enter number for file to use ( 1 - 5 ): _

```

Figure 12-6 CROSSTALK screen displays.

```

          _____  CROSSTALK - XVI Status Screen  _____  Off line
Name      CROSSTALK defaults (Novation modem)      Loaded  A:NEWUSER.XTK
Number    _____  Capture  Off

```

```

_____  Communications parameters  _____  _____  Filter settings  _____
Speed 300  Parity None  DUplex Full  DEbug  Off  LFauto  Off
Data 8    Stop 1      EMulate None  TABex  Off  BLankex Off
Port 2    Mode Call   _____  INfilter On  OUTFiltr On

```

```

_____  Key settings  _____  _____  SEnd control settings  _____
Atten Esc  Command ETX (^C)  CWait  None
Switch Home  BReak End  LWait  None

```

In order to make a call, CROSSTALK must first know several things. For example, you must first tell the program what the phone number is, and what modem speed you wish to use to make the call.

This script file will guide you through all of the steps necessary to help you make a call with CROSSTALK.

Press ENTER to continue: \_

```

          _____  CROSSTALK - XVI Status Screen  _____  Off line
Name      CROSSTALK defaults (Novation modem)      Loaded  A:NEWUSER.XTK
Number    408-748-8588  _____  Capture  Off

```

```

_____  Communications parameters  _____  _____  Filter settings  _____
Speed 300  Parity None  DUplex Full  DEbug  Off  LFauto  Off
Data 8    Stop 1      EMulate None  TABex  Off  BLankex Off
Port 2    Mode Call   _____  INfilter On  OUTFiltr On

```

```

_____  Key settings  _____  _____  SEnd control settings  _____
Atten Esc  Command ETX (^C)  CWait  None
Switch Home  BReak End  LWait  None

```

If you wish, CROSSTALK will display the name of the location you are calling at the top of the status screen. It is not necessary to enter this information if you don't want to.

Enter system name or comments: \_

Figure 12-6 (Continued)

```

          _____  CROSSTALK - XVI Status Screen  _____  Off line
Name      TIMES-MIRROR MOSBEY PUBLISHING      Loaded  A:NEWUSER.XTK
Number    408-748-8588                        Capture Off

```

```

_____  Communications parameters  _____  _____  Filter settings  _____
Speed 300  Parity None  Duplex Full  DEbug  Off  LFauto  Off
Data  8    STOp  1     EMulate None  TABex  Off  BLankex Off
Port  2    MObde Call   INfilter On  OUFiltr On

```

```

_____  Key settings  _____  _____  SEnd control settings  _____
ATten Esc      COmmand ETX (^C)  CWait  None
SWitch Home    BReak  End         LWait  None

```

The next thing CROSSTALK needs to know is the modem speed you wish to use. If you have a 300 baud modem, your only choices are 110 and 300. If you have a 1200 baud modem, your choices are 110, 300, and 1200 baud. Note that 110 is entered as 0110 to distinguish it from 1200.

0110, 300, 600, 1200, 2400, 4800, or 9600 baud? \_

```

          _____  CROSSTALK - XVI Status Screen  _____  Off line
Name      TIMES-MIRROR MOSBEY PUBLISHING      Loaded  A:NEWUSER.XTK
Number    408-748-8588                        Capture Off

```

```

_____  Communications parameters  _____  _____  Filter settings  _____
Speed 300  Parity None  Duplex Full  DEbug  Off  LFauto  Off
DATA  8    STOp  1     EMulate None  TABex  Off  BLankex Off
Port  2    MObde Call   INfilter On  OUFiltr On

```

```

_____  Key settings  _____  _____  SEnd control settings  _____
ATten Esc      COmmand ETX (^C)  CWait  None
SWitch Home    BReak  End         LWait  None

```

You have now entered all of the essential information to place a call. If you wish, CROSSTALK can save this information for future reference in a "command file". You may recall the command file at a later time to place a call to this particular system.

Do you want to save this setup in a command file? \_

Figure 12-6 (Continued)



In addition, many industrial and laboratory computer centers have acoustic couplers or data sets available on loan to employees. Although such organizations are not likely to have communications software for your PCjr, by using COMM and the coupler you will have a total package that will give you access to the computer center from home via your PCjr.

## Information Services

Nearly 2,000 commercially available data bases and hundreds of free public access systems already exist. Now that you have your communications support package set up and ready to operate, it is time to examine some of the information services open to your PCjr. These services range from information utilities and encyclopedic data bases to on-line news bulletins and public access message systems. All of these services are just a phone call away from your PCjr. We will start our tour of this information universe with the two services that have earned the name *information utilities*.

### The Source And CompuServe—Information Utilities

The Source, a video text service, calls itself "America's information utility," and it certainly has begun to live up to that name. After a relatively slow start in 1979, The Source reached 41,500 members by the end of 1983, and it continues to grow by leaps and bounds. Six main categories of services are available with The Source—namely, communications, news and sports, business and finance, travel, games, and consumer services. These services consist of data bases that provide detailed, up-to-date information on these topics. They also offer interactive communications, such as electronic shopping and electronic mail.

To join The Source, you have to pay a one-time initiation fee, plus connect-time fees that vary depending on the number of times you use the system and the speed of your modem. As you might expect, you can save money if you use the system after business hours and during the weekend (up to one-third of the cost). Hookup for a 1,200-baud modem costs about 30 or 40 percent more, but it lets you transfer data at four times the rate, significantly offsetting

this cost difference (provided you use the down-line and up-line loading capabilities of your PCjr).

Usually, you can get a starter package at a nominal fee to sample The Source's wares. We recommend this option before you make any bigger commitments to the service. Before you begin, we should warn you that these information utilities can be addictive, particularly the interactive communications aspects. Some of our colleagues have spent \$400 a month and more because they weren't aware of just how much time they were spending on the system. But few regretted their experiences.

Some of the special features we dabbled with are: the travel data base, the electronic mail facility, and the stock exchange listings. We even perused job listings (blind listings from major headhunters around the country). Our trip on the travel data base consisted of an excursion through the airline flight data bases of over 600 carriers throughout the world. This comprehensive listing contains flight numbers, times, and connections. If you decide to be your own travel agent, this is the place to start. The Source also offers its own electronic travel agent to handle this chore, should you decide to leave the details to them.

Membership in The Source gives you access to electronic shopping. Your shopping list can include up to 50,000 items, and their prices are a good basis for comparison. We've actually found some to be significantly less than local retail stores. One thing is for sure, they have a larger selection than you'll find at most of your favorite shopping centers.

The electronic mail service is one of the features offered by The Source that has really begun to catch on. It logs messages, eliminates "telephone tag," and lets you exchange messages with any other subscriber. Some people include their Source ID numbers—along with their telephone numbers—on their business cards. As the number of subscribers increases, the value of this kind of utility skyrockets.

You can get information about The Source by calling 703/734-7500 or toll free 800/336-3300. Source Telecomputing Corporation (The Source) can be reached by mail at 1616 Anderson Road, McLean, Virginia 22101.

CompuServe offers many services similar to The Source. CompuServe's rates are lower, but the service is available only from 6 p.m. to 5 a.m. weekdays and all day on weekends. CompuServe

claims a larger subscribership than The Source, but most of the people we know prefer The Source for all-day convenience and ease of use (although many are members of both). The choice is a matter of personal preference, so try them both before you make your decision.

You can reach CompuServe by phone at 614/457-8600 or toll free at 800/848-8900, or write: CompuServe Information Services, Inc., P. O. Box 20212, Columbus, Ohio 43220, for more information about their services and rates.

### Dow Jones

This name is known throughout the business world for outstanding publications (the *Wall Street Journal* and *Barron's*). Dow Jones also has joined the electronic revolution and is now the largest interactive information service in the country. In fact, with 115,000 users, this service is about twice the size of either The Source or CompuServe.

What does Dow Jones have that nobody else has? Nothing, really. What they have is a reputation for quality, which they live up to in a high degree in this new enterprise. When financial news breaks, it is usually the *Wall Street Journal* that publishes it first, and this system lets you sit right on that hot line. We have been tuning in to this service and find it exciting and informative.

Dow Jones has not yet entered the broad-based information utility arena, but they soon will. As of now, they have twenty-four separate data bases, which include stock information, company profiles, an Academic American Encyclopedia (with over 30,000 entries), and even movie reviews. They have added the official airline guide by Dun & Bradstreet and have teamed up with a Japanese news service. First and foremost, they are a news service, and that is the primary reason to get on their bandwagon. But they offer electronic mail service as well.

Dow Jones offers a flexible membership plan, so phone them at their toll-free number (800/345-8500, extension 49) and ask for the free trial offer (which generally is available). You can write Dow Jones & Co., Inc., at P. O. Box 300, Princeton, New Jersey 08540, for further information.

Logging on (making the connection) with these three communications services is easy, because all of the communications

settings are built into your communications software packages: PCM and CROSSTALK. Even COMM (the freebie) has the settings for The Source and Dow Jones. Local telephone access numbers are available for all of these services, so there is no long distance rate to worry about.

## Other Encyclopedic Services

There are two other "encyclopedic" data base services that you may be interested in investigating: BRS/After Dark and Dialog. These services offer an enormous amount of information on a wide range of subjects—from medicine to bilingual education. The trend toward offering such services is continuing, and you're likely to see an explosion in their growth in the near future.

Generally, they charge a per-hour rate (after the initial subscription charge) to examine the data bases. A drawback to this type of service is its sheer scope—you can spend a lot of time (and money) trying to find where the information you need is located. This venture is one you might do well to share with a fellow computer user, at least until you get the hang of using it. However, these services can revolutionize research practices and be a boon to the students in your household, once you know how to make efficient use of them.

All the services come with instruction manuals (available at a separate charge or as part of the initiation fee). Unfortunately, the manuals are not too easy to read.

BRS/After Dark is available from 6 p.m. to midnight. It is located at 1200 Route 7, Latham, New York 12110. The phone number is 518/783-1161, toll free 800/833-4707.

Dialog Information Services can be reached toll free at 800/982-5838 in California (800/227-1927 elsewhere), or you can write them at 3460 Hillview Avenue, Palo Alto, California 94304.

## Public Access Systems

There are hundreds of free public access computer bulletin board systems. These bulletin boards contain information on games, programs, religious topics, and even sexually oriented messages. For

the most part, however, they are used for mail and information interchange. Like most of the communications-oriented information services, they have mushroomed in size since the first public access message system was established in Chicago in 1978. By the way, that pioneer still is going strong.

However, most of these public access facilities are shaky and unstable, and you need a daily score card to keep track of the active ones and their operating times. Fortunately, a free listing of these services is available through the Novation twenty-four-hour computer bulletin board. To get the listing, set your computer parameters to 300 baud, eight data bits, one stop bit, and no parity. Then call 213/881-6880 (in Los Angeles) to log on. Once the connection is made, press [ENTER] and wait for the log-on message to appear on your screen. Then type:

### CAT

(in upper-case letters) to obtain the menu listing of the various offerings.

These on-line data bases and communications services also are listed in three publications that you may want to look at: (1) *Directory of On-Line Databases*, published by Cuadra Associates, 2001 Wilshire Boulevard, Suite 305, Santa Monica, California 90403 (213/829-9972); (2) *The On-Line Computer Telephone Directory*, published by J. A. Cambron Co., Inc., P. O. Box 10005, Kansas City, Missouri 64111 (816/756-1847); and (3) *Omni On-Line Data Base Directory*, published by Collier Books, 1983.

Your own public access "grapevine" can put you in touch with other services and fill you in on the value of some of them. We have used public access networks and found them to be extremely erratic. Because they depend on volunteer labor, their quality fluctuates routinely.

## Further Reading

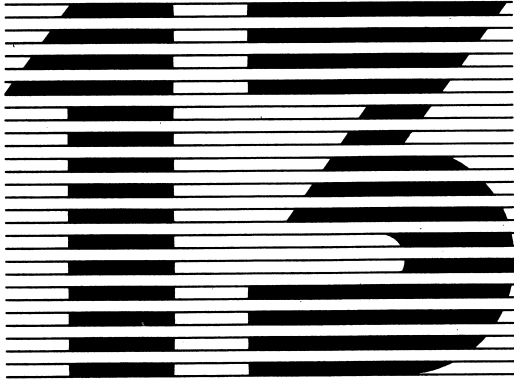
*The Complete Handbook of Personal Computer Communications*, by Alfred Glossbrenner (St. Martins Press, 1983). The book is not as complete as its name boasts, but it is one of the better

books on the subject. It is not oriented to the PCjr, however, so expect to modify much of the material. Nevertheless, if you're serious about telecommunications, you will want to examine this fine reference.

*Crosstalk XVI* (Microstuf). The reference manual for the CROSS-TALK communications program. A good manual that facilitates the use of this communications package.

*Personal Communications Manager Manual* (IBM Corporation). Presents the documentation for the PCM program, but not to the quality of the other IBM manuals.

*The Small Computer Connection*, by Neil L. Shapiro (Micro Text, 1983). An Apple-oriented communications book with a large listing of public access systems. Most of it is not particularly relevant to the PCjr environment, but the book is worth looking at if you can borrow a copy.



## Success Beyond The PCjr: Where Do You Go From Here?

We titled this book *The Complete Guide To Success With The IBM PCjr*, but the book would not live up to its name and be “complete” without a look at what the future holds for you and your PCjr. You see, one of the greatest design features of the PCjr is its expandability. Happily, your computer is prepared to grow as you grow!

The PCjr is so new, however, that it still can't do everything other computers can do—even tasks that theoretically are within its grasp. But as the history of the computer industry has shown, whenever a void exists between capability (theory) and technology (actual hardware and software), it is soon filled by an enterprising supplier with a novel solution. Consequently, you can expect an explosion of new possibilities at any time.

With that in mind, our goal in this final chapter is to prepare you just as thoroughly for what *may* be available in the near future as for what currently *is* on the market as optional additions to your PCjr.

### The Future And Your PCjr

When you are ready to expand your PCjr, there are many different directions you can take. You can upgrade your memory capacity, storage devices, communications capabilities, printer technology, and image processing—to name just a few. Naturally, as the capacity of your hardware increases, the range of software available to you expands dramatically as well. For example, witness all the soft-

ware available to owners of the enhanced version of the PCjr (because it contains 128K of memory and a disk drive) that is unavailable to you if you own the entry model. Now, multiply by a factor of ten to get the amount of software available to owners of a double disk drive system.

Even though the PCjr is capable of nearly unlimited expansion, there comes a time when upgrading to a more powerful system is the only alternative—if you want to meet your needs completely. Therefore, in this chapter we will examine the other personal computers currently available (both from IBM and other suppliers), with an eye toward deciding which ones would be the logical choices for moving up in the computer world.

Lastly, we will try to anticipate some of the future trends in the home computer industry, such as increasing the interchangeability of software. And most important, we will show you what these trends will mean to you as a home computer owner.

Remember, the toughest decision about personal computing is the one you have already made: Should I buy a computer in the first place? All the other “crises” —Should I get a dot-matrix or letter-quality printer? Or, Oh my gosh, do I want VisiCalc or MultiPlan as my spreadsheet?—are merely the choices you make to get the most value out of your initial decision, which was to invest in a computer.

There's other good news, too. The next big “breakthrough” probably will be so unforeseen (yet so simple in retrospect) that it will come as a surprise to everyone. All the challenges and decisions that this new breakthrough creates for you will be yours to handle on your own. Have no fear. No matter how big the breakthrough—or how expensive to implement—the basic principles will be the same as we set out in this section and throughout the book. Those basic concepts won't change regardless of what comes along to “revolutionize” your computer. So you have already accumulated all the tools you'll need to participate in the exciting and profitable world of personal computing.

You may never expand your PCjr one bit (though that's unlikely), or you may find yourself (in three or four years perhaps) with two printers, a fixed disk, 440K of RAM, and a light pen. Either way, your PCjr should work for you—not the other way around.

Keep one rule in mind at all times: If you don't know what you're going to do with a new device or program, *don't* buy it and



find out after it's too late. Know what you need and how you'll use it before you buy, and you will rarely be disappointed in the purchase. Otherwise, prepare extra shelves for these potential dust collectors. That's the only "job" most of them will be doing.

## Hardware Enhancements For The PCjr

The two major hardware enhancements you can expect to see very soon are an 83-key keyboard and a second disk drive. From a technological point of view, neither of these enhancements represents a difficult engineering problem. The difficulties are purely economic. The keyboard must be in the \$100 range to be a viable alternative to the one that comes as part of the system. At present several \$200-to-\$300 83-key units are offered for the PC, so a complete function-key version of the keyboard for the PCjr should not be long in coming. (We saw an 83-key test unit while we were preparing this book.) The advantage to the 83-key keyboard is that it eliminates the necessity to double up keys—by pressing a function key and a number key together, for example—thus making typing simpler and faster.

A second disk drive will be a welcome addition to your system. Convenience aside, it will open up a new world of software for the PCjr. There are more design problems facing the engineers working on drive B than those facing the keyboard builders. The drive will have to be housed externally, and the problem of a power supply must be solved. But we are confident that the enormous potential market will spur the designers to come up with a reliable second drive soon. We expect the price to be in the \$350-to-\$400 range initially and then to drop steadily as competition builds.

The 128K barrier won't be there long. That's the next big breakthrough. The engineers can approach this problem in two ways. Either they can offer an alternative to IBM's memory module, or they can design an external memory unit and build it into the same housing as the second disk drive. Using an external unit with added memory and storage capacity has been a common practice in the industry. It's usually called "expansion interfacing." Both of these solutions probably will be in the marketplace within a year.

The other useful add-on that should be right around the corner is a clock/calendar card. This unit will automatically enter the time

and date when the computer is powered up. Similar units are available for the PC in the \$80-to-\$100 range, so the technology is at hand.

Whatever the actual design solution, we forecast that you will soon be able to own a double-drive, 256K PCjr "Plus" with a clock/calendar card for about \$700 in add-on charges (or even less).

Printer technology is already undergoing a remarkable development. In fact, two areas are now on the verge of becoming the next wave of personal computer support units. These are ink-jet printing and laser printing. They offer quiet, fast, letter-quality printing on plain paper—just the ticket for the personal and home computer. Stay on top of what goes on in this area and check the market before you purchase that upgraded printer.

Don't write off the low-cost alternatives to IBM equipment. The market will be flooded with excellent IBM look-alikes, from color monitors to joysticks. For example, Amdek produces an outstanding line of monitors that rival IBM in quality and performance, despite their lower prices. Taxan, USI, and Princeton monitors are other viable alternatives to the IBM color monitor. Wilco makes a very fine joystick. And Epson printers are good enough for IBM (Epson manufactures the IBM graphics printer), so there's no reason to worry about the quality of this product.

The only pitfall to look out for is the service. There is a distinct advantage to having a homogeneous system—and a vendor as reliable as IBM to back up all the components. Shop around, by all means, but keep these qualities in mind. Don't be blinded by a bargain price.

## Software Breakthroughs

We believe the most exciting new developments will come in the area of software—under three classifications: software conversions, new applications, and transfer/access software. Let's look at each of these areas separately.

Literally thousands of software packages are available for the PC, many of which easily can be converted to the PCjr. The driving force here is the marketplace. Entertainment and educational software will be converted in droves—because the demand is so fierce.

American Educational Computer, Inc., has a line of fine educational programs for youngsters that now are available for the PCjr. Harcourt Brace Jovanovich is converting its SAT test preparation package to the PCjr, and you can be sure others will soon follow.

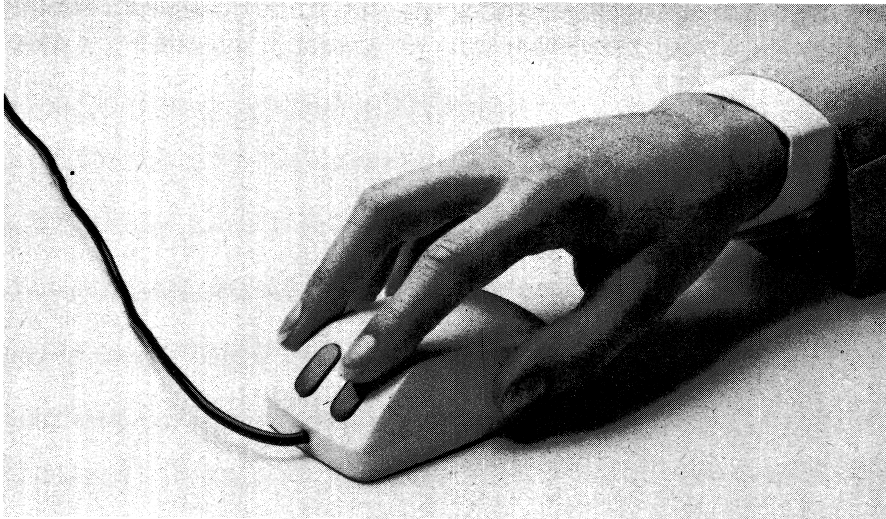
The going will be a bit slower in converting personal productivity programs. The bulk of these programs require two disk drives and at least 128K memory. The best seller for the PC is the integrated spreadsheet–data base package, Lotus 1-2-3, which requires 192K. Other packages need at least 256K. Therefore what you'll probably see first are several of the more popular word processing programs, such as WordStar and Volkswriter, scaled down for the PCjr before any of the other packages filter down.

The next area of expansion will be the emergence of a new type of software specifically for the home market. We are calling this software "personal development packages." They will include programs for diet, exercise, health care, language development, and other topics that will help you help yourself—self-improvement programs, if you like. Such programs are not available in great numbers as yet, but they soon will be. And they will make a real dent in the market.

The last frontier will be what we call "transfer and access software." Included in this class of programs will be packages that allow users to convert files between computer systems. This will make it possible for users to upgrade their systems without losing all the files and programs they developed for less sophisticated models. An example of this is Alpha Software's Apple/IBM Connection. This program is designed to move files from the Apple to the PC and vice versa. You can expect to see variations of this transfer capability for other home computers.

Access software is a special simplified software that allows you to hook up with large commercial data bases. Firms like Dow Jones are already offering software packages for communications containing their news and financial services. And we expect that PCjr versions will be available shortly, as the demand increases.

You will soon see a new set of software that uses an input device called a "mouse" to improve personal productivity and reduce the keyboarding demands associated with computer interaction. With a mouse you can move the cursor around the screen without even touching the keys. You merely roll the mouse around your desk or table surface in the direction you want the cursor to move,



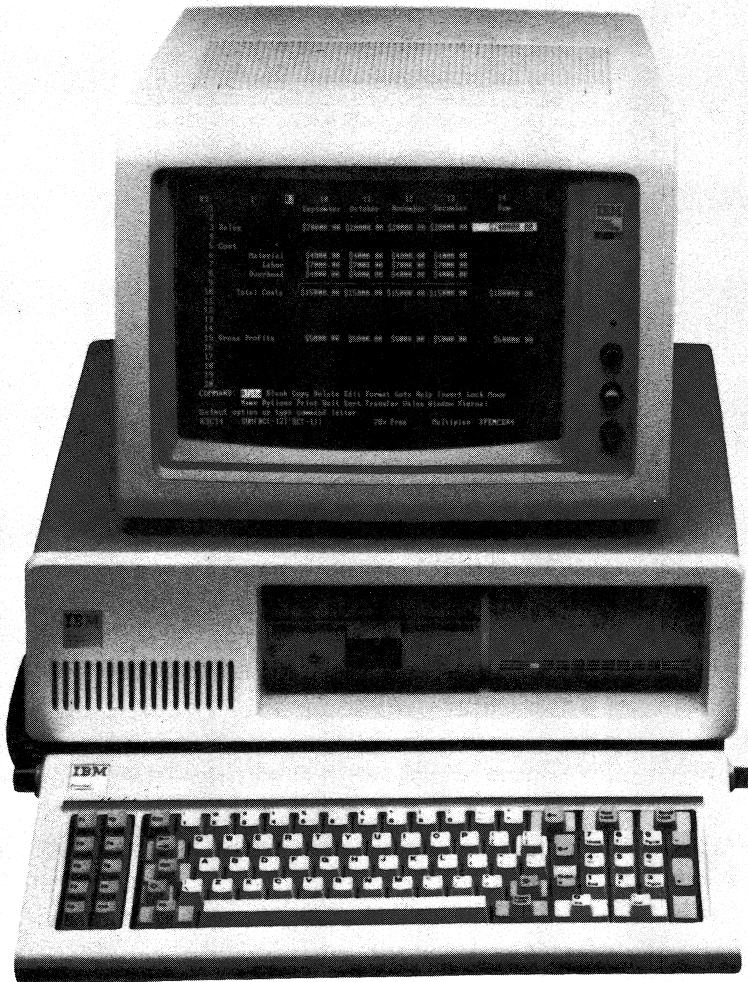
A "mouse" enables a user to move the cursor around the screen without touching the keyboard.

and the cursor responds. Buttons on the mouse allow you to trigger actions in the program. This handy device will be especially beneficial for poor typists, as it lessens the need for typing skills.

One of the strengths of the PCjr is its outstanding sound and graphics capabilities. The current software does not exploit these potentials, but you will see more efficient use of both capabilities soon. For example, graphics packages—like 4-Point Graphics for the PC—can be used to draw pictures, designs, and business graphics. These packages surely will be modified and expanded for use on the PCjr.

## **Beyond The PCjr**

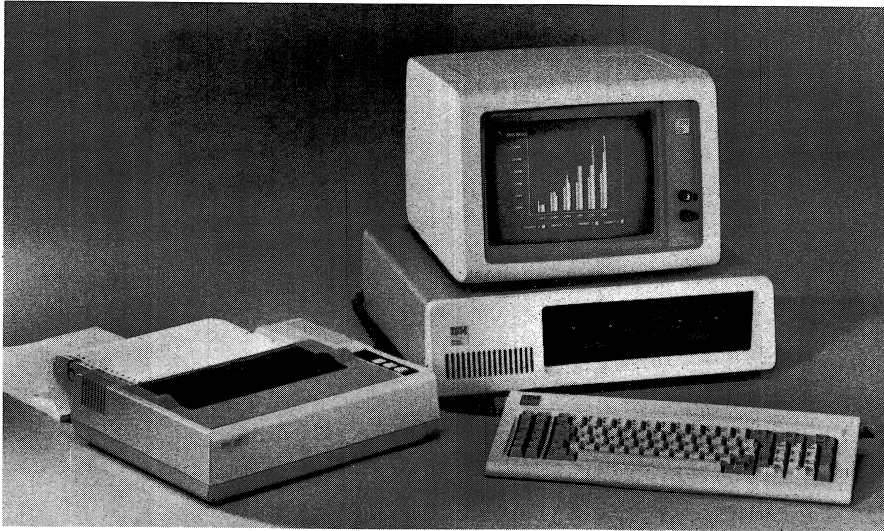
Where do you go from here? The answer is as complex as your needs are, but you have already examined all the options open to you and your PCjr in these pages. Bear in mind, though, that after a point adding more hardware components to your PCjr just won't pay. Once you have invested more than \$800 in expanded features for



The IBM PC, considered by many people to be one of the finest personal computers on the market.

this system, you will have to seriously consider moving up the computer line.

Fortunately, you can move up from the PCjr without sacrificing the software you have worked so hard to develop and accu-



The IBM PC XT, which has a ten-megabyte hard disk storage unit. The IBM "family" of personal computers consists of the PC XT, the PC, and now, the PCjr.

multate. You can move right over to an IBM PC. The PC is more expensive, of course, but it will increase your horizon for expansion many times, and it is compatible with the entire IBM product line. With the PC you can enlarge your memory capacity by a factor of four. The five expansion slots in the PC allow you to choose from a wide range of options that currently are not available on the PCjr.

Even beyond the PC, the IBM family includes the PC XT—a very powerful personal computer with a 10-megabyte hard-disk storage unit. This machine is flexible enough to handle all the needs of a small business, and it is unlikely that any personal computer user will "outgrow" its potential for expansion. Somewhere down the line, this is probably the computer you will want for your home—even if your only "business" is personal.

The PCjr even offers you the opportunity to step outside the IBM product line to upgrade. Machines like the Compaq, the Chameleon by Seequa, and the Columbia PC are all compatible enough with IBM products to be potential candidates for your expansion. But before you leave Big Blue, check the performance of your pet software on these machines.

## The Last Word

We've gone about as far as we can go together, but your journey in computer land is just beginning. You're as ready as you'll ever be to go it alone, so bon voyage!

# Appendix A

## Data For pfs:FILE And pfs:REPORT Data Base Example\*

### Form 1

Brown, Kay W.  
63 Drew Place  
Newport Beach CA 93661  
1631267 Badasci S  
Breast cyst

80-04-11

\*  
\*

84-02-03

---

Cyst aspiration

80-04-11 Benign

### Form 2

Connors, Jason  
11 Winchester Ave  
La Mesa CA 92119  
0904409 Handley J  
Rt inguinal hernia  
Hypertension, cardiac disease  
HCTZ, Inderal

83-12-30

84-01-11

\*See Chapter 10.



---

Aortocoronary bypass	82-06-03 ASCVD
Rt inguinal hernia repair	84-01-04 Sched

**Form 3**

Crenshaw, Benjamin	84-01-03
8990 Milton	
San Pedro CA 90105	
2142385 Badasci S	
Cholelithiasis	
Emphysema	

\*

84-01-04

---

Tonsillectomy	28-99-99 *
{Remarks/Other} Ultrasound scheduled AM 84-01-04. If positive, admit for surgery	
[* means no data for this entry]	

**Form 4**

Dover, Bennett	77-10-01
87 Spring St	
Los Angeles CA 90034	
850040022 Badasci S	
Colon CA	
Cirrhosis	

\*

84-03-14

---

Cholecystectomy	69-03-99 Gallstones
Rt hemicolectomy	77-10-17 Dukes B CA

**Form 5**

Gardner, Roger	83-12-14
123 Hope St	
San Rafael CA 94112	

NONE Handley J

Thyroid storm

\*

Synthroid

84-04-05

---

Subtotal thyroidectomy

83-12-14 Graves disease

### Form 6

Levitt, Irene

83-08-22

63380 Sepulveda Blvd

Inglewood CA 90067

2330766 Carlson F

Adrenal adenoma

Hyperparathyroidism

Cimetidine, prednisone

84-03-21

---

Parathyroidectomy

81-09-11 Hyperplasia

Lt adrenalectomy

83-08-28 Adenoma

### Form 7

Martin, Scott

80-04-23

123 Hilgard Ave Apt 3

Los Alamitos CA 90025

NONE Sprule B

Gastric polyps

Cardiac disease

\*

\*

---

Partial gastrectomy 81-01-12 Polyps

{Remarks/Other:} Expired 81-01-12, bleeding

## Appendix A

**Form 8**

Trapletti, Fredrick  
 8 Norris Center Rd  
 Beverly Hills CA 90010  
 3450876 Christie W  
 Pheochromocytoma  
 MEA-IIB

80-06-24

\*

84-01-04

---

Rt adrenalectomy  
 Subtotal parathyroidectomy

80-06-28 Pheochromocytoma  
 82-10-10 Hyperplasia

**Form 9**

Webb, James  
 3445 Henderson Dr  
 Lake Tahoe NV 87760  
 1342586 NONE  
 Laryngospasm  
 Chronic renal failure  
 Cimetidine, Vit D

83-05-11

PRN

---

Tracheostomy  
 {Remarks/Other:} Returned to Nevada for F/U care

83-05-11 Spasm

**Form 10**

Young, Gene E.  
 4990 Landover  
 Westchester CA 90087  
 3234258 Economou D  
 Rt breast mass  
 Asthma, allergy to iodine  
 Quibron (PRN)

83-05-09

84-01-06

---

Rt breast bx

83-05-09 Benign

# Appendix B

## PCjr Technical Specifications

This appendix covers the technical specifications for your IBM PCjr. Some of the equipment listed here is standard with the basic PCjr system. Other equipment is included in the enhanced version. The remainder is optional equipment that can be added at any time. Each is listed under the appropriate category.

### PCjr System Unit

Microprocessor—Intel 8088:

- 4.77 Megahertz (Mhz) clock speed
- Cycle time: 210 nanoseconds (ns)

Memory:

- Read only memory (ROM): 64 kilobytes (KB):
  - Cycle time: 375 nanoseconds (ns)
  - Access time: 250 ns
- Random access memory (RAM): 64 KB:
  - Cycle time: 210 nanoseconds (ns)

Ports and interfaces:

- Serial—RS232
- Two ROM cartridge slots
- Cassette
- Two joysticks
- Keyboard
- Modem
- Light pen
- Direct drive video

- Composite video
  - Television
- Input/output (I/O) expansion bus  
 Sound subsystem: TI 76496  
 Dimensions and weight:
- Width: 13.9 inches (355.4 centimeters (cm))
  - Depth: 11.4 inches (20 cm)
  - Height: 3.8 inches (9.7 cm)
  - Standard model weight: less than six pounds (2.72 kilograms (kg))
  - Enhanced model weight (with diskette drive): less than nine pounds (4.09 kg)
- Transformer:
- Separately housed, step-down 60-volt ampere type
  - Weight: 2 pounds, 13 ounces (1.27 kg)
- Operating specifications:
- Electrical:
    - 120 VAC 60 Hz power
    - 33 watt, three-voltage-level, two-stage power supply
    - 10.06 feet (3.08 meters) power cord
  - Temperature: 60–90°F(15.6–32.2°C)
  - Humidity: 8–80 percent relative humidity
  - BTU output: 164 BTU/hr
  - Noise level: 45 decibels
  - Altitude: to 7,000 feet (2,134 meters)

### **PCjr Enhanced Model—Additional Features**

- Diskette drive  
 64 K RAM memory (total memory 128K)  
 Display expansion (80 column and full color graphics)  
 Color graphics capability
- Alphanumeric
    - 40 by 25 color
    - 80 by 25 color
  - Graphics
    - 320 by 200 4 color
    - 320 by 200 16 color
    - 640 by 200 4 color
    - 160 by 200 16 color

## Keyboard

62 keys, including function control and cursor control keys

Tilt: 5- or 12-degree angle

Power requirements: four AA batteries (alkaline batteries recommended)

Optional keyboard cord: may be necessary if more than one unit is operating in the same room

Dimensions and weight:

- Length: 13.45 inches (34.15 cm)
- Depth: 6.61 inches (16.8 cm)
- Height: 1.06 inches (2.6 cm)
- Optional cord: 6 feet (1.83 meters)

## PCjr Diskette Drive Option

Self-contained, slim-line, 5¼-inch double-sided disk drive

360K memory using double-sided diskettes

Can read/write on single- or double-sided diskettes

Specifications:

- 512 bytes per sector
- 9 sectors per track
- 6 ms track-to-track access time
- 250 kbits per second data transfer rate
- 48 tracks per inch track density
- 40 tracks per side (80 total)
- Modified frequency modulation (MFM)
- 300 RPM
- Includes power cable and data cable

Dimensions and weight:

- Width: 5.8 inches (14.6 cm)
- Depth: 8.3 inches (20.8 cm)
- Height: 1.6 inches (14.6 cm)
- Weight: 2.2 pounds (1.1 kg)

Operating altitude: 7,000 feet (2,134 meters) above sea level

## PCjr 64K Memory And Display Expansion

Increases system memory size by 64K to 128K; permits the use of higher density video modes and 80-column text support

## Specifications:

- 8 64K by 1, 150ns, random access memory modules
- Hooks up to 44-pin connector on system board

**Joystick**

Offers two-dimensional control for games and other applications

## Specifications:

- Two-dimensional resistive positioning control
- Two push-button switches
- Single-pole, single-throw push buttons
- 0–100K ohms linear resistance range

**PCjr Carrying Case**

For storing and transporting system; not to be used as shipping container

Includes capacity to store:

- System unit
- 62-key keyboard and cable
- Parallel printer attachment
- 4 program cartridges
- 5 diskettes
- Transformer and cord
- TV connector

Dimensions:

- Width: 20 inches (50.8 cm)
- Depth: 6.25 inches (15.9 cm)
- Height: 18 inches (45.7 cm)

**PCjr Compact Printer**

A lightweight thermal printer

Specifications:

- Thermal technology
- Serial interface
- APA graphics (all-points addressable)
- Paper:
  - Friction feed
  - 8.5-inch width maximum (thermal sensitive)

- Includes power cord and connector cable

Dimensions and weight:

- Width: 12.3 inches (31.2 cm)
- Depth: 8.7 inches (22 cm)
- Weight: 6.6 pounds (3.0 kg)

Acoustics: 55 decibels

Power: 36 watts

Print modes:

- 80 characters per line: 10 CPI
- 40 characters per line: 5 CPI
- 68 characters per line: 8.75 CPI
- 136 characters per line: 17.5 CPI

Print matrix:

- Thermal head technology
- 5 by 8 dot matrix
- 5 by 7 for all alphabetic characters (except lower case)

Interface:

- Serial, EIA (modified RS232C)
- 1200 bits per second
- 256 character buffer
- 8-bit ASCII

Print speed:

- 50 cps in standard character print
- 2400 dots per second in image print
- Throughput: usually 25 cps

Printing mode: Unidirectional (left to right)

Print quality:

- 480 dots per 8-inch line
- 8 dots vertically

Line spacing: Program selectable at 6 or 9 lines per inch

Character set:

- Full 128 ASCII
- 191 printable characters in print ROM

Voltage: 120 VAC 50/60 Hz

Diagnostic capabilities:

- Electromechanical self-test plus system-driven isolation to customer
- Replaceable unit (printer level)



### Personal Computer Color Printer

High-speed, near letter-quality printer that produces 8 colors  
Specifications:

- Parallel interface
- Paper

Tractor, manual, or automatic cut-sheet feed

Single sheet or continuous-roll fan fold

5 inches (12.7 cm) to 14.875 inches (37.5 cm) width

Dimensions and weight:

- Width: 21.6 inches (54.9 cm)
- Depth: 12.4 inches (31.5 cm)
- Height: 9.1 inches (23.1 cm)
- Weight: 36 pounds (16.3 kg)

Print mode and specifications:

- Bidirectional, horizontal and vertical paper movement
- 200 cps draft mode (standard)
- 120 cps correspondence (proportional spacing)
- 110 cps correspondence (standard spacing)
- 35 cps near letter quality

Print quality: emphasized, offset, double pass

Print head:

- 9 wire staggered design
- 200 Mcharacter life

Programmable features: line spacing, print mode, intercharacter spacing, margins, forms length, tabs, automatic underline

Line spacing: 6 or 8 lines per inch, switch selectable

Dot plot graphics: 82.5 by 82.5 dots per inch, 9-wire head matrix

APA capability

Character set: 96 character upper- and lower-case ASCII and true lower-case descenders

Ribbon: "easy change" cartridge, 0.75 inches, 36 yards, 1.5-million-character, 4-color ribbon (cyan/magenta/yellow/black) and color mixing of 8 colors

### PCjr Parallel Printer Attachment

Used to connect the PC Graphics Printer to the system unit

Specifications: attaches to right side of system unit to support PC Graphics Printer

## PC Graphics Printer

Program compatible with the Parallel Printer Adapter Card for the PC and PC XT

Dimensions and weight:

- Width: 1.25 inches (24.4 cm)
- Height: 3.0 inches (7.6 cm)
- Length: 9.6 inches (24.4 cm)
- Weight: 12 ounces (336 grams)

## PCjr Internal Modem

Provides access via telecommunications to data sources and systems throughout the country over telephone lines. Can be installed easily and requires no system reconfiguration

Specifications:

- Automatic dial, dual-tone modulated frequency (DTMF), Touch-Tone or pulse dialing (rotary) by software command
- Automatic/manual answer and originate
- Programmable via ASCII characters
- Bell 103 series compatible
- 300 bits per second data transmission rate
- FCC part 68 approved direct connection to telephone line
- Includes asynchronous communications element
- Includes error detection and diagnostics
- Internal diagnostic capabilities:
  - Loopback controls for link fault isolation
  - Break, parity, overrun, framing error simulation
- Includes cable
- Uses modular phone jack
- Call progress reporting
- Dial tone, ring-back, and busy signal detection
- Communications mode is full duplex on 2 wire-switched network channels (typical home)
- Tandem dialing
- Line break detection
- Status reporting capabilities
- Programmable interface:
  - 7 pt 8-bit characters
  - Even, odd, or no-parity bit

- 1 stop bit generation
- Baud rate generation
- Prioritized interrupt controls

### **PCjr Connector For TV**

Interconnects with home television set for use as display unit  
Specifications:

- Easy hook-up to system unit
- Attaches to VHF antenna terminals
- 2-channel selection switch (supports channel 3 or 4)
- For U.S. and Canadian TV sets
- Certified for an FCC Class B computer
- Sealed radio frequency (RF) modulator
- Weight: 12.5 ounces (350 grams)

### **PCjr Adapter Cable For Cassette**

Connects cassette recorder/player to system unit so device can be used to store computer programs and data

Requirements:

- Belden Style 51 (or equivalent) miniature phone plug (auxiliary)
- Belden Style 51 (or equivalent) miniature phone plug (earphone)
- Belden Style 56 (or equivalent) subminiature phone plug (remote)

### **PCjr Adapter Cable For Color Display**

Designed to link PCjr to IBM PC computer color display  
Specifications: Certified for an FCC Class B computer

### **PCjr Adapter Cable For Serial Devices**

Designed to link PCjr to serial devices such as printers and external modems

Specifications:

- Allows connection of typical RS232-C signals
- Standard RS232-C D-type 25-pin connector
- Length: 3 inches (7.2 cm)

# Glossary

**Acoustic coupler** A modem that utilizes the telephone handset to transmit and receive information over telephone lines

**Alphanumeric** A combination of letters and numbers in data

**ASCII code** A computer code that represents data as seven or eight bits (depending on the version). ASCII (pronounced as-key) is usually the code used for transmitting data across telephone lines

**Asynchronous transmission** A type of data communications that transfers data over telephone lines at a rate dependent on the length of the instruction (as opposed to a time-dependent or synchronous mode)

**BASIC** Acronym for *Beginner's All-purpose Symbolic Instruction Code*, a high-level programming language designed to be easy to learn and easy to use

**Baud** A unit of measurement for the number of bits transmitted over a communications line

**Benchmark** A program used to measure the speed and performance characteristics of a computer system

**Bit** Acronym for *Binary Digit* (0 or 1)

**Block** A unit of storage on a diskette or a unit of text in editing

**Board** Short for circuit board; see *Circuit board*

**Boot** To start up the computer with the operating system

**Buffer** A storage area that holds data temporarily (during processing)

**Bug** An error in a program; see also *Debug*

**Byte** A binary character string, usually eight bits in length

## Appendix B

- Cartridge** A printed circuit board enclosed in a plastic case that contains programs for use in the computer
- Cassette** A reel of magnetic tape encased in plastic, used to store programs and information for personal computers
- Chip** A tiny integrated circuit that contains thousands of electronic components
- Circuit board** A laminated plastic board containing circuits and sockets for integrated circuit chips; see also *Board*
- Code** The actual computer language of the program
- Coding** Translating a problem into a programming language
- Command** A computer instruction that is executed immediately
- Constant** A quantity that remains unchanged during a computer program
- Cursor** The flashing bar of light on the screen of a video monitor that indicates where the next character will appear
- Data base management system** A set of programs to store and retrieve information
- Data pointer** A pointer that keeps track of items read from DATA statements
- Debug** The process of removing errors from a program; see also *Bug*
- Default** Any option that is built into the system or program
- Demodulate** The translation of data from tones back to bits after they are transmitted over telephone lines; see also *Modulate*
- Directory** A listing of all the files on a diskette; includes the size, date, and time of creation or last revision
- Disk** Magnetic disk used to store information
- Diskette** A flexible magnetic disk, enclosed in a protective container, used to store information (also known as a floppy disk); see also *Floppy disk*
- Documentation** A collection of documents, such as training guides, user manuals, and flowcharts, for using a system or program
- DOS** Acronym for *Disk Operating System*, the operating system for IBM personal computers (comes in different versions—version 2.10 is the current version for use with the IBM PCjr)
- Dot-matrix printer** A kind of printer in which letters and symbols are formed by a collection of dots
- Down-line loading** Transferring data and programs from a remote computer to a local one, usually by telecommunications linkup; see also *Up-line loading*

**Dumb terminal** A computer terminal with no processing or storage capabilities; capable only of sending and receiving data; see also *Intelligent terminal*

**EDLIN** The line editor supplied for use with DOS

**Electronic mail** A system of transferring messages between computers at various sites via a telecommunications network

**Electronic pulses** Abrupt voltage changes that to the computer represent a series of 1's and 0's

**Extension** A three-character suffix for a file name

**External command** A command requiring the insertion of the DOS diskette into the drive in order to be carried out; see also *Internal command*

**Field** The smallest accessible unit of information in a file

**File** A collection of related data organized in the form of records and fields

**Filename** A file identification used by DOS

**Floppy disk** Informal name for diskette; see also *Diskette*

**Formatting** Setting up the arrangement of data on a medium, as in formatting a diskette

**Full duplex (FDX)** A communications mode in which each end can simultaneously send and receive data over transmission lines; see also *Half duplex*

**Function** A subprogram or a procedure that returns a value depending on the parameters entered

**GIGO** A popular expression in computer jargon that means if you don't have good data to begin with, you won't get good results from your computing efforts; an acronym for *Garbage In, Garbage Out*.

**Graphics** Pictorial material or graphs produced with the aid of computers

**Half duplex (HDX)** A communications path capable of sending and receiving, but not simultaneously; see also *Full duplex*

**Hard copy** Printed or permanent output as opposed to data displayed on a video monitor; see also *Soft copy*

**Hardware** The physical equipment, such as the computer and the printer, as opposed to the programs; see also *Software*

**Help file** A file contained in a program that displays instructions about the commands and functions to serve as a guide for inexperienced users

**Hertz** A single unit of frequency that equals one cycle per second

**Hidden files** Part of the operating system; cannot be accessed and do not appear on the directory

**Home position** The upper left-hand corner of the display screen

**Independent variable** A variable whose value is represented by a single number or string; see also *Variable*

**Initialize** To set up a file

**Input** The data or the device that handles data entering the computer and enables the computer to perform the processing operation; see also *Output*

**Integrated circuit** A chip containing large numbers of electronic components

**Intelligent terminal** A sophisticated terminal or personal computer that can store and process as well as send and receive data over a communications network; see also *Dumb terminal*

**Interactive** A program that accepts direct input from the user at each stage of operation and involves the user with each decision

**Internal command** A command that stays inside the computer memory and thus does not require that the diskette from which it originated be in the disk drive for its execution

**Internal storage** Storage available to the computer directly without the use of input/output channels

**Joystick** Movable control handle for inputting data directly on the display screen of the computer (usually used for game playing)

**Justification** Aligning the right or left margin of text

**Kilobytes** An abbreviation for 1,024 bytes; often written as kBytes or kB; see also *Megabytes*

**Label** An identifying remark or title; an identifier for a particular portion of a program

**Line editor** A program used to create and modify files on a line-by-line basis; see also *Screen editor*

**Line printer** A printer that produces a complete line of type in a single operation (usually used with large, high-speed computers)

**Log on** The process of gaining access to a communications service via a series of commands and passwords

**LOGO** A programming language developed to teach children about computers and programming

**Loop** A sequence of instructions that is repeated until a condition is satisfied

**Megabytes** One million bytes; often written as MBytes or MB; see also *Kilobytes*

**Memory** Synonym for storage or the device used to store information within the computer

**Menu** A program display that lists commands or options and invites the user to choose one of the selections

**Microcomputer** A small computer that uses a microprocessor as its processing element

**Microprocessor** A central processor entirely contained on a single chip

**Modem** A device that interfaces with the computer to transmit data over telephone lines, converting digital signals to analog or audio tones and vice versa

**Modulate** The conversion of bits on the originating end of a transmission to tones for transmitting over telephone lines; see also *Demodulate*

**Monitor** A cathode ray tube used to display information for the computer; also refers to a unit of the computer or its software that keeps track of other activities for the system

**Mother board** A large printed circuit board that contains the main components of a personal computer system. Normally contains memory, the microprocessor, and slots for additional printed circuit cards, such as input/output interfaces

**Mouse** A small device connected to the computer by a cord and used to move the screen cursor (has buttons that act as function keys)

**MultiPlan** A programming package for creating spreadsheets

**Nested** Refers to the grouping of statements inside one another within a program

**Network** A communications pathway between computers or terminals

**Operating system** A set of programs that tells the computer how to perform all of its tasks, including the execution of other programs; must be the first set of programs entered into the machine.

**Output** The results of computer processing; see also *Input*

**Parameter** A quantity used in a command to specify a range or value

**Parity bit** An extra bit transmitted with the data over communications channels to check the accuracy of the transmission



## Appendix B

**Pascal** A popular higher level programming language noted for its simplicity and structured design

**Personal productivity programs** Programs that increase one's personal productivity—word processing programs, for instance

**Phantom disk drive** The use of a single disk drive as both the A and B ("phantom") drives by swapping two diskettes between operations

**Primitive** A built-in procedure

**Printed circuits** Laminated plastic boards containing integrated circuit chips and chip sockets and the connecting wiring

**Processing** A unit of the computer and the actual process of executing programs on a computer

**Programmer** A specialist who designs and writes computer programs

**Programs** The software or instructions that the computer follows in order to perform tasks

**Prompt** A signal issued by the computer to indicate that it is ready for a command or instruction

**Protocol** The features that allow data communications between computers via communications networks

**Pseudorandom number** A computer-generated random number (not totally randomized, but sufficient for most practical purposes)

**Public access message systems** Local telecommunications bulletin board systems that allow the general public to post or receive messages free of charge

**Public domain software** Programs that are uncopyrighted and are available free of charge for general use

**RAM** Acronym for *Random Access Memory*, direct storage memory (the fastest type available); see also ROM

**Range** Any contiguous group of cells (in a spreadsheet) or statements (in a BASIC program loop)

**Record** The basic unit of information in a file

**Recursion** In a program, the repetition of the same basic function over and over, often with a slight variation with each repetition

**Refresh rate** The frequency with which data on the video monitor is renewed

**Resolution** The number of dots that make up the letters or graphics on the video screen—the more dots, the higher the resolution and the clearer the image

**ROM** Acronym for *Read Only Memory*, a storage medium that can be retrieved but cannot be written on or changed; see also **RAM**

**Screen editor** A program to create and modify files, in which any position on the screen can be reached by a cursor control movement; see also *Line editor*

**Seed** An outside event that alters the sequence of computer-generated random numbers (to reduce the likelihood that they will come up in predictable patterns)

**Soft copy** A temporary image displayed on a video monitor, as opposed to a permanent copy of the data or output; see also *Hard copy*

**Software** The programs and documentation used by the computer, as opposed to the physical equipment; see also *Hardware*

**Source** The original or master diskette used to create copies or for comparison purposes; see also *Target*

**Spreadsheet** A program that accepts input and formulas into cells in a matrix and automatically calculates output values

**Start bit** The first element in a serial transmission, the signal that a new character is being transmitted; see also *Stop bit*

**Statement** A computer instruction normally composed of a line number, keyword, and list of variables

**Stop bit** The last bit in a serial transmission, the signal that the entire character has been transmitted; see also *Start bit*

**String** A contiguous group of characters arranged in a sequence

**Subroutine** A set of statements that can be used repeatedly to perform a specific task within a program

**Surge protector** A device that filters the current transmitted through a power line to the computer to prevent a power surge or voltage spike from damaging the computer or destroying its data

**Systems unit** The part of the computer where processing is performed

**Target** The diskette on which a copy is to be made; see also *Source*

**Telecommunications** The use of communications networks to transfer data between terminals and computers

**Template** A term for the original line in line editing

**Trip record** The last record in a file, specially marked with a series of four Xs (XXXX) to designate that record processing is ready to be terminated

**Turtle** An on-screen symbol used in the LOGO language to guide one around the screen and to facilitate graphics programming

**Appendix B**

**Up-line loading** The transfer of data and programs from a local computer or intelligent terminal to a remote location; see also *Down-line loading*

**Variable** A named quantity in a program that can assume different values during the execution of a program; see also *Independent variable*

**Video display or video monitor** The cathode ray tube used to display data and graphics output for the computer

**VisiCalc** A programming package for creating spreadsheets

**Wild card** A symbol used to indicate any value—for instance the asterisk and question mark used in DOS file manipulations

**Window** A portion of the screen allocated to show a different part of the output simultaneously with the main display

**Word processing** The use of the computer and special programs to prepare text and documents

**Write/protect notch** A small square hole in the corner of a diskette, which, when covered, makes the diskette incapable of receiving new information

## Index

- Accounting applications, 10, 158, 205-28  
See also Spreadsheets
- American Standard Code for Information Interchange (ASCII), 341, 346
- Arithmetic functions  
in BASIC, 270-72, 317  
in LOGO, 328-32  
with pfs:REPORT, 261, 262
- Arrays  
in BASIC, 289-91  
defined, 289
- ASCII (American Standard Code for Information Interchange), 341, 346
- BASIC (Beginner's All-Purpose Symbolic Instruction Code)  
arithmetic functions in, 270-272, 317  
arrays in, 289-91  
Cartridge and Disk, 267  
Cassette, 68, 91  
command execution in, 78, 79, 80  
commands used in, 75, 79-86, 316-17  
compared with LOGO, 318  
cursor movement in, 277  
data pointer in, 387-88  
description of, 67-68  
error handling in, 297  
features of, 80-81  
file types in, 292  
formulas in, 271-72  
functions, list of, 317-18  
games in, 304-12. See also Games  
graphics in, 304-12. See also Graphics  
inputting data from files in, 293-96
- BASIC (continued)  
inserting comments and remarks in, 286  
LOCATE statement in, 277-78  
loops in, 281-84  
nested loops in, 283-84  
opening and closing files in, 292-93  
outputting data to files in, 293-96  
playing tunes with, 311  
popularity of, 266  
PRINT statement in, 274-77  
PRINT USING statement in, 278-81  
printing in, 274-81  
random file programs in, 299-302  
random number generation in, 306-7  
relational operators in, 272-73  
statements used in, 79-86, 313-16  
storing and using information in, 286-89  
string and character functions in, 307, 317-18  
subroutines in, 284-86, 297  
supplementary programs in, 311-12  
trip records in, 287, 295  
use of sound with, 311  
variables in, 269-70, 274-75
- Baud, 344, 346
- Bit, defined, 21
- Board, defined, 20
- Boot, cold and warm, 99-100
- BRS/After Dark, 362
- Budget preparation and balancing, 10, 208
- Bulletin boards, computer, 362-63
- Business applications. See Data
- Business applications (continued)  
base management; pfs:FILE; pfs:REPORT Spreadsheets; Word processing
- Byte, defined, 23
- Cables, how to handle, 37
- Calorie counting, via spreadsheets, 209
- Card(s)  
for additional functions, 20, 24  
clock/calendar, 367-68  
defined, 20
- Carrying case, 58
- Cartridges, as storage devices, 33, 34, 75, 76. See also Cassettes; Diskette(s)
- Cassettes, 33, 37, 91. See also Cartridges; Diskette(s)
- Cell(s)  
defined, 206-7  
locating, with cursor, 214-16  
types of data contained in, 207
- Checkbook balancing, 10, 205
- Clock/calendar card, 367-68
- Clubs, computer, 169
- COMM, 355, 359
- Commands  
defined, 78  
external versus internal, 117  
imbedded, 195-96  
indirect, 78
- Communication. See Telecommunications
- Compatibility between computers, 94
- CompuServe, 360-61
- Computer clubs, 169
- Computer fairs, 169
- Computers  
functions of, 3-8, 15-33. See also Input devices; Memory; Output devices; Processing

- Computers (*continued*)  
 history of, 1  
 home/personal/micro-, defined, 3  
 limitations of, 1  
*See also* Personal computers
- Consumer services, available through information services, 359
- CROSSTALK, 355
- Cursor  
 description of, 19–20, 69  
 function of, 69–73  
 home position of, 73  
 and insert mode, 72
- Cursor control character, defined, 84–85
- Data, storage devices for, 33–37.  
*See also* Cartridges; Cassettes; Diskette(s)
- Data base(s)  
 defined, 231  
 how data is stored in, 231–32  
 on-line, 363  
 using, 232, 233  
*See also* pfs:FILE; pfs:REPORT
- Data base management, 231–63  
 advantages of, 158–61  
 applications, 206  
 description of, 158–62  
 designing a file in, 233–43  
*See also* pfs:FILE; pfs:REPORT
- Data communications. *See* Telecommunications
- Dialog, 362
- Dictionary, for spelling verification, 200–201
- Directories, diskette, 100–102
- Disk drive(s)  
 advantages of a second, 113–14  
 cleaning, 58  
 future options in, 367  
 how to operate, 104  
 “phantom,” 114–15
- Diskette(s)  
 advantages of, 35–36  
 backup copies of, 108–9  
 copying a, 108–13  
 description of, 56  
 directory of the, 100–102  
 formatting, 103–6, 115–17, 125–27  
 handling, 106–7  
 source versus target, 109  
 as storage devices, 33, 34–36
- Diskette(s) (*continued*)  
 storing, 53, 56  
 verifying a copied, 111–13  
 write/protect notch on, 107–8  
*See also* Cartridges; Cassettes
- DOS (Disk Operating System)  
 commands used in, 97  
 conventions, 129–31  
 description of, 94–95  
 functions of, 95–96
- Dow Jones, 355, 361–62
- Duplex, full (FDX), 344–46
- Duplex, half (HDX), 345–46
- Dust covers, 57
- EasyWriter  
 additional commands menu, 190–91, 195, 197  
 aligning text in, 190  
 backup copy of, 178  
 block operations in, 186–89  
 command keys, 177  
 compatibility with other computers, 175  
 deleting text in, 183, 184, 186  
 editing a file in, 180, 185, 186, 189  
 error correction with, 183–85  
 features of, 175–78  
 file management with, 177–78  
 file system menu, 178–80, 182  
 help menu, 181–82  
 imbedded commands in, 195–96  
 inserting and aligning text in, 189–95  
 paragraph symbol in, 176  
 printing with, 196–200  
 saving a file in, 182, 186  
 screen format in, 175–76  
 search and replace command menu, 192–95  
 transferring operating system onto, 178  
 “word wrap” feature in, 176  
*See also* Editor(s); Word processing
- Editor(s)  
 defined, 136  
 EDLIN, 141–49  
 line, 136  
 page, 157  
 screen, 175  
*See also* EasyWriter; EDLIN; Word processing
- EDLIN  
 commands used with, 137–49  
 how to use, 136–49  
 introduction to, 136–37
- Education, computer-assisted, 10, 328, 369
- Electrical power  
 and fire protection, 58  
 and power surge protection, 45, 48–49, 66  
 requirements, 47  
 and use of power bar, 48
- Encyclopedic services, 362
- Energy management, via spreadsheets, 209
- Entertainment. *See* Games (computer)
- Environment, optimum work, 45–47, 59
- Equipment checklist, 60–61
- Fairs, computer, 169
- Field(s)  
 defined, 119  
 setting up length of, 119–20
- File(s)  
 accessing, 130  
 in BASIC, 291–97  
 comparing, 131–32  
 copying, 128–31  
 creating, 136–37  
 defined, 119  
 deleting, 132–34  
 displaying contents of, 135  
 EasyWriter, 177–78  
 editing, 136–37, 141–44  
 names and use of “wild cards,” 123  
 naming, 121–25  
 opening and closing, 292–93  
 random, 292, 299–302  
 renaming, 134  
 saving, 140  
 sequential, 292, 293–97, 299  
 uses of, 121  
 verifying copies of, 131  
*See also* pfs:FILE
- Finances, personal, management of, 208–9
- Fire, protection against, 58
- Formatting  
 defined, 103  
 of diskettes, 103–6  
 verifying diskette, 115–17
- Formulas, in BASIC, 271–72
- Full duplex (FDX), 344–46

- Function keys, learning to use, 73-74
- Furniture layout and requirements, 50-55. *See also* Workspace
- Games (computer)  
 in BASIC, 304-12  
 "Crossfire," 75-77  
 drawing lines in, 306  
 educational, 154  
 generating random numbers for, 306-7  
 history of, 153-54  
 how to buy, 154-55  
 learning to use, 75-77  
 in LOGO, 328-32  
 sources of, 153, 359  
 string and character functions in, 307  
 use of color in, 304-6  
 use of sound in, 311  
 workspace layout for, 53-55
- Graphics  
 in BASIC, 304-12  
 future software for, 370  
 in LOGO, 319-28  
 upgrading system for, 20  
 use of color in, 304-6
- Half duplex (HDX), 345-46
- Hardware, 15-41  
 cables, 37  
 future, 367-68  
 joystick, 39-40, 76-77  
 keyboard, 2, 15-20, 17-18, 38, 68-74, 367  
 light pen, 41  
 modem, 41, 338-39, 342, 349, 355  
 monitor, 25-28  
 printer. *See* Printer(s)  
 system unit, 20-25  
*See also* Software
- Hazards  
 dust covers as protection against, 57  
 and insurance coverage, 55-56  
 power surges, 45, 48-49  
 protection of diskettes from, 106-7  
 static from carpet, 50
- Home computers. *See* Personal computers
- Home position, 73
- HomeWord, 174-75
- Information services, 355, 359-62
- Input devices, 3-5, 2-20, 41
- Installation  
 turning on the PCjr, 66-67  
 unpacking and assembling the PCjr, 59-63
- Insurance coverage, 55-56
- Interactive program, defined, 73
- Joystick, 39-40, 76-77
- Keyboard  
 cable connection for, 38  
 and cursor key functions, 69-73  
 defined, 2  
 description of keys on, 17-18  
 future version of, 367  
 infrared connection for, 38  
 as input device, 15-20  
 learning to use, 73-74  
*See also* Keys  
 "Keyboard Adventure," 73-74
- Keys  
 alphanumeric, 17  
 cursor control, 18  
 description of, 17-18  
 function, 18, 68-69, 73-74  
 insert and delete, 18  
*See also* Keyboard
- Keywords, BASIC, 87
- Kilobytes, description of, 24
- Languages, computer. *See* BASIC; LOGO; Pascal
- Letters, form, 173, 201
- Light pen, 41
- LOGO, 318-32  
 commands in, 321  
 compared with BASIC, 318  
 editing in, 325-26  
 games in, 328-32  
 graphics in, 319-28  
 recursive feature of, 318, 327-28  
 saving procedures in, 325-27  
 text manipulation features of, 328  
 use of the turtle in, 319-21  
 writing a quiz program in, 328-32
- Loops  
 in BASIC, 281-84
- Loops (*continued*)  
 defined, 281  
 nested, 283-84  
 use of, 90-91
- Lotus 1-2-3, 210-11
- Mailing List Manager, 201
- Mail-order software houses, 163-65
- Mail system, electronic, 347, 349-54, 359, 360
- Maintenance  
 preventive, 57-58  
 of printer, 200  
 of programs, 268-69
- Manuals  
 judging quality of, 167-68  
 operating, 62-63, 74  
 software, 167-68
- Memory  
 capacity of, 24  
 defined, 3  
 functions and kinds of, 23-24  
 future capacity of, 367  
 and power outage, 49
- Microchip, 3, 20, 23
- Microcomputers. *See* Personal computers
- Modem  
 described, 40-41, 338  
 external, 41, 338-39, 355  
 function of, 342  
 internal, 41, 338-39, 349  
 non-IBM, 355
- Monitor(s)  
 color, 27  
 description of, 25  
 monochrome, 27  
 TV set as, 28  
 types of, 25-28  
 and viewing comfort, 28
- Mouse, 369-70
- MultiPlan, 209-10
- News, available through information services, 359, 361
- Novation bulletin board service, 363
- Office layout. *See* Workspace
- Operating manuals, 62-63, 74
- Operating system (OS)  
 booting, 99-100

- Operating system (*continued*)  
 and compatibility, 94  
 copying function of, 108  
 defined, 93  
 external and internal  
   commands in, 117  
 learning to use, 97-100  
 need for, 93-94
- Output devices, 25-33  
 printer, 29-32  
 TV set, 28  
 video monitor, 25-28  
*See also* Monitor(s); Printer(s)
- Paper, purchasing, in bulk, 200
- Parity, 343-344, 346
- Pascal, 333
- Personal Communications  
 Manager (PCM), 349-54
- Personal computers  
 distinction between home/  
   microcomputers, 3  
 expandability of, 20  
 future enhancements for, 365-  
   70  
 hardware components of, 15-  
   42. *See also* Hardware  
 setting up your, 37  
 upgrading, 365-67  
 uses of, 10  
*See also* Computers
- Personal productivity programs,  
 future, 369
- pfs:FILE, 232-54  
 assigning filenames in, 236  
 cursor movement within, 236  
 deleting forms from, 253-54  
 designing a file in, 233-43  
 entering data in, 239-43  
 exact match in, 245, 249  
 main menu in, 233, 243  
 numeric matches in, 246-47,  
   249  
 partial match in, 245-46, 249  
 printing files from, 250-52  
 retrieving data from, 244-49  
 single disk drive limitations  
   with, 232-33  
 sorting in, 251-52
- pfs:REPORT, 254-63  
 generating reports with, 254-  
   62  
 headings and column widths  
   in, 257-58, 260, 262
- pfs: REPORT (*continued*)  
 performing calculations with,  
   261, 262  
 predefining a report in,  
   258-60, 262  
 report name in, 255-56  
 single disk drive limitations  
   with, 232-33  
 size limitations of, 256, 260,  
   262  
 sorting in, 256-57
- Phantom disk drive, 114-15
- Pixels, defined, 25
- Portability, with carrying case, 58
- Power, electrical. *See* Electrical  
 power
- Power bar, 48
- Power surges, 45, 48-49, 66
- Printer(s)  
 as aid in writing programs, 89  
 choosing, 29  
 dot-matrix, 29  
 future options in, 368  
 letter-quality, 29  
 need for, 32  
 paper for, 56-57, 200  
 parallel, 32-33, 252  
 re-inking ribbons for, 200  
 ribbons, 57  
 screen contents printed with, 136  
 selecting a, 33, 34  
 serial, 32-33, 256  
 terms, dictionary of, 198-99  
 thermal, 29-31  
 troubleshooting guide to, 199-  
   200  
 types of, 29  
 with BASIC, 274-77  
 with EasyWriter, 196-200  
 with pfs:FILE, 250-52  
 with pfs:REPORT, 256-57, 262
- Processing, 5-8, 20-25
- Programming  
 accuracy in, 283  
 as an art, 265-66  
 learning techniques for, 312  
*See also* BASIC; LOGO; Pascal;  
 Programs
- Programming languages, 265-333  
 commonality between, 267  
 for the PCjr, 266-67  
*See also* BASIC; LOGO; Pascal
- Program(s)  
 arithmetic, 78-86  
 backup copies of, 108-9  
 defined, 9
- Program(s) (*continued*)  
 documenting, 268  
 guessing-game, 86-88  
 interactive, 73  
 learning to run, 73-74  
 learning to write, 77-88  
 maintaining, 268-69  
 "personal productivity," 205-6  
 placing comments and remarks  
   in, 286  
 random file, 299-302  
 recursive, 327-28  
 statement numbering scheme  
   in, 79  
 storing and using information  
   within, 286-89  
 successful, 266, 268-69  
 testing, 268  
 writing versus buying, 152-53  
*See also* Data base  
 management;  
 Programming; Software;  
 Spreadsheets; Word  
 processing
- Prompt  
 in BASIC, 69  
 defined, 101  
 DOS, 101  
 EDLIN, 139  
 operating system, 101
- Public access systems, 362-63
- RAM (random access memory),  
 description of, 23
- Recipes, filing, 10, 160-61,  
 205-6
- Record, defined, 119
- Recursion, 318, 327-28
- Refresh rate, defined, 25
- Reports, generating, from files,  
 254-61. *See also* pfs:  
 REPORT
- Resolution, defined, 25
- Ribbons, 57, 200
- ROM (read only memory),  
 description of, 23-24
- Sales reports, 160, 205, 217, 228
- Setting up the PCjr, 59-63
- Shopping, electronic, 337-38,  
 359, 360
- Softcopy, 28

- Software  
 backup copies of, 108-9  
 "big three," 155-56  
 "borrowing," 165-66  
 components of, 9  
 cost of, 151-52  
 data base management. *See* Data base management  
 defined, 8-9, 151  
 documentation, 165-68  
 efficiency of, 168-69  
 future, 368-70  
 information sources about, 169  
 mouse with, 369-70  
 "personal productivity," 205-6  
 public domain, 166  
 purchasing, 152-55, 162-65, 202  
 quick reference card with, 168  
 selecting, 166-67  
 self-improvement programs in, 369  
 spreadsheet. *See* Spreadsheets  
 telecommunications, 339-40, 346-59. *See also* CROSSTALK; Personal Communications Manager (PCM); Telecommunications  
 for transferring programs between computers, 369  
 types of, 155-56  
 "user friendly," 168  
 word processing. *See* Word processing  
*See also* Hardware; Program(s)  
 Software terminal emulator, 355  
 Source, The, 355, 359-61  
 Spelling verification programs, 200-201
- Sports  
 analyzing, via spreadsheets, 209  
 available through information services, 359
- Spreadsheets, 205-28  
 description of, 158, 206-7  
 mathematical capabilities of, 228  
 programs available, 209-10  
 self-updating, 207-8  
 types of data in, 207  
 use of formulas in, 207, 219-22  
 use of labels in, 207, 217, 222  
 use of values in, 207, 218, 222
- Spreadsheets (*continued*)  
 uses for, 208-9  
*See also* VisiCalc
- Statements (program)  
 defined, 78  
 multiple, on one line, 89  
 numbering scheme for, 79  
 use of loop in, 90-91, 281-84  
 use of subroutine in, 284-86, 297  
*See also* Programming; Program(s)
- Stocks, information services and, 338, 361
- Subroutines, 284-86, 297
- Surge protector, 49
- System unit (SU), description of, 20-25
- Table, computer, 51. *See also* Workspace
- Tax returns, calculating, 205
- Telecommunications, 337-63  
 ASCII code with, 341, 346  
 asynchronous transmission in, 342  
 COMM, 355, 359  
 CompuServe, 360-61  
 cost of, 348, 359, 362  
 CROSSTALK, 347, 355-57  
 Dow Jones, 361-62  
 down-line/up-line loading in, 348  
 encyclopedic services available through, 362  
 error-detection system in, 342-44  
 free communications program for, 355, 359  
 full duplex (FDX) mode of operations in, 344-46  
 fundamentals of, 340-46  
 future techniques for, 337-40  
 half duplex (HDX) mode of operations in, 345-46  
 information services available through, 359-62  
 "intelligent terminals" and, 347-48  
 modem use in, 40-41, 338-39  
 Personal Communications Manager (PCM), 347, 349-54  
 public access systems available through, 362-63
- Telecommunications (*continued*)  
 setting up equipment for, 346  
 software for, 339-40, 346-59.  
*See also* CROSSTALK; Personal Communications Manager (PCM)  
 The Source, 359-60  
 start and stop bits in, 342, 346  
 transmission speed in, 344, 346  
 use of parity with, 343-44, 346
- Television set  
 damaged by computer video signals, 38  
 as video monitor, 28
- Terminal, "intelligent" versus "dumb," 347-48, 355
- Travel information, available through information services, 359
- Trip record, 287, 295
- Turtle, 319-21. *See also* LOGO
- Unpacking the PCjr, 62-63  
 Upgrading the PCjr, 365-66
- Variable(s)  
 in BASIC, 269-70, 274-75  
 data types contained in, 269-70  
 definition of, 269  
 learning to use, 83-86  
 in LOGO, 319  
 naming, 269  
 numeric, 269-70  
 string, 269-70
- Video monitor. *See* Monitor
- VisiCalc, 209-28  
 commands, 223-24, 227  
 cursor movement within, 214-16  
 editing in, 226-27  
 entry line in, 212-14  
 filenames in, 225  
 format changes in, 224, 227  
 print command in, 226  
 prompt line in, 213  
 saving a file in, 225  
 screen display in, 212-14  
 transferring operating system onto, 211-12  
 use of formulas in, 219-22  
 use of labels in, 217-18, 222  
 use of values in, 218, 222



VisiCalc (*continued*)

"what if" calculations in, 225–  
26, 227

*See also* Spreadsheets

## Word processing

advantages of, 137, 156, 171–73  
by "dedicated" word processors,  
156

defined, 156

and ease of typing, 171–72

EasyWriter, 175–98. *See also*

EasyWriter

Word processing (*continued*)

error correction in, 172

form letters in, 173, 201

future software for, 202, 369

HomeWord, 174–75

mailing lists in, 201

major text revisions with, 173

options available with, 200–  
201

programs compatible with the  
PCjr, 173–74

spelling verification with, 200–  
201

*See also* EasyWriter; Editor(s)

## Workspace, 45–55

children's needs in, 50

and electrical power  
requirements, 47–50

furniture needed for, 50–55

for games versus business, 53–  
55

layout of, 50–55

office supplies needed for, 56–  
59

protection from sunlight in, 46


Write/protect notch, 107–8

# The Only book you'll ever need!

Whether you're at home, at work or at school, you **can** succeed with the PCjr. This book is a complete guide to building confidence in your computer skills.

How do you make the PCjr a part of your home and make yourself feel at home with the PCjr? The authors cover **EVERYTHING**—from taking the PCjr out of its carton and setting it up to purchasing the right software. The book's jargon-free, understandable writing style completely de-mystifies the PCjr. **Buy this book and save yourself time, money and grief!**

## Featuring

- ✔ Checklists for quick and easy reference—look for the  to save you time & money
- ✔ Hands-on exercises to walk you through your first experiences with the PCjr
- ✔ A complete glossary of computer terms
- ✔ Original programs that not only explain the fundamentals of programming, but also provide hours of computing fun
- ✔ Over 100 illustrations
- ✔ Suggestions for choosing games and entertainment software
- ✔ Coverage of BASIC and Logo
- ✔ An introduction to useful software packages, including EasyWriter, VisiCalc, PFS File, and PFS Report

## THE COMPLETE GUIDE TO SUCCESS WITH THE IBM PCjr

Written **BY** a family **FOR** the family. **Vernon Sondak**, M.D., provides expert insight into the health and safety considerations of using computers. **Eileen Sondak**, professional writer, ensures the readability of the book. **Norman Sondak**, DEng, Chairman of the Department of Information Systems at San Diego State University, is the author of many successful computer books. He has introduced thousands of beginners to the world of computers. **The Sondaks' combined background and skills give them an expertise you will not find in other books.**

**Times Mirror/Mosby**

U.S. \$14.95 Canada \$16.95

ISBN 0-8016-4671-5