

The Commodore 64

\$24.95

HOME COMPANION



The
Commodore 64
Home Companion

The Commodore 64 Home Companion

by
George Beekman

**Illustrations by
Martin Cannon**



DATAMOSTTM
INC

20660 Nordhoff Street
Chatsworth, CA 91311
(818) 709-1202



ISBN 0-88190-294-2

**Copyright © 1984 by DATAMOST, Inc.
All Rights Reserved**

This Manual is published and copyrighted by DATAMOST, Inc. Copying, duplicating, selling or otherwise distributing this product is hereby expressly forbidden except by prior written consent of DATAMOST, Inc.

The word Commodore and the Commodore logo are registered trademarks of Commodore Business Machines.

Commodore Business Machines was not in any way involved in the writing or other preparation of this manual, nor were the facts presented here reviewed for accuracy by that company. Use of the term Commodore should not be construed to represent an endorsement, official or otherwise, by Commodore Business Machines.

Acknowledgements

I would like to thank several people for their involvement and support during the writing of this book.

Dave Gordon, Marcia Carrozzo, and the rest of the DATAMOST people, for the tremendous effort they put into making my manuscript into a book.

The folks at Commodore (especially Diane Ottinger) and all of the other hardware and software manufacturers and distributors whose cooperation made this book possible.

Will Brown, Larita Brown, Jon North, and Steve Lenz of Edu-Tech Computer Center, my favorite Corvallis Commodore consultants.

Members of the Corvallis/Albany Commodore 64 User's Group, who shared ideas with me.

Ted Lewis, Tony White, Steve Johnson, Barbara Gladstone, Neal Gladstone, Pat Kalvin, Jack Dymond, Jan Dymond, Peter Eberhart, John Swanson, Jim Folts, Steve Sakuri, Dennis Corliss, Fred Tong, Mary Jane Tong, Jay Nave, Kathryn Brooksforce, Rick Houghton, Audrey Perkins, Rosie Beekman, Ed Beekman, Larry Beekman, Jo Ann Heisner, Wally Heisner, Cheryl Robinson, Dorothy Hyde, and all of the others who listened to half-baked ideas, read half-finished chapters, and gave feedback and support.

Ben and Josie, who time and again helped me over rough spots with a smile or a hug.

Most of all, to Sue, my constant source of support, assistance, and inspiration, whose role in this project is much too big to fit on a thank you page.

George Beekman

Table of Contents

How To Read This Book	9
Introduction: On Inviting a Computer Into Your Home	13
Chapter 1 — Hardware Facts — Putting Your System Together	29
A Shopper's Guide To Technobabble	29
Planning A Computer System You Can Live With	34
The Nuts and Bolts of Buying	42
Choosing A Home For Your Computer	44
Putting It All Together	45
Chapter 2 — Getting Started: Bit by Bit	49
Let Your Fingers Do The Talking	50
Making Your Expensive Computer Into A Cheap Calculator ...	56
Poking Around In Memory	61
Chapter 3 — Getting Down To BASIC: A Programming Primer	67
Commands and Statements: The Basics of BASIC	68
Getting Around... And Around, And Around	72
Endless Graphics	76
Using Variables For Flexibility	78
The Amazing FOR/NEXT Loop	81
Loops Within Loops	86
Making Basic Decisions	89
Chapter 4 — A Peripheral User's Guide	97
A Sound Program For Saving	98
Saving And Recalling Programs On Tape	102
Saving And Recalling Programs On Disk	108
Keeping Your Disk In Order	114
Using the Printer: A Primer	126
Sounding Off!	128
Chapter 5 — Software Shopping, Swapping, and Scrounging	129
Intelligent Software Buying	130
Public Domain Programs: Software For Free	135
Almost Free Software: Books and Magazines	136

More BASIC: Subroutines, Functions, Arrays, Odds and Ends	138
More BASIC Input and Output	148
Recognizing Generic BASIC: A Translator's Guide	153

**Chapter 6 — The Marvelous Toy:
Recreational and Educational Software 159**

A Taxonomy of Computer Games	160
Games by Name: A Sampler	165
Automatic Learning: A Survey of Possibilities	179
Programs for the Very Young	180
The Turtle as Teacher: A Look at LOGO	193
Music and Art: Creativity Software	201

**Chapter 7 — The Marvelous Tool:
Working Software 207**

Typing Without Tears: Word Processing for Beginners	208
Calc Power: At Home With Spreadsheets	226
Using Your Computer as a Filing Cabinet	237
Taking Care of Business	243

**Chapter 8 — Creating Your Own Software:
When The Bug Bites 247**

The Art of Program Design	248
Making BASIC Better	252
Beyond BASIC	258
The Power of Pascal	260
The Coming of Comal	268
Assembly Language: Bit-Twiddler's Delight	269
The Tower of Babel: Forth, Pilot, and More	272

**Chapter 9 — The Imagination Machine:
Today and Tomorrow 279**

The Modem: A Window Into The Wired World	280
The Strange World of Information Utilities	283
More Timesharing and Networking	289
Giving Your 64 The Soul Of A New Machine	292
What's Next	299

Appendix A 311

For More Information	313
Periodicals For Keeping Up	314
Good Books	320
Addresses And Phone Numbers	324
Commodore User Groups	333

Index 351

How To Read This Book

“Everybody tells me that computers can simplify my life, but I’m skeptical. I want to know just two things: What can this machine do for me, and how can I make it do those things? If I have to suffer through a bunch of technobabble about programming and chips, forget it.”

“I want to use my computer. But I want to know more than which button to push. I want to know terms and concepts so that I can intelligently browse in computer stores or computer magazines. I want to be computer literate.”

“I’m fascinated by computers. I believe they’re the wave of the future, and I want to ride that wave. Sure, I want to play computer games, but I want to create my own computer games, too! I want to know not just how, but why.”

I wrote this book for my friends. When I asked them what they wanted from a computer book, those were the kinds of answers they gave me.

But each of these three people was asking for a different book. Which one should I write? The answer came when I realized that the typical family—the typical person, for that matter—is a mixture of all three of these attitudes.

This book, then, is three books in one. What you get out of it depends on how you read it. If you’re looking for a computer-as-home-appliance instruction manual, you can take the express lane through the essentials in two or three short sittings. If, while you’re learning to use your machine, you want to learn about computer programming, for practical reasons or fun, you can take a more leisurely path through the book. And if you’re one of the hard-core curious, there are plenty of scenic diversions along the way for you. (You can even read the book more than once, taking different paths through each time.)

Specifics: The introduction is an orientation to home computing and this book; it's quick reading, but you can skip it if you're in a big hurry. (If you aren't sure which path to take through the rest of the book, the introduction can help you to decide.) Chapter 1 talks about buying, putting together, and taking care of your computer system. Take what you need, and leave the rest.

As much as possible, this book has been written so that you can pick and choose the material that you want to read. But there is some "core" material in Chapters 2 through 5 that's essential for understanding later parts of the book. That material will vary depending on which of the three levels of understanding you're after. You'll find boxes like this one in those chapters to guide you to the things you need.

Most of the material in those four chapters is designed to be "read" in front of a computer. Don't try to memorize everything, just play with your machine. You'll learn most of the important things automatically in the process. And in case you forget something, there's a tear-out-and-slide-under-the-computer crib sheet tucked away in the back. Move at your own pace, and take breaks when you feel like it.

The last four chapters are mostly about specific programs that you can use to make your computer do the things you want it to; they are designed to be read (or not read) in any order. Those chapters don't cover every program ever written for this computer. I've included what I consider to be the best home-oriented programs for the C-64, based on usefulness, price, performance, popularity, and uniqueness.

No computer book can tell you everything, so Appendix A shows you where to look for information that you couldn't find in this one.

This book is for you. If it doesn't meet your needs, or if it does, I'd like to know about it. If you have suggestions or ideas for future editions of this and other books in the *Home Companion* series, please let me know. Thanks. Enjoy your trip!

George Beekman
c/o DATAMOST, Inc.
20660 Nordhoff Street
Chatsworth, CA 91311

Introduction: On Inviting A Computer Into Your Home

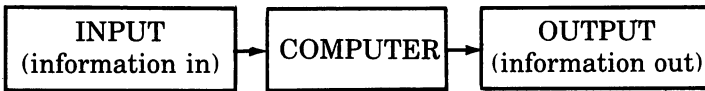
"The trouble with machinery is that charm doesn't work on it."

—Hawkeye Pierce, *M*A*S*H* 4077

Thirty years ago Hawkeye Pierce would have scoffed at the idea of living with a home computer. Today, Alan Alda sells them. The computer is coming home.

WADIZZIT?

A computer is just a machine that changes information from one form to another. Because of the many forms that information can take, the computer is an incredibly versatile tool, capable of everything from computing federal income taxes to piloting the missiles that those taxes buy. Think of a computer like this:



If the computer is used for calculating taxes, the input might be numbers representing wages, salaries, tips, other income, deductions, exemptions, write-offs, and tax tables, and the output might be the number representing the taxes owed. If it's used for controlling a missile, the inputs might be radio and radar signals locating the missile and the target, and the output might be electrical signals to control the flight path of the missile.

In either case, the computer goes through a series of steps or instructions called a *PROGRAM*, to transform the input *DATA* (information in a form it can read) into the necessary output. In the same way, your washing machine goes through a series of steps (wash, extract, rinse, spin) to transform its inputs (dirty clothes, detergent, clean water) into the desired outputs (clean clothes, dirty water). You probably have a certain amount of control over the washing machine's program: you may choose the "delicate" program, the "permanent press" program, or the "regular wash" program. But you certainly can't get your washing machine to wash and wax your floors, because the floor won't work as an input or output, and even if it could you have no way of adding a "floor" program to the washer's repertoire. The thing that makes a computer so special is that it is a general purpose machine: (1) it can be connected to a variety of input and output devices, and (2) it can be programmed to perform countless tasks.

Computers have been around in one form or another since World War II, and the idea goes back at least 100 years before that. But until the space race of the sixties, computers were big, expensive, and finicky. Only the largest business or government institutions could afford to buy a computer, not to mention the large climate-controlled computer center needed to house it and the staff of technicians needed to keep it running.

The USSR changed all that by launching Sputnik in 1957. The American government responded with a single-minded commitment to conquer space, and technology here on Earth took giant leaps forward as a result. Scientists and engineers reduced the size of computers as they increased their reliability, until in 1971 a complete computer was housed in a silicon chip

smaller than an astronaut's nose! That first microcomputer on a chip cost American taxpayers a bundle for research and development. But once the machinery was in place for that one, microcomputers could be mass-produced for little more than the cost of the silicon they were stamped out of. Which wasn't much, since silicon is the second most common element (behind oxygen) in the Earth's crust. (It's the main ingredient in beach sand, among other things.)

American entrepreneurs, who smelled gold in those silicon wafers, soon flooded the marketplace with digital watches and pocket calculators. (These devices actually contain tiny microcomputer chips that have been permanently wired to perform their specific tasks.) But the real consumer computer revolution occurred in the mid-seventies with the birth of the personal computer industry. For the first time, an individual could buy (for the cost of a good stereo) a complete general-purpose computer. These first computers were bought mostly by hobbyists who knew how to handle soldering guns and schematic diagrams, but as the computers got better and cheaper, the general public showed an interest.

Today it seems like everybody is buying (or thinking about buying) a home computer. But why would an otherwise sane person plunk down his or her salary for a computer?

WHAT GOOD IS IT?

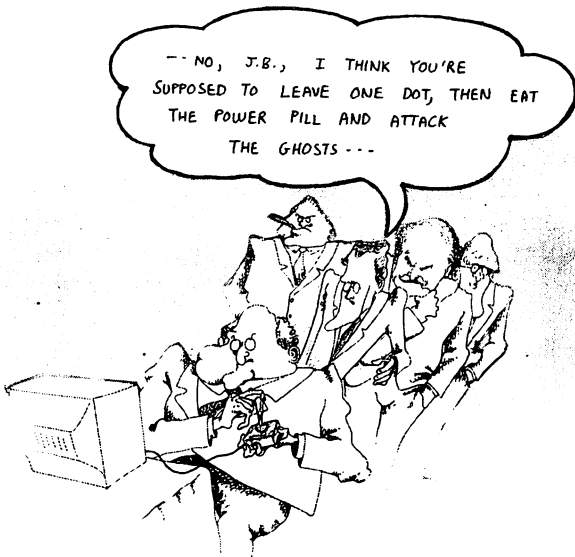
Most everything we buy can be considered either a tool or a toy—either it works for us or we play with it. The tools we own, at least in theory, make our work go faster and easier so we have more time for other things. Our toys help us to pleasantly while away that extra time. Some of our possessions are hard to classify in this simple dichotomy: bicycles and educational games, for example, are both toys and tools.

Home computers fall squarely into both categories. It's completely up to the owner of a computer whether he chooses to make time with it, take time with it, or both. Everybody's needs are unique, and the choices are nearly limitless. Here are some examples.

Computer as Entertainer. Everybody knows Pac-Man, the highest paid entertainer in recent years. He's the most widely-known symbol of our culture's mania for computer-controlled video games. The game industry now makes more money in America than the movie and the record industries put together! However you feel about the games, they are among us.

With the advent of the home video game machine, the focus is shifting from the video arcade to the family room, transforming the TV set into an audience-participation machine. Most home video game machines are poor imitations of their professional cousins, but a good home computer can be an impressive game machine, much closer to the quality of the (\$2,000.) coin-operated games.

So far, most home video games have been patterned after their arcade ancestors, which are designed to maximize the coin flow, and are still based on the three-balls-for-a-quarter model of their pinball parents. But a new generation of home games is emerging that promises much more than fast action and macho consciousness. These games, designed for many different age groups, make Space Invaders look primitive.



When TV was a new medium, most of the programming was little more than video vaudeville and radio programming with faces; it took a few years for the industry to develop its own identity. The home video game is still in its infancy as an art form—the next few years should prove very interesting.

Maybe you prefer more traditional pastimes like board games, card games, or logic and memory games. Your home computer can (with the proper programs) serve as a worthy opponent when you can't find a human to satisfy your itch for a game of chess, Monopoly, Othello, blackjack, poker, or whatever. Of course, it won't be much of a conversationalist between moves, and it won't fix the popcorn for you. But then again, it (probably) won't try to cheat, either.

There are other computer gaming possibilities, too. For a more detailed survey see Chapter 6.

Computer as Tutor. Computer Assisted Instruction (CAI) has been around since the sixties, when programs were first written with the expressed purpose of teaching. Most of these CAI programs involved drill and practice of some set of facts or skills, like arithmetic. Some were clever, others were basically boring. But the real problem with these early programs was that only the wealthiest school districts and the military could afford to use them. Personal computers have changed all that. Your computer has the capability of doing everything those early CAI programs did and more.

Why use a computer to duplicate what teachers do? The fact is that teachers can't possibly do the job of educating our children alone. Shrinking budgets and increasing class loads make individual and small-group tutoring nearly impossible, so that the student who needs extra help (or an extra challenge) is often out of luck. A computer can fill the gap by providing instruction geared toward those individual students' needs. The student can proceed at his or her own pace, because the machine never loses patience.

Students learn better and faster if they get quick and accurate feedback on their progress. Quick and accurate are two things computers do best. Then there's the question of motivation. Ask your kid whether he'd rather learn math (or grammar, or vocabulary) from lectures and worksheets or from a computer. Or visit a classroom where kids learn with computers. They're having fun!

But children—and adults, for that matter—can learn a lot more than rote skills like arithmetic and French verb conjugation from a good home computer. They can learn to play and compose music using the computer as a music synthesizer. They can learn how color, form, and motion interact in the visual art of computer graphics. They can learn how to make complex decisions using programs that simulate real-world problem settings such as a stock exchange or an airplane cockpit. Or they can learn how to program computers!

There are reasons for learning computer programming, even if your kid (or yourself) has no intention of becoming a programmer. Learning to program is a concrete way of learning to solve problems. Learning to program is learning to think in creative ways. Learning to program is learning to accept your own mistakes, and to feel good about your accomplishments. Learning to program is, in the eyes of many, learning to be comfortable with the future.

Your child may already have access to computers through her school. Chances are that access is limited, though. Your home computer can remove the time pressure and allow the child to explore at her own pace. It will also allow her to use the computer in ways besides those your school curriculum dictates.

If your child doesn't have access to computers at school, then it's even more important that she have it at home. Not just for use as a high-powered calculator or as a term-paper generator; not just as a path to computer literacy; but as a form of insurance against mediocre teachers, inadequate learning materials, and other possible shortcomings of her school.

Besides, computers can help make education fun, like it should be. That's the best way to ensure that your child will continue to get the best education possible.

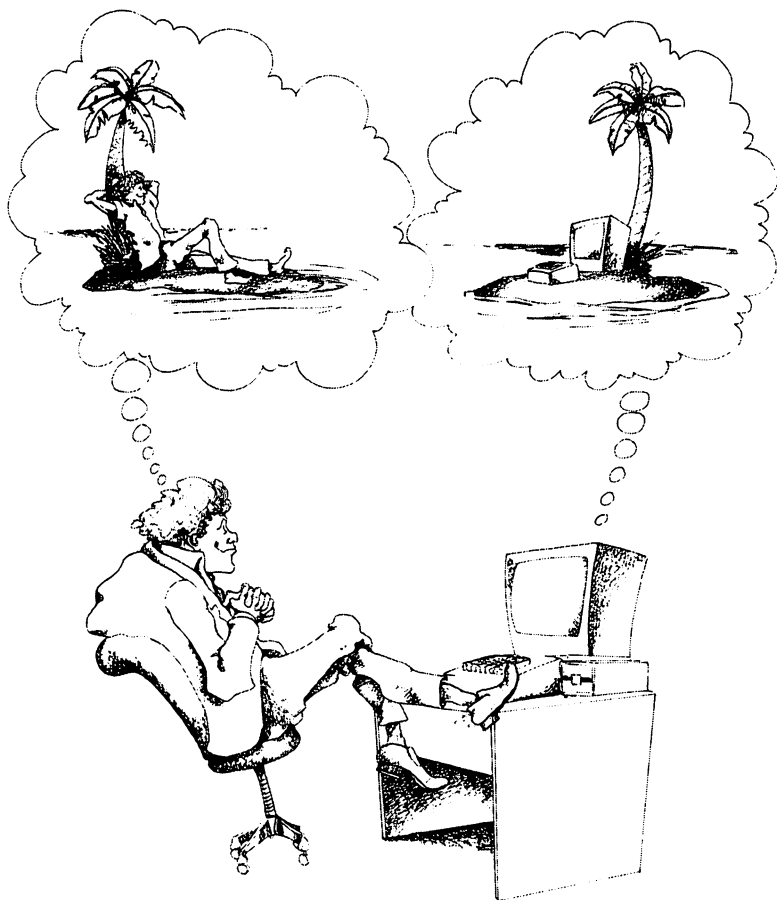
For information on educational programs available for the Commodore 64 (many for free), see Chapter 6. That chapter also introduces LOGO, a powerful computer programming language designed especially for children. Chapters 2, 3, 4, 5, and 8 talk about beginning programming in the BASIC computer language. Chapter 8 also describes several other computer languages, including Pascal, a language that many educators and computer scientists prefer over BASIC, and Pilot, a language designed especially for making your own educational programs. Appendix B has a couple of educational game programs that are ready to type in and run on your computer.

Computer as Clerk/Consultant. Frank Gilbreth, a turn-of-the-century pioneer of motion study in industry, applied "scientific management" techniques to his home. He required that each of his 12 children keep records on bathroom "work-and-process charts" of each hair-combing, tooth-brushing, bed-making, and bathing. He gave them demonstrations on efficient bathing techniques to minimize "unavoidable delays."

While it may have worked for Gilbreth, this "scientific management" approach to home life is not likely to catch on in a big way in today's world. Still, there are certain aspects of family life that are unavoidably businesslike: keeping records for the taxman, balancing the checkbook, figuring out how to pay for Osgood's college education without selling the camper, and so on. Most of us dislike these household management jobs because we find them boring, repetitious, and unimaginative. Unfortunately, we can't wish those tasks away, so we postpone them, argue about them, and fantasize about desert islands with palm trees, hot and cold running natives, and no money hassles.

Enter the home computer, which never complains about repetitious, unimaginative and coldly calculating tasks. In fact,

those are just the kind of tasks it does best. This is not to say that your home computer can eliminate procrastination, money arguments, and fantasies about the South Seas from your life (even if you wanted to), but there are some specific ways that home business matters can be made a little more pleasant.



- **BALANCING THE CHECKBOOK.** Until the day comes when you can hook-up your computer directly to the bank's computer (it's already possible for a few), you'll need to type a bunch of numbers to balance your checkbook by computer. (If you're a perfectionist, you'll need to enter some data every time you

write a check.) But the computer can do the organizing and calculating for you, so your mind can be off on your island or wherever else you want it to be. It can, with the right program, keep records for a household budget or tax time. And best of all, your computer won't botch the calculations.

- **MAINTAINING YOUR LITTLE BLACK BOOK.** Why computerize a simple thing like a personal address book? So that you can easily make copies of part or all of the list. (How about a list of relatives to call about the family reunion? Or a list of members of your pizza-frisbee team?) Or so that you can easily and legibly make changes in the list. ("Is this little squiggle a five or a piece of last night's dinner?") Or so that you can quickly generate pre-addressed form letters and mailing labels ("Dear Harry, I just wanted to remind you that this Friday at 8:00 is the next meeting of the Crook County Citizens Task Force for Straighter Trees").

Of course, the computer can be used to store other kinds of information besides names and addresses. A good **DATABASE MANAGEMENT PROGRAM** can make your computer into an electronic filing cabinet.

- **FINANCIAL DECISION-MAKING.** "Can we afford a trip to Mexico this year?" "Should we rent or buy a vacation home?" "Should we pay off the loan early or invest the money in hog futures?" These kinds of questions are never easy to answer, but a home computer with the right program can help.

A Spreadsheet program can be particularly helpful for answering these "what-if" questions. It can also be used for balancing checkbooks, bookkeeping, counting calories, and countless other applications.

- **TYPING.** If you spend more than an hour a week typing, you'll definitely want to use your computer as a word processor. (If you spend less time than that, maybe you're avoiding important tasks because you hate to type. If so, you need a word processor, too.) A word processor is basically a typewriter with

brains. It allows you to quickly change words or move paragraphs of a document without starting over. It allows you to refine your letter, term paper, or recipe on the screen before you produce one or more perfect paper copies. Some say that word processors are to typewriters what typewriters are to pencils. (I say that this book never would have been written without one!)

Chapter 7 has more on word processors, spreadsheets, database managers, and other useful programs for the Commodore 64.

Computer as a Business Partner. If you have a small business of any kind—rentals, retailing, or writing—you should be able to find dozens of business-related uses for your computer. Computer programs exist to do everything from inventory to accounts receivable.

Maybe you don't have a business, but you've always wanted to start one. You don't need to look past your home computer to find dozens of possible business opportunities. The economy may be jittery, but the personal computer industry is exploding right now. It's moving way too fast for the conglomerates to monopolize the action. Every day little guys and gals are making big profits by writing original programs, designing useful computer accessories, and doing other clever things with little computers. So, if self-sufficiency is your goal, your personal computer may be the first step.

This book can be your first step toward getting to know your home computer. Appendix A suggests further steps.

Computer as Communicator. Every application we've discussed so far takes place in the privacy of your own home. But by hooking up your computer to the outside, via telephone lines, you can open up a whole new set of possibilities. You can have instant access to any one of hundreds of *information utilities*, so that your computer becomes a massive library of knowledge. You can send electronic messages to any other person in the world with the same or similar equipment, or participate in computerized cross-country conferences or cocktail parties. You can use your computer to talk to or program large computers. You can bank or shop using your computer rather than your car.

If your fancy is sufficiently tickled by these possibilities, you'll want to look at Chapter 9.

Computer as a Slave. Maybe I haven't mentioned your special computer fantasy yet. Maybe you don't give a datum about video games or balanced budgets. What you really would like to do with a computer is _____. Don't despair. Maybe somebody has written a program to do just what you want. If not, maybe you can write it yourself! Computer programming isn't all that hard, and it can be lots of fun if you approach it with the right attitude. Once you know how to program, you can go a long way toward customizing your computer to do exactly what you want it to do!

If you're curious, look into Chapters 2 through 5 and 8. If you're terrified, relax. There's probably a program somewhere to do your bidding, and a *user group*, (a club of computer users, what did you think it was?) might just be able to help you find it (or write it). Chapter 7 and Appendix A have more on user groups.

If you aren't afraid of soldering guns and terms like RS-232, you can do more than just write programs for your computer—you can, in effect, give it new sense organs. Your computer can be thought of as an electronic controlling device. It can, with the right attachments, control or monitor (keep track of) any other device that can be hooked up to electricity: your burglar alarms, your stereo system, your coffee maker, your solar collector, even your wood stove. If your needs are esoteric enough, you may have to do at least part of the hookup yourself, or get somebody else to do it for you. But if you're adventurous and clever, there are plenty of possibilities for exploration. It's up to you to define your own limits.

Chapter 9, and some of the resources in Appendix A, might help you along the way.

PITFALLS

There are lots of exciting possibilities to explore with your computer. Think of this book as an explorer's guide, if you like. One of the duties of a guide is to point out potential hazards, and to provide information on how to deal with them. The world of personal computing is not filled with man-eating sharks and poison ivy, but there are some aspects of computer ownership that bother some people, and it's only fair that I warn you so you don't stumble blindly into them. (If you're the kind of person who never reads "Don't Feed the Bears" pamphlets, you probably should move on.)

Computer Frustrations. As machines go, computers are relatively friendly. You don't have to get your hands greasy to use them, and I know of no recorded case of injury from cleaning a loaded computer. But if you identify with Hawkeye, remember: a computer is a machine. Furthermore, it's a relatively new machine. Home computers aren't too much past the

“get out of the buggy and crank it up” stage. A home computer can make your life easier and more rewarding in many ways, but it will also bring moments of total frustration. On those days when everything seems to go wrong, it all seems to go “wronger” with a computer. Fortunately, those days do pass.

Computer Dependency. Home computers don’t require 5,000-mile checkups, they never need shots from the vet, and they seldom require the services of a plumber. All in all, the home computer is an extremely reliable item.

But all machines, biological, mechanical or electronic, break down from time to time. So let’s suppose you have your family appointment calendar stored in your computer; you turn it on to find out about today’s appointments, jobs, and birthdays, and can’t get the computer to tell you anything except ERROR FILE NOT FOUND. What do you do?

Businesses have been trying to deal with that type of question for years, and the only real answers they’ve come up with are (1) buy reliable equipment; (2) take precautions against failure (take care of the equipment); and (3) keep a backup of your records in case the system fails. In the supermarket, this means making sure customers can check out even when the computer is down. In the home, it means keeping duplicate copies of important pieces of information, and making sure you aren’t completely dependent on the computer for really important things.

Computer Addiction. A new breed of college student has evolved on college campuses over the last couple of decades: the hacker. The classic hacker spends all of his waking hours (which are mostly after dark) hunched over keyboards in the sterile halls of computer centers, munching on vending-machine food while tinkering with programs or typing messages to compatriots sitting at terminals on the other side of the campus or the other side of the room. He is, plain and simple, a computer addict.

The hacker phenomenon shows that computers, like almost anything else, can be addictive to certain types of people looking for escape routes. As addictions go, computer addiction is

usually relatively benign. (In fact, many of those early computer junkies are now very rich, powerful, and happy as a result of their hacking.) So maybe the best way to deal with it is to accept it at a reasonable level. One counselor who knew all too well that her husband was a potential hacker agreed to buy a home computer on the condition that the TV go. She decided being a computer widow was preferable to being a football widow. He's probably better off, too.

Computer Disillusionment. Computers are getting a lot of hype these days. And with good reason. They can do amazing things, many of which our ancestors never dreamed possible. But one effect of all this hype is that new computer owners have expectations that can't possibly be fulfilled by their machines.

Even the best home computers have limitations. Some of those limitations result from the fact that home computers are fairly small in memory capacity. Others have to do with the newness of the home computer industry. Still others are due to the fact that some problems don't lend themselves to computer solution at all.

Most books on home computers include a list of wonderful things you can do with your computer. And most of those lists include something like "use your computer as a recipe file." Most of those happy little books fail to mention that somebody has to type all of your recipes into the computer before the computer can give them back. They also forget to point out that keeping your cookbooks in the computer isn't very practical unless you use a printer or keep your computer in the kitchen, where it's susceptible to all kinds of natural and not-so-natural disasters.

Now there are probably lots of folks out there who get great satisfaction from using their computers to organize their recipe collections, and that's good. But for most of us, it would probably be more of a hassle than a help. (At least until the day one can buy quality cookbooks in a computer-readable form.) The point

is that sometimes a computer doesn't make a job any easier; it just changes the nature of the work. 🐾

Sometimes using a computer to do a job means even more work, but a higher quality end-product. For example, if you're willing to take the time and energy to type your recipe collection into a good DATABASE MANAGEMENT PROGRAM (Chapter 7), you'll be able to ask your computer questions like, "What can I cook that uses broccoli and eggplant?" and get back a complete list of possibilities in minutes!

In general, you'll stay out of trouble when computerizing some aspect of your life if you can answer to your own satisfaction each of these questions before you invest money and time: (1) What exactly do I want to do? (2) Can the computer do it for me? (3) Is there a better way?

Computer Unknowns. No matter how many books are written about the wonderfulness of the home computer revolution, the fact remains that nobody really knows the affects a home computer can have on a family over the long haul. Home computers are just too new. Who could have predicted in 1950 the profound effects—both positive and negative—that television would have on our world?

We're always making blind choices, as individuals and as a society. The decision to make a home computer a part of your life is ultimately a blind choice. Using a home computer today is a little like driving a horseless carriage way back then; it requires a certain sense of adventure. So if you're ready, let's go!

🐾The same can be said of the automobile. Researchers have determined that, when you take into account the time we spend working to pay for our cars, fuel, highways, etc., we could get around faster by walking.

CHAPTER 1

Hardware Facts—Putting Your System Together

“What’s one and one and one and one and one and one and one and one and one and one and one?”

“I don’t know,” said Alice. “I lost count.”

“She can’t do addition,” said the Red Queen.

—from Lewis Carroll’s *Through the Looking Glass*

Whether you own a computer or you’re on the verge, chances are you’ve got more shopping to do to make the most of your machine. If you don’t know what you’re doing, the process of planning and buying a computer system can be harder than using the system. This chapter will help you put together your personal computer system gracefully and painlessly. We’ll start with a short course in computer anatomy, complete with buzzword names for the body parts so you can be in the know if you want to. (Ignore them if you don’t; jargon isn’t important.) Then we’ll discuss the process of buying and financing your system. Finally, a few words on what happens after you get it home.

A SHOPPER’S GUIDE TO TECHNOBABLE

Fortunately, you don’t need to know how a computer works to buy one. But computer shopping can be easier, less intimidating, and more interesting if you know a little bit about the major pieces of a computer system and how they fit together. So we’ll take a brief tour and be back before the sun sinks over the CPU.

Glad you asked! CPU stands for Central Processing Unit, the name given to the computer’s “brain.” 🐾

Every computer has a CPU to interpret and carry out the instructions in each program, to do arithmetic and logical data

🐾 In case you hadn’t noticed, computer people have a fondness for acronyms like this; if you don’t share that passion, you’ll just have to persevere.

manipulations, and to communicate with all of the other parts of the computer system. As impressive as all of this sounds, the CPU is nothing more than a collection of electronic circuits. When all of those circuits are housed in one itty-bitty silicon chip (as they are in all popular home computers), that chip (and therefore, the CPU), is referred to as a *microprocessor*. In the Commodore 64, the microprocessor lives along with some other chips and gizmos under the keyboard.

The keyboard is the standard input device for the computer; it's the usual method you'll use to send information or requests to the computer. The standard output device your computer will use to communicate information back to you is your TV set. Lots of other kinds of input and output devices can be connected to the computer; more on those later.

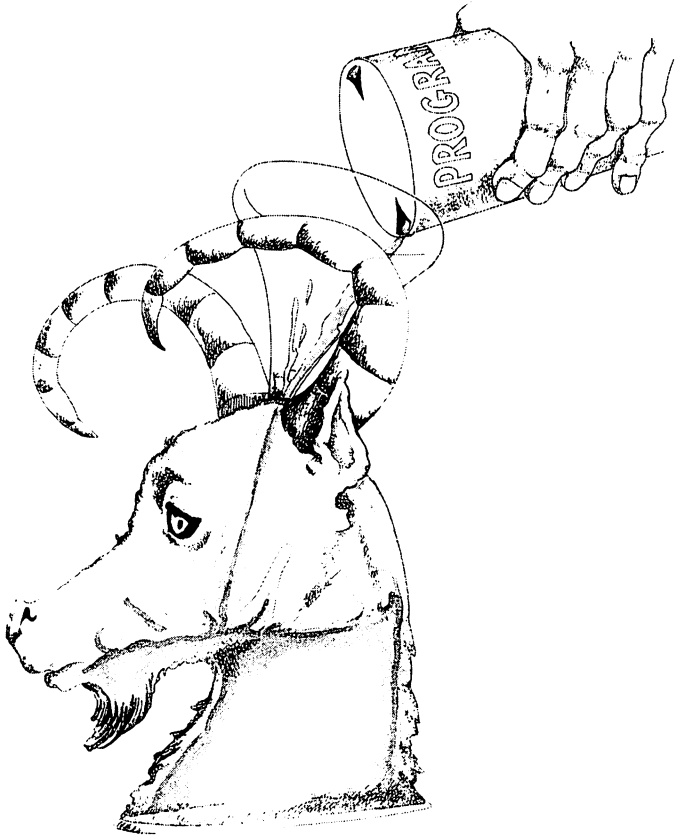
The CPU's main job is to interpret and execute program instructions. But like Alice, it can only handle one instruction and a few pieces of data at a time. So the computer needs a place to store the rest of the program and the data until the processor is ready for them. That's why the computer needs RAM.

RAM stands for Random Access Memory, which is the name applied to those circuits in the computer designed for the storage and recall of program instructions and data. RAM is divided into small memory locations called BYTES. 🐾

Each of these memory locations, like the houses in a subdivision, has a unique address, so that the computer can tell them apart when it's saving or retrieving some piece of information. That piece of information can be stored in any RAM location—you could pick one at random—and the computer could, if so instructed, quickly retrieve it. Hence the name, random access memory.

If all of this is a little bit confusing, don't despair. The important point is that general-purpose computers need RAM, and the more, the better. Since each byte can hold only about one character's worth of information, a computer needs thousands of

🐾 To confuse matters further, these units are sometimes called words.



THE "RAM" STORAGE TANK

bytes of RAM to do anything useful. So people usually talk about how many K of RAM a computer has, where K (for kilo) means about 1000 (bytes). 🐾 The Commodore 64 is named for the fact that it has 64K (64,000) bytes of random access memory built in, more than the average home computer can boast. 🐾

🐾 The real truth is that K usually means 1024, because computers think best in powers of two. But let's not nitpick.

🐾 Alert footnote readers know that it actually has 65,536 bytes. Don't tell.

There's one other thing you should know about RAM. In a microcomputer, the information stored in RAM is nothing more than a pattern of electrical current flowing through microscopic circuits in silicon chips. This means that when the power goes off, for whatever reason, the computer instantly forgets everything it was remembering in RAM. Memories like this are called *volatile*; most pocket calculator memories work the same way.

This could be a real inconvenience if it weren't for the fact that the computer has another type of memory containing information that's too important to lose (like how to understand commands from humans). This *non-volatile* memory is called ROM — Read-Only-Memory — because the computer can only read information from it; it can never write any new information on it. Some ROM is built into the computer when you buy it. The information in this ROM is available to the computer whenever you turn it on. You can also buy ROM cartridges to plug into the back of your C-64 that contain the information it needs to play games or understand another computer language. But whatever its form, the important limitation of ROM is that everything it remembers was burned in at birth; neither you nor your computer can store any new information in ROM.

So if RAM forgets everything whenever the computer is turned off, and ROM can't learn anything new, how will you save programs and data in the computer? Most likely, you won't! 🐾

Instead, you'll have to attach an *external* memory device to the computer to save information before the power is turned off. The two most popular kinds of external storage devices for microcomputers are the *cassette recorder* and the *floppy disk drive*. The basic idea of both of these devices is to give the processor a place to write information (translated from electrical pulses into tiny magnetic spots) in such a way that the same computer (or another one, for that matter) can later read the information back into the RAM.

🐾 There are other types of internal memory available for microcomputers that combine some of the features of ROM and RAM: PROM (Programmable Read-Only Memory) and EPROM (Erasable Programmable Read-Only Memory), but their use is beyond the scope of this book.

A cassette recorder stores (and retrieves) information sequentially on a cassette, similar to the way songs are recorded (and played) in a sequence by an audio cassette recorder. A floppy disk drive can store or recall information in any order on a flexible round *diskette* (also called a *floppy disk*). So an external storage device serves as both an output device (to which the computer can send information) and an input device (from which the computer can receive information). At least one such device is essential in your computer system. Cassette recorders and floppy disk drives are discussed in more detail in the next section.

There are lots of other input and output devices that can be attached to your microcomputer to extend its usefulness. In addition to the keyboard, the computer can receive information from such input devices as game paddles (volume-control-type knobs named for their early use in pong-type games), joysticks (gearshift-like devices a la Pac-Man), light pens (special wands designed to detect light patterns on the screen), or exotica like voice-recognition devices, which allow you to (in a limited way) talk to your computer! In addition to the standard TV, your machine can send output to, among other things, a printer (so that you can have a paper copy of the output), your stereo system (for true high-fidelity musical synthesis), or a speech synthesizer (to allow your computer to talk back to you)! Another device, called a *modem*, allows the computer to send and receive information via telephone lines to other computers.

From the computer's point of view, it doesn't matter which of these devices are used. Each input device is just another source of electrical signals, and each output device is just another place to send electrical signals. Read from here, write to there—the CPU doesn't care; it just dutifully follows instructions. And those invisible electrical instructions are the driving force behind your computer system.

Up to now, we've been discussing computer *hardware*— the physical parts of the computer system. At least as important is the *software*— the programs that tell the hardware what to do. You can type your programs into the computer memory using the keyboard, or you can buy cassettes, diskettes, or ROM cartridges containing program written by professionals (and

some not-so-professionals), but without some kind of software your computer is nearly worthless. And a good selection of quality software can make your computer priceless.

Enough jargon. You've got the background information you need to start planning your system. If you found some of it confusing, let it go for now. The only real problem facing you right now is too much freedom of choice. Software choices, hardware choices—lots of decisions to make. Just keep reading, though, and you'll find help.

PLANNING A COMPUTER SYSTEM YOU CAN LIVE WITH

Putting together a personal computer system that can work for you is easy if you do a little planning before you buy. The planning is easy if you can just answer two questions:

- (1) "What do I want my computer to do for me?"
- (2) "How much can I afford to spend on my system?"



It's important that you have answers to these questions before you lay down your money, because every purchase decision you make depends in part on your answers, and mistakes can be costly. So let's look for some answers.

(1) "What do I want my computer to do for me?" If you read the introduction you should have a pretty good idea of the kinds of things you can do with a home computer. Now you need to decide which of these applications are for you. If the introduction didn't give you enough information, browse through the later chapters for more detailed discussions of home computer applications. It might help to write down a list of things you want from your computer. Put the most important uses at the top of the list, and the least important at the bottom, so that it's easy to make adjustments if your answer to the second question forces you to do so.

(2) "How much can I afford to spend on my system?" This kind of question is easier to answer if you have a computer to help, but you'll have to make do. A complete computer system is still no small investment if you buy it all at once. A common strategy is to buy a starter system consisting of a bare minimum of equipment, and to add the more exotic peripherals each time the wallet gets heavy. (Of course, if you build your system this way, you can't take advantage of the package discounts offered by many stores.)

Then there's always the possibility that you can use part of the national debt to help finance your computer. If you'll be using your computer to help you run your business or do your job, or if using your computer will help you to maintain or improve some job-related skill, you may be able to deduct part or all of the cost from your taxes. Congress likes to rewrite the tax laws each time a new lobbyist or president hits Washington, and the IRS changes its interpretations of the laws at least as often, so it's best to check with an expert before you spend your anticipated windfall. But if you can work it out, you'll be able to stretch your buying power considerably.

Once you're satisfied with your own personal answers to the two questions, you're ready to start considering specific items

for your hardware want list. Here are the major items to consider:

Choice #1: Cassette recorder and/or disk drive? You'll need at least one external storage device. Which one you choose will be dictated by your needs and your budget.

The cassette recorder (Commodore calls it a *Datasette*) works pretty much like your standard music-and-voice-type tape recorder, except that it deals in chunks of computer information instead of notes. You can tell the computer to save a program or some data, hit the record button, and wait for the information to be recorded as little magnetic spots on the tape. Later, you can wind the cassette back into the computer, punch the play button, and wait for the computer to find the item you're looking for and read the information from the tape back into internal memory.

The datasette recorder is simple in concept, it can withstand reasonable abuse, it's relatively inexpensive, and it uses standard audio cassettes. But it's not without problems.

The biggest drawback to storing information on cassettes is the speed, or lack of it. Cassettes just don't move very fast, and waiting for a program or data to be transferred to or from tape gets old in a hurry. This is compounded by the SEQUENTIAL nature of the cassette. If you have several packages of information stored on a single cassette, and you want to retrieve the last one, you must either use the fast forward button and the tape counter to (hopefully) locate the position of the desired item before you load it from tape, or drum your fingers on the table while the datasette laboriously searches through the tape for the cherished information. To make matters worse, the datasette occasionally has trouble finding previously stored information on a cassette, so you may have to go through the whole process again!

But storing information on cassettes requires more than just your time; it also requires extra effort on your part. The computer doesn't keep track of what you have stored on any given

tape. It will gladly record new information right over the top of an important program if you tell it to. So to avoid potential disaster, you have to keep an index of the exact contents of each tape you use. No big deal if you're an occasional user, but not exactly the kind of thing you'd build a "Fun with Computers" course around, either.

There is a better way, as it turns out. If you have the extra money, you can invest in a floppy disk drive instead of a datasette, and save yourself a lot of frustration in the bargain.

If the datasette is similar to a simple audio cassette recorder, it's tempting to think of a floppy disk drive as a magnetic phonograph. It stores information on a rapidly rotating disk using a device (called a READ/WRITE HEAD) that touches one spot on the disk at a time. It can rapidly retrieve information from any part of the disk without regard for the order in which the information was recorded, in the same way that you can quickly move the record needle to any cut on a record. (In jargonese, the disk drive is a random access device rather than a sequential access device.)

But the magnetic phonograph analogy obscures some of the most important features of the floppy disk drive. For one thing, the read write head on the drive is capable of recording information as well as retrieving it. For another, the disk rotates much faster than your phonograph turntable (not 33 or 45 revolutions per minute, but 300). This makes for much faster information access. Another thing that improves the response time of the disk drive is the organization of information into pie-shaped *sectors* and concentric *tracks*, as opposed to the long spiral groove of the phonograph. But the most distinguishing feature of the disk drive is its smarts.

The drive actually has a built-in microprocessor to do the housekeeping chores that the datasette expected you to do. It keeps an up-to-date directory of information on each diskette, organizes the information on each disk for efficient access, and does what it can to protect information on the disk from accidental erasure.

It's easy to see why most serious computerists prefer disk to tape. But all this is not to suggest that the datasette is without merit. It does cost a lot less than a disk drive, and if you don't anticipate heavy usage, you might be willing to sacrifice speed and convenience for spending money. You can always add a disk drive later. 🐾

There is an advantage to having both disk and tape units in your system: programs written on othe Commodore 64s—by professional programmers or your neighbors—can be transferred to your computer via disk or tape (and vice versa). But not all programs are available on both media. So having a disk drive and a datasette maximizes your options when it comes to software shopping.

Choice #2: TV or monitor? Your Commodore 64 comes complete with an RF modulator. This gizmo acts like a miniature television transmitter, converting the computer's output signals into broadcast waves which your TV set can translate into color displays. This means if you have a color TV in your home, you can use your C-64 without having to invest in an additional output device. And if you have an extra color set lying around, you can use your computer without having to compete with the family for tube time. If you don't have an extra set, and you don't want to schedule your computer time around football and soap operas, you might want to pick up a cheap portable at a garage sale.

Then again, you might not. Television sets are not designed to be used as computer viewing screens. While they work just fine in this capacity for most purposes, the truth is that a TV picture lacks the sharpness needed for some precision displays. (This is the main reason why most home computers only print 40 or fewer characters per line on the screen; smaller characters

🐾 For that matter, you can add up to five disk drives to your Commodore 64 system. Unless you're working with very large amounts of information, you'll most likely be content with just one or two. But some folks find it comforting to know that there's lots of room to expand. If you're one, you'll feel even more comfortable knowing that it's possible to hook a Winchester disk drive to your 64. Winchester drives contain permanently built-in hard disks that can store more and move faster than their floppy cousins. They also cost much more.

would not be readable on a color television screen.) So if you're going to have to buy a viewing screen anyway, you might want to consider a computer *monitor* instead of a television. A monitor looks like a TV without all of the knobs, and it works pretty much like the display part of a TV, too. But since it was designed to be used with a computer, it has a much sharper display. So if an extra-sharp picture is important to you for whatever reason (detailed graphic displays, large quantities of word processing work, etc.), a monitor would probably be a valuable addition to your system—and it won't cost you much more than you'd pay for a comparable TV set.

Of course, if cost is a major factor in your decision, and you aren't too concerned about color, you might want to consider a *monochrome* display: a black-and-white TV, a black-and-white monitor, a green-and-black monitor, or an amber-and-black monitor. Besides being cheaper than color displays, these devices generally have much sharper pictures than their color counterparts do. In fact, with a monochrome monitor you can take advantage of devices that allow your computer to display 80, rather than 40, characters per line. (See Chapter 9 for details.) But be forewarned that many programs written for the C-64 assume you have a color screen. Some of these programs generate displays that are difficult to read in monochrome. Also many monitors, especially those of the monochrome persuasion, lack sound capabilities. If you want your computer to sing to you while you're using one of these you'll have to use an external speaker and amplifier—a good idea anyway if you want true high fidelity.

Commodore makes a reasonably-priced monitor that can be used for both color and monochrome displays, and has a built-in speaker and amp for sound reproduction. But there are lots of other good monitors on the market, and most of them are completely compatible with your Commodore computer.

Choice #3. To printer not to print. Another decision you'll have to make when putting together your system is whether to invest in a printer, and if so, which one. A printer is an output device that attaches to your coputer, allowing you to create *hard copies* (that's computerese for paper copies)

of your computer creations. If you're planning to use your computer mainly for gaming and casual programming, you can certainly get by without a printer. But if you'll probably want to buy a printer as soon as possible, to avoid having to use a pencil to translate output from the screen. If you intend to use your computer as a word processor, a printer is a must.

There are so many variables that choosing a printer can be a very complicated process. If you want to keep things as simple as possible, consider buying a printer specifically designed to work with the Commodore 64 and VIC-20 computers. Other printers can be connected to the C-64, but not without a special interface to function as a communication channel between machines. Commodore has introduced several printers that connect directly to the C-64 (some of which have been dis-introduced), so you aren't sacrificing your freedom of choice completely by choosing a C-64-compatible printer.

Take a sample printout from one of the earliest of these printers—the VIC 1525. The 1525 is called a dot matrix printer because each character is made up of a series of dots. The tails on the lowercase g, j, p, q, and y don't hang down below the other letters; the printer doesn't print *true descenders*. While the text is very readable, it may not be good enough for many business and personal applications because it doesn't look like the typewritten copy we've become used to.

Commodore markets other dot matrix printers with true descenders and more dots per character. Output from printers like this one look almost — but not quite — like they were produced by a typewriter. Unlike a typewriter, though, a good dot matrix printer can produce complex graphic designs, proportionally-spaced text (so skinny letters like “i” take up less space on a line than fat letters like “w”), and a variety of typefaces (italics, boldface, etc.).

If you don't like the type produced by dot matrix printers, you'll probably want a letter-quality printer. (That's the term for any printer that produces copy that's pretty much indistinguishable from typewritten text.) Letter-quality printers come in lots of designs, some of which allow the printer to be used without the

computer as a simple electric typewriter. Although they can produce very high quality copy, they can't print graphs or pictures using Commodore's special graphics characters.

When you're choosing a printer you should also take into account the speed of printing, the noise level, and compatibility with your computer and the programs (like word processors) you'll be using with it. Speed is a major factor if you'll be doing lots of printing, and noise level is crucial if the printer is within earshot of other human activities. But the most important single consideration is compatibility. Make sure before you buy that the printer can be easily made to work with your Commodore 64. Unless the printer was specifically designed for use with the C-64 or the VIC, it'll need a special RS-232 or IEEE interface, but that may not be enough. Double check before you buy, to prevent headaches later. And finally, if you choose a printer because it has all kinds of special word processing features, make sure that those features are compatible with the word processing program you'll be using. Try them and buy them together if you can.

Choice #4: Joysticks, anyone? If you plan to use your computer for game playing, you'll probably want at least one joystick. Most arcade games, and some educational programs, won't work properly without a joystick. Commodore markets a low-cost joystick for use with its computers, but serious gamers generally prefer other brands. Any joystick that will work with an Atari game machine or computer will work with the C-64, so you have lots of choices. As usual, the best way to shop for joysticks is to try as many as you can, and choose the one that works best for you. Look for one that seems sturdy, and responsive, and above all, comfortable. If there are little hands in the family, or lefties, take them into account. (The "fire" button on many joysticks is positioned on the left side of the base, where southpaws can't easily get at it.) My current favorite is the Prostick II, from Newport Controls. It's tough, responsive, comfortable, and ambidextrous. But new ones are introduced on the market all the time, so shop around. Or watch the pages of *Creative Computing* magazine — they review the latest game controllers from time to time.

Choice #5: Modem? A modem (short for MOdulator/DEModulator) is a device that can translate your computer's

output signals into a form that can be transmitted by telephone lines to other computers, and vice versa. Depending on your communication needs, it may prove to be one of the most valuable peripherals you can buy for your Commodore 64. While lots of modems are available on the market, Commodore makes a low-cost modem that can be connected directly to the C-64 without modification. Modems and their uses are discussed in detail in Chapter 9.

There are, of course, lots of other doo-dads that can be attached to the Commodore 64, many of which are discussed in Chapter 9. But these are the ones that most home computerists use. You should now be able to turn your “what I want to do with my computer” list into a “what I want to buy for my computer” list. You can then use a current price list to figure the total cost for the hardware part of your projected system. Hopefully this number is considerably less than your “how much I can afford to spend” figure, because you’ll probably want to use some of that money for software, diskettes, cassettes, printer paper, books, and magazine subscriptions. If it doesn’t work out that way, you’ll either have to lower your expectations or raise more money before you go shopping.

THE NUTS AND BOLTS OF BUYING

If you’ve looked through any of the microcomputer magazines, you’ve probably noticed ads for companies that sell computers and supplies by mail at discount prices. Most of these firms are reputable and honest, and the products they sell are in fact the same items you’d probably pay more for in your neighborhood computer store. Even so, your local shop may be able to offer you more for your money in the long run. Before you dial that 800 number, consider these advantages to shopping locally:

- A good shop will let you try out each piece of equipment or software before you lay your money down; mail-order shopping means buying blind.
- If you uncrate everything and discover the installation instructions aren’t clear, it’s usually easier to get help by phone from a local store.

- If the manual for your disk drive is missing from the package, you're a lot more likely to be able to get an immediate substitute.
- If you get everything wired up and something doesn't work right, returns are easier.
- When your disk drive breaks down, your local shop may be able to have it working and back in your hands sooner than a mail-order house. (Especially if it has a service department.) Some dealers even offer loaner equipment to keep your system running while repairs are being made.
- When your computer does something you don't understand (and it WILL!), it's nice to have a local number you can call for help.
- Many computer stores offer classes in programming and other mysterious arts to their customers.
- And finally, the local computer store is a great place to meet fellow computerists. Some stores sponsor user groups, so computer users can get together and share programs and stories.

Of course, not all local shops provide all of these services. (Discount and department stores rarely do.) So it's important to ask some questions before you buy. (Servicing? Loaners? Help?) The answers you get to these questions can tell you something, but equally important is the feeling you get for the store while you're there. Are these the kind of people tht you can trust for straight answers and honest help? Will this store be around next year if your printer fizzles out? Will it still be selling computers, or will the computer department have been replaced by an organic jewelry boutique? If you're a new computerist, it pays to shop around.

While you're shopping, don't forget the accessories: cassettes, diskettes, software, printer paper, interfaces (if you're using an exotic printer or other device), and maybe a book or two to help you along. (Appendix A includes an annotated list of recommended books.)

CHOOSING A HOME FOR YOUR COMPUTER

Where you keep your computer once you've bought it is your business. Your personal patterns of work and play should determine your computer's location. But before you go to the trouble of hooking up all those cables, please take a minute to consider these factors.

In the first place, think about the traffic patterns of your family. If you just have one TV, and you plan to use the computer mainly for video games, then you'll be happy having the computer in the TV room. But if you have lots of serious work planned for the machine, you should consider a quieter location, even if it means investing in another TV or monitor.

While you're exploring other spaces, keep an eye out for electrical outlets. Home computers don't take much power, but they can be pretty picky about the quality of power they receive. In particular, it's not a good idea to plug your machine into any circuit that has big fluctuations in voltage, because momentary surges in power (called spikes) can damage your data or your computer. These power surges are usually caused by sudden changes in the power demands of large energy-eaters like heaters, air conditioners, and table saws, so make sure your computer isn't sharing a line with one of those. (If you have any doubts, or if your local electricity supply is particularly flaky, you can buy a device to filter the spikes out of your computer's dinner. Ask your dealer or watch the ads in the micro mags.)

Wherever you put your computer system, you're likely to need a multi-way adapter or an extension cord with multiple sockets to plug all the pieces in. An adapter or cord with a built-in on/off switch can be especially handy, because you can use that one switch to turn everything in the system on and off simultaneously. It also saves you the bother of having to unplug the computer when you aren't using it, which is what you probably should do if you don't plug it into a switchable outlet. Why? Because the on/off switch on the computer doesn't turn off the juice to the power supply, so that box stays warm and eats energy as long as it's plugged in.

Of course, if you plan to do any telecomputing, you'll have to locate your computer near a phone (or locate a phone near your computer.)

Try to choose a space with room to spread out your system, because some of the components don't get along very well in close quarters. The cassette recorder might misbehave if it sits too close to the TV or the power supply box or the telephone; it's very sensitive to magnetic energy. For the same reason, your diskette library should be kept a couple of feet from the telephone and the TV. If you have a printer, remember to allow space behind and below it for the box of paper that feeds it.

And finally, try to take your own comfort into consideration when choosing your computer's home. Common sense says you'll be happier computing in a well-heated, well-ventilated, dust-free area. And if you plan to use your computer heavily for work and/or play, the *National Institute for Occupational Safety and Health* has a few other tips for you. 🐾

For one thing, the lighting should be adjustable so that it can be cut in half while you're staring at the computer display. If the computer will be located near a window, light from the window should be reduced to eliminate glare and reduce eyestrain.

To protect the rest of your body from fatigue, the Institute advises that your keyboard and chair should be arranged so that while you are typing your arms are horizontal, your knees are bent at right angles, and your feet are flat on the floor. (They also suggest that you take a 15-minute break every two hours. Good advice, but then again, that's your business.)

PUTTING IT ALL TOGETHER

Once you've chosen a home for your computer system, installing it is relatively easy. Each piece of equipment comes with

🐾 Actually, these tips are for workers at video display terminals, in response to widespread reports of headache, eyestrain, fatigue, and more serious maladies. See *InfoWorld*, August 26, 1981, page 21, for more details.

an instruction manual or user's guide. If you're new to computing, you're likely to find the how-to-use-it sections of these texts intimidating. 🐾 But never mind. The how-to-install-it sections are generally very clear, and you can always ask your dealer if you run into problems.

Of course, you don't need to hook-up the whole system at once; you can start with just the computer and the TV. The next two chapters will help you to become familiar with that combination; Chapter 4 will orient you to the datasette, the disk drive, and the printer; Chapter 9 talks about modems.

CARE AND FEEDING OF YOUR COMPUTER

You spent a lot of money on your computer, and you want to take care of it. There's no need to baby it, but a little bit of courtesy can go a long way. So here are some tips to protect your investment and your peace of mind.

The solid-state circuitry of your computer is protected by a tough plastic case that can withstand a surprising amount of knocking around. There are three ways for hazards to get through the protective box to the delicate circuitry—the power supply, the keyboard, and the plug-in ports.

As mentioned earlier, the two major ways to protect your computer from power-supply problems are (1) don't depend on the on/off switch on the computer to turn it off between sessions (unplug it or use an external switch); and (2) use a clean power source or, if necessary, a filtering device.

What about the keyboard? If you like flirting with disaster, here's a useful tip: spilling a drink on the keyboard

(continued)

🐾 If your computer came with the first edition of the Commodore 64 User's Guide, you're especially likely to find it intimidating, because it's full of errors. The October/November 1982 issue of *Commodore Magazine* contains six pages of corrections.

is one of the surest ways to mess up the guts of your machine. The liquid can short out your circuitry faster than you can say (@%#?●! Peanut butter and other gooey substances are less of a threat to the internal works and more of a threat to the keyboard mechanism itself. For maximum security, do your consuming and your computing in different places.

A good way to protect your keyboard from dust and other hazards when it's not being used is to keep it covered. Several custom covers are available for the Commodore 64/VIC 20 case; ask your dealer or look for ads in any of the Commodore-oriented magazines listed in Appendix A. Or make one yourself! (Suitcase-type cases are also available for carrying your computer and accessories around.)

The other potential gateways to trouble are all of those input ports on the back and side of the computer box. An improperly-connected accessory can confuse or even damage your computer, so connect each device carefully according to instructions. Make sure the computer is OFF before you insert or remove any cartridge. Make sure all accessories are compatible with the Commodore 64. If you're an electronic hobbyist, make sure you know what you're doing when you plug in a home-made peripheral.

None of these tips will protect your investment against fire, theft, or volcanic eruption. Hopefully, your homeowner's insurance policy will. But don't assume it does! If you use your computer for any kind of business or work-related purpose, it may not be covered by your policy. Ask your agent to be sure. If it's not, you may be able to extend your coverage for a reasonable price. Or you can buy a policy from a company that specializes in this kind of thing. Ask your dealer or peruse the ads in *Infoworld* or one of the microcomputer monthlies.

CHAPTER 2

Getting Started: Bit by Bit

*I hear and I forget,
I see and I remember,
I do and I understand.*

—*Ancient Chinese Proverb*

Children are driven to discover and learn new things; adults often seem to have lost or suppressed that drive. Teachers of computer classes and workshops invariably report that children usually jump right in and start pounding on the keyboards, while adults need to be gently encouraged to overcome their fear. (Fear of what? Damaging something? Doing something that looks dumb? Seeing things in a new way?)

This chapter is designed to show you how to talk to your computer and make it do a few interesting things, and to prepare you for using and/or programming your machine. But the chapter can't do its job without your active participation. Don't try to memorize every detail, and don't worry if you don't understand everything. Think of this chapter like a guided tour of Shanghai; you may learn a few things that will someday prove important to you, but your main goals are to explore a different world and to have a good time!

So whatever your age, be a kid for a while and play! You won't hurt your computer, and it won't hurt you. In fact, you'll probably end up liking each other!

One last word before we start! If at any point in your explorations something happens that you don't understand, don't panic. Just try again, or go on as if nothing happened, or turn the computer off for a few seconds to clear its circuits, or take a break to clear your circuits, or skip over that part altogether. If you can't remember how to do something that you need to do, you can always refer back, or use the pull-out crib sheet inside the back cover of this book. Have fun!

LET YOUR FINGERS DO THE TALKING

We'll start with the keyboard—your main tool for communicating with your computer. In the next section you'll learn how to use the keyboard to give simple commands to the computer. Until then, we'll just concentrate on using the keyboard to put things on the screen.

First off, just look it over. If you look at just the letters and the numbers, you'll recognize that these keys are arranged exactly like they are on a typewriter keyboard—numbers in order across the top, letters in the next three rows arranged the way we've come to expect them. 🐾 This means that if you're a touch-typist, you're already familiar with the major portion of the keyboard. If you never learned to type, that's OK too. With the right software (program) your computer can be an excellent typing teacher, and Chapter 6 will help you find the right software.

To find out what those other keys and characters are for, you'll need to have your computer on. So plug it in and flip the switch on the right-hand side of the console. Turn on the TV to the appropriate channel (3 or 4) and switch the antenna junction box to "COMPUTER." Your TV screen should now be displaying the following message:

```
-----  
**** COMMODORE 64 BASIC V2 ****  
64K RAM SYSTEM 38911 BASIC BYTES FREE  
READY  
■  
-----
```

🐾 This seemingly senseless arrangement of the alphabetic keys is known as the QWERTY keyboard, named for the order of the letters in the second row. This clumsy keyboard is our heritage from the earliest days of the typewriter, when the moving parts of the machine could easily jam if the operator typed too fast. The alphabetic keys were arranged like this to slow the typist down! Lots of people since then have developed keyboards that are easier to learn and faster to type with, but technological traditions die hard. (If you delve deep enough into the inner workings of your computer it's possible to reprogram the keyboard to some other arrangement!)

The message isn't of much importance to us now. 🐾

The cursor. What is important now is that flashing blue square. It's known affectionately as the *cursor*, think of it as a current position indicator; it tells you where the next character you type will appear. If you press the key marked RETURN on the right side of the keyboard, the cursor will move down to the next line on the screen. (Try it if you like.) You can also move the little rascal around the screen with those two buttons marked CRSR, located right below the RETURN. Try pressing each of them a couple of times to see how they move the cursor down and to the right, respectively. (If this doesn't work right, check to see if the SHIFT LOCK button (lower left keyboard) is engaged; press it once to disengage it if it sits lower than its neighbors.) Got it? Now try pressing each of those keys while you're holding one of the SHIFT keys down; the cursor should move up and to the left this time.

Both of these cursor-moving keys will automatically repeat if you hold them down for more than a second, shifted or not. Try sending the cursor as far to the right as it will go by holding down the cursor right key. Notice how it automatically "wraps around" and reappears on the left-hand side of the next line of the screen and continues from there as if nothing had happened. Now try the same thing to the left (same key, shifted). Next, use the cursor-down key to try to send it off the bottom of the screen. The cursor never leaves the screen, but the printed message scrolls right off the top. No matter; you didn't need that message, and in a minute you'll be typing your own, anyway. Play around with the cursor-moving keys until you understand them, then send the cursor home to the upper-left corner of the screen by pressing the CLR/HOME key in the upper-right portion of the keyboard.


Now you're ready to start typing. For the time being, don't use the SHIFT key; any letters you type will be capitalized without

🐾 In case it's important to you: you no doubt know what Commodore 64 is, and Chapter 3 defines 64 K RAM SYSTEM; BASIC is the name of the language that is built into the ROM of your computer; V2 means that this is Commodore's version 2 of BASIC; 38911 is the amount of RAM that's available for you to fill with a BASIC program and data (you can get at the rest of the RAM with some fancy programming tricks); and READY is a come-on.

shifting. Also, avoid using the RETURN key for now; if you fill a line the text will automatically wrap around to the beginning of the next line, even if it's in the middle of a word. (If you do accidentally press RETURN in the course of your typing, the computer will probably respond by typing SYNTAX ERROR back at you. Just ignore it—it's just telling you that it doesn't understand what you typed, and that's OK. More on this later.) So with that in mind, type HELLO, your name, or whatever you like, and watch the cursor leave a trail of letters as it moves across the screen.

Changing colors. You may find the blue-on-blue text difficult to read. Try this. Find the key that says CTRL (for ConTRoL). This button is like the shift key on a typewriter in that it doesn't do anything by itself, but works in conjunction with other keys. For example, hold the CTRL key down while you press the key with the 2 on top and WHT (for WHiTe) on the front face. Now type HELLO (or something) again and look at the difference. (I used to type CTRL-2 first thing when I turned on the computer to save my eyes, but my latest TV looks great in two shades of blue.)

Notice that each of the first eight number keys has a color abbreviation stamped on its front; experiment with the CTRL key in conjunction with other numbers to see what text in those colors looks like—you'll probably find some colors nearly unreadable against the blue background (especially blue (CTRL-7), which makes the cursor and your typing invisible!).

Where is the light blue that we started with? It's hiding, along with seven other colors, behind those same number keys. To see those other colors, repeat what you just did, using the COM-MODORE key  (the one at the lower left corner of the keyboard) instead of the CTRL key.

Now that you know how to set the text color, set it to white, or anything you like, and try this trick: hold down CTRL while you punch 9, then type a few words. You are now typing in *reverse video*, which you turned on with CTRL-9. While you're there, try holding the space bar down for a few seconds to create a solid white bar. Change colors (as before) a few times and

make different colored bars with text and/or spaces. (You can use these bars to adjust your TV's color.) When you've had enough of reverse video, turn it off with CTRL and 0 (zero, the one with the line through it). 🐾

Switching modes. For another cheap thrill, try pressing the R, S, Z, and X keys with the SHIFT held down. Instead of typing letters, you're typing the shapes in the squares stamped on the right front of the keys. Try some others. (SHIFT-LOCK is handy here.) If you hold down the COMMODORE key instead of the SHIFT, you'll display the shapes shown on the LEFT front face of the keys. All of these strange shapes are called *graphic characters*, because they can be used (in conjunction with the color-control and reverse-video features) to build graphic displays for video games, educational demonstrations, and other programs where text isn't enough. You can type any of these shapes when your computer is in UPPER-CASE/GRAPHIC MODE—the mode it's in when you first turn it on.

Upper-case/graphic mode is great for drawing pictures and writing programs, but it has one serious drawback—it has no lower case letters, which are essential for word processing and the like. Your computer has another mode called UPPER/LOWER-CASE MODE, which sacrifices some of the graphic characters (the ones you shifted for) for the lower-case alphabet. To get into that mode, press SHIFT and COMMODORE at the same time and watch what happens. Now you can type sentences, using the SHIFT key as you would a normal typewriter—to make little letters big, and to type the special characters on the tops of the number keys. (Remember—no RETURN!) Though you can't get at the right-front graphic characters on the keys in this mode, you can still use the COMMODORE key to type the left-front characters.

Now try the SHIFT-COMMODORE combination again. You're back in upper-case/graphic mode, and all those little letters you typed are now big again. You can use these two keys to

🐾 The difference between the number zero and the letter O is very important to computers, so most computer people and books put slashes through their zeroes to make the difference obvious to humans. But beware, some people and books diabolically slash their letter O's instead! Consensus is rare in this industry.

gether just about anytime to change the display on the screen from one display mode to the other. Nothing changes inside the computer when you do this except the messages it sends to the TV about which characters to display. (Most programs you'll use will automatically put the display into the appropriate mode; this is simply a method of manually choosing.)

Editing. What if you make a mistake typing? This section will show you how much easier it is to correct typos on your computer than on a typewriter.

First, clear the screen and send the cursor home by pressing SHIFT and HOME together. (That's the overkill method of eliminating typing errors.) Next, type what you see here, warts and all. (You need to have errors before you can correct them.) Remember to avoid RETURN; let the text just wrap around.

```
We all liv inn a yellllllow submarinee!  
A yeow submarine? A yellll ow submarine#
```

Lots of errors to correct. Let's start by repairing the final punctuation mark. Your cursor should be positioned just after the offending #. You could just move it back one space with the cursor-left key and type ! right over the #. But the usual way to correct this kind of error is to use the key in the upper-right corner of the keyboard labelled INST/DEL, for INSerT/DELeTe. Unshifted, the DEL key works like a combination backspace key and eraser—it deletes the character just to the left of the cursor (usually the one you just typed), and puts the cursor in its place. So hitting DEL will erase the # and leave the cursor positioned so that you're ready to type !.

You can use the DEL key to erase the embedded blank in yell ow, too. Use the cursor-left (shift) key to put the cursor on the o to the left of the blank; then hit DEL. Notice how the cursor and everything to its right shifts left to fill the void. Hit DEL again to remove the third l.

Shifted, that same key allows you to insert extra spaces in text. Move the cursor over to the o in yeow, and press SHIFT-INST/

DEL. See how it moves the cursor and everything to its right one space rightward? Do it again. Now you're ready to type ll into the two-space hole you just created. Got the idea?

You can practice using this important key by moving the cursor up to the first line (SHIFT-CRSR-UP) and correcting the misspellings there. Then, if you want, you can insert extra spaces to make the text look pretty and eliminate the word-chopping at the end of the line. When you're through, clear the screen and send the cursor home (SHIFT-CLR/HOME).

If you find this a little awkward, be patient. In no time you'll have the cursor flying around the screen. If you like the convenience of correcting errors this way, make sure you read in Chapter 7 about word processing. You ain't seen nuttin' yet!

The remaining keys on the keyboard won't be of much use to you until you start working with programs in the next section: RUN/STOP can (often) be used to stop the computer in the middle of a program; RESTORE can (sometimes) be used in conjunction with RUN/STOP to return things to the way they were before you ran a program; and the tan keys to the far right (called FUNCTION KEYS) are like wild cards—they can be programmed to do (almost) anything.

Finally, let's go back to the RETURN key for a second look. Up to now, if you've been successful at avoiding RETURN, the computer has been simply ignoring everything you typed, except for mindlessly displaying it on the screen. When you hit RETURN, you signal the computer that the current line contains a message for it—usually a command telling the computer to do something. 🐾Type this and press RETURN:

🐾 In this context, a "line" can be more than 40 characters long. If you type 80 characters before hitting RETURN, all 80 are part of the current line from the computer's point of view, even though they cover two lines on the screen.

WASH THE WINDOWS!

When the computer responds SYNTAX ERROR, it's just telling you in a grumpy way that it doesn't recognize the command. The catch is that you have to tell the computer to do something it knows how to do, and you have to tell it in a language it understands. And it doesn't understand plain English. (And it doesn't do windows.) You're ready now to learn how to talk to your computer in a language it DOES understand.

MAKING YOUR EXPENSIVE COMPUTER INTO A CHEAP CALCULATOR

Your computer understands BASIC (Beginner's All-purpose Symbolic Instruction Code), a computer language developed at Dartmouth College to simplify the process of communicating with computers. The BASIC language is made up of commands and statements which are built around English-like keywords. Your computer is programmed to respond in very specific ways whenever it sees a keyword that it recognizes. The command you'll be using in this section is PRINT, which, in its simplest form, tells the computer to write something on the TV screen. 🐾

You'll use the PRINT command to get comfortable with talking to the computer and editing in BASIC, and in the process learn how to use your computer as a calculator. You'll start with simple calculations that would probably be easier to do on a \$5 calculator. But we'll also see how BASIC can make short work of more complicated problems.

🐾 When BASIC was invented, most computer output was on paper rather than screens; hence the word PRINT.

A NOTE FOR SUFFERERS FROM MATH ANXIETY AND ALGEBRA ALLERGIES. Try to think of this section as just a continuation of the keyboard tour, using numbers instead of colors and words. And don't feel like you have to understand everything here. If, after typing a few of these examples, you develop symptoms, feel free to skip to the next section, POKING AROUND IN MEMORY, which is 97 percent calculation-free.

When you're typing BASIC commands like the ones in this section, remember two things: First, every command you type must be followed by a RETURN before the computer will even look at it, let alone act on it. Second, your computer isn't very smart about some things. If you type PRINT 5 instead of PRINT 5, it won't have any idea what you want it to do, because PRINT isn't a keyword that it has been programmed to recognize. It will simply respond SYNTAX ERROR again.

When you mistype something and get a syntax error message, you have two choices. You can type the offending line from the beginning, being careful to type it correctly this time. Or, you can move the cursor to the error, correct it using strikeover or INST/DEL, and hit RETURN; the computer acts as if you typed the whole line anew. This second method is especially handy for correcting long lines. Experiment with both methods until they feel comfortable.

Basic Arithmetic So with that in mind, try typing in each of these examples exactly as they're shown here. It's OK to leave out the blank spaces—they're optional in BASIC. (Remember to end each command with a RETURN.)

```
-----  
PRINT 5  
? 12.345  
-----
```

The question mark is the abbreviation for PRINT in Commodore BASIC to save your fingers and your time. PRINT and ? are completely interchangeable in the remaining examples. (Don't shift for the plus sign.)

```
-----  
PRINT 5 + 3  
? 66 + 34, 1 + 2 + 3 + 4, 16 + 33.3  
-----
```

See, it really can compute! Note how you can even ask it for several computations at once, as long as they're separated by commas.

If you're in a hurry and you have no interest in programming or calculating, consider the rest of this chapter optional.

Subtraction is just as easy:

```
-----  
PRINT 98765.43 - 21  
PRINT 9 - 6 + 2  
-----
```

Note that the last result was calculated by performing the addition and subtraction operations from left to right: $9 - 6 = 3$, then $3 + 2 = 5$. Sensible enough. But suppose what we really wanted to do was add the 6 and 2, and then subtract that from 9? We can use parentheses to group operations, like so:

```
-----  
PRINT 9 - (6 + 2)  
-----
```

(I bet there's not a parenthesis button on your \$5 pocket calculator)

Multiplication and division work about the same way. In BASIC, the symbol for multiplication is the asterisk (*), and the symbol for division is the slash (/). Armed with that knowledge, see if you can predict how your computer will respond to each of these commands:

```
Print 1.2 * 5
? 36.9 / 3
? 6 * 5 / 2 * 3
? 6 * 5 / (2 * 3)
? 2 / 3
```

Look at the answer to the last one. That gives you an idea of the accuracy of your computer's arithmeticker. Ten significant digits (rounded to nine) should be good enough for most recipes.

You can, of course, ask the computer to do calculations that involve multiplication, division, addition, and subtraction all together. (In this next set, try using the cursor control keys and overstriking to change the first PRINT statement into the second, and then into the third. Each time you're done changing the line, hit RETURN and a new answer will take the place of the old one on the screen. Handy?)

```
PRINT 1 / 2 + 3
PRINT 1 + 2 / 3
PRINT 1 + 2 * 3 - 4
```

To understand how the computer came up with these answers, you need to know its strict rule of order for mixed calculations. (Computers have strict rules for everything; they can't tolerate ambiguity.) Simply stated, the rule is: do all multiplications and divisions (from left to right) before doing the additions and subtractions (from left to right). So, in this ridiculously complicated example,

```
PRINT 1 - 4 / 2 + 3 * 6 - 5
```

the computer will do the division first, replacing the $4/2$ with a 2. Multiplication comes next, replacing the $3*6$ with 18, leaving $1 - 2 + 18 - 5$. The remaining calculations are done lock-step, left to right: $1 - 2 = -1$; $-1 + 18 = 17$; $17 - 5 = 12$.

Of course, you can always insert parentheses to make the computer do the calculations in a different order; it will completely evaluate the mini-expressions inside the parentheses before moving out to the larger expression. Here, for example, are the wrong way and the right way to calculate the average of three numbers. (For more editing practice, use the same line of text for both examples: just insert (INST) parentheses in the right place and RETURN again.)

```
PRINT 3 + 2 + 1/3
PRINT (3 + 2 + 1)/3
```

Parentheses are especially handy if you have trouble remembering the computer's ordering rules, because you can always put enough parentheses in to ensure that you get the results you want.

Advanced arithmetic. Addition, subtraction, multiplication, and division are about all most folks need in the way of arithmetic most of the time. But for some people the world occasionally demands more high-powered stuff. So BASIC has an exponentiation operator, symbolized by the \uparrow , which allows you to raise a number to a power (multiply it by itself the number of times specified after the \uparrow). (People who do math with pencils instead of keyboards don't need an exponentiation operator; they just write the power smaller and higher, like so: 4^3 .) Here is an example calculated with the \uparrow and then with *s:

```
? 4  $\uparrow$  3
? 4 * 4 * 4
```

You can, of course, use the \uparrow with the other four operators in an expression, as long as you know that the computer will do the exponentiation (left to right) before it does anything else. An example from geometry, for calculating the area of a circle with a radius of 4:

```
?  $\pi$  * 4  $\uparrow$  2
```

(π is the shifted \uparrow key, but it won't show up on the screen as π unless you're in upper-case/graphic mode.)

Your pocket calculator has a square root button. In BASIC you take the square root of a number—say 225—by typing SQR(225). Here are some examples of using SQR, including another one of geometry's greatest hits—the hypotenuse formula:

```
-----  
? 1 + SQR(2)  
? SQR(3 $\uparrow$ 2 + 4 $\uparrow$ 2)  
-----
```

SQR is one of many mathematical FUNCTIONS built into Commodore BASIC. Others include SIN, COS, TAN, ATN, LOG, EXP, and ABS. If any of these look like they might be useful to you, consult your *User's Guide* or the *Commodore 64 Programmer's Reference Guide* for details.

There's more, of course. But if you need to do math that's more complicated than this you should probably write a program to do it (you'll learn how to do that in the next chapter) or use a calc-type spreadsheet program (Chapter 7). First, let's look at a couple of other things you can have your computer do with simple commands.

POKING AROUND IN MEMORY

Type these and watch what happens:

```
-----  
POKE 53280, 2  
POKE 53281, 15  
-----
```

How did you do that?

You may already know (from Chapter 1) that memory is made up of about 64,000 storage locations called bytes, and that each byte has a unique address so the computer has a way of telling them apart. Some of those memory locations are reserved for special information, like messages to be sent to SID, the Sound

Interface Device that makes your computer produce music and noise; characters to be displayed on the screen; and, of course, the colors of the screen border and background. It so happens that the numerical address of the locations in RAM that are reserved for border color code and screen color code are 53280 and 53281, respectively. The computer repeatedly checks the values stored in those locations to determine what color to paint the screen.

Normally when you talk to the computer in BASIC you have no reason to know or keep track of memory addresses; you just tell the computer what you want it to do and let it worry about the details of storage and retrieval. When you start playing around with special sound and graphics features of your computers, though, you often need to tell the computer to put a number in a particular location by address. That's what you just did with those POKE commands—you POKEd color codes into the slots reserved for screen background and border codes, replacing whatever codes were there before. All 16 color codes (numbered 0 through 15) are listed in the drawing on the opposite page. Experiment with other color schemes until you find one that looks nice with your room's wallpaper.

Or, if you like, you can press the RUN/STOP and RESTORE keys at the same time to get things back to the way they were when you turned the computer on. This key combination is worth remembering, because there are lots of situations where it can help get you out of tight situations. It has the same effect as turning the computer off and back on, except that it doesn't wipe out programs and data in memory. (Sometimes this doesn't work for getting things unstuck; the ON/OFF switch is a last resort.)

Programming a computer would be a tedious business indeed if you had to memorize a five-digit memory address for every piece of information you wanted to store or retrieve in RAM. Fortunately, your computer is particularly good at that kind of tedium, and the BASIC language is designed to take advantage of that skill. In BASIC, most of the memory locations you use can be referred to by names, called VARIABLE NAMES, instead of addresses. We call the memory location a VARIABLE

(because its contents can vary), and the contents of that location the VALUE of that variable.

Next chapter you'll see how easy (and useful) it is to create and work with variables. Now you're going to look at the value of a variable that was created automatically by your computer:

```
-----  
PRINT TIS  
-----
```

TIS is the name given by the computer to a memory location that contains a time clock. 🐾 This PRINT statement is asking the computer to display the value stored in the location labelled TIS. The value in TIS is always the number of hours, minutes, and seconds that have elapsed since the clock was last set, which, in this case, happens to be when you first turned the computer on. (The first two digits represent hours on a 24-hour clock, the middle two are minutes, and the last two seconds.) Every second the computer is on it replaces the value store in TIS with a new time value.

While it can be handy to be able to check how much time has elapsed since you turned the computer on (it's easy to lose track of the time when you're playing with a computer!), you can use TIS for other things as well. For one thing, anytime you want you can set TIS back to zero hour by replacing the string of numbers stored in TIS, whatever they might be, with zeros. Here's how: 🐾

```
-----  
LET TIS = "000000"  
-----
```

You just filled TIS with zeros, so that it could continue its endless count from there. Now when you PRINT TIS you'll see the time elapsed from that instant. So you can use your computer as an accurate, but not very portable, stop watch.

🐾 Nit-picks: The time clock actually takes up six storage locations, but it's easier to think of a variable as a single box for storage. Also, TIS is technically more of a function than a variable, but the distinction isn't important here.

🐾 The reason for the quote marks, and for the \$ for that matter, will become clear next chapter. For now, blind obedience is in order.

Similarly, you can replace the value stored there with the current time of day. For example, if your watch says 2:33PM (which is 14:33 on a 24-hour clock), type

```
-----  
LET T1$ = "143300"  
-----
```

Now, PRINTing T1\$ will give you the exact wall-clock time as long as you leave the computer on, so you can loan your watch to the family jogger without losing track of the time!

Of course, your computer was designed to be more than a calculator/clock. All of this printing and poking we've done hasn't scratched the surface of your computer's power. To really see what makes you machine special, you have to see how it can be programmed. The things you did in this chapter, simple as they were, introduced many of the fundamental concepts of computer programming in BASIC. So even if you never use your computer as a calculator or a clock, you haven't wasted your time here. You just took your first step toward understanding computer programming!

A BIT ABOUT BITS (FOR THE CURIOUS)

By now you know that your computer's memory is made up of thousands of separate storage locations called bytes, each of which can be used to store a peice of information, like the letter "C" that's being displayed on the screen or the color blue that fills the border of the screen. But what is a byte, and how can it possibly contain that letter or that color?

The easiest way to understand the answers to these questions is to remember that computer memory is nothing more than electronic circuitry; think of it as a collection of switches that can be turned on or off. Each of these

(continued)

switches can be used to store a tiny amount of information: a signal to light a light, for example, or the answer to a yes/no question. This small unit of information is called a BIT, for Binary digIT (binary meaning two), because it can represent only two values: on or off, yes or no, zero or one, or anything else you want to call them.

Bits don't exist just in computers; they're the fundamental unit of information everywhere. Remember Paul Revere's famous "One if by land, two if by sea," midnight ride? His co-conspirators used a simple lantern to convey a choice between two messages—a BINARY choice. The lantern communicated one bit's worth of information. While it's theoretically possible to send a message like this with just one lantern, the patriots cautiously used a second lantern to convey the fact that a message was indeed being sent; "one if by land, zero if by sea" wouldn't have worked very well in the Boston night sky. If they had wanted to send a more complex message, they could have used more lanterns. ("Three if by subway!")

In much the same way, computer memory can be used to store bigger chunks of information by treating several bits together as a unit. A byte, then, is a collection of eight bits. If we think of each bit as a light that can be either on or off, then we can make different combinations of lights represent different messages. (Computer scientists usually speak in terms of zeros and ones instead of on and off, but the concept is the same either way.) The computer has an advantage over Paul in that it can see not just the number of lights turned on, but also their ordering, so 01 (off-on) is different than 10 (on-off). It turns out there are exactly 256 unique ordered patterns that can be made out of a string of eight bits, a few of which are shown here.

Since most of us don't find it convenient to read binary numbers—those long strings of zeros and ones—the com-

(continued)

puter allows us to refer instead to the corresponding DECIMAL numbers (the kind we're used to, with 10 different digits instead of two) between zero and 255. Each time it finds us trying to POKE one of these numbers into a memory location, it translates the number into a binary representation, which is the only thing it really understands. That string of binary digits can be read by the computer as a color code, a character code, a part of a number, or almost anything else, depending on where it is in memory and how the program treats it.

So when you POKEd 2 into 53280, you were really telling the computer to set the circuits of that location to the pattern 00000010, the code for red. Another coding scheme, detailed in Appendix E of your User's Guide, is used to translate numbers into characters for screen display. For example, if you POKE 3 (which in binary is 00000011) into location 10240, the computer responds by placing the letter C in the upper-left-hand corner of the screen, which is the spot controlled by that location.

There's a lot more detail where this came from. Thank goodness we have computers to take care of all of this translation for us! If we didn't, it wouldn't be any fun to play with computers at all!

CHAPTER 3

Getting Down To BASIC: A Programming Primer

*One thing at a time,
And done very
well,
Is a very good rule,
As many can tell.
—Mother Goose*

In the last chapter you learned to talk to your computer in BASIC, using the simple command PRINT. You may have also used the POKE and LET commands. Each time you told the computer to PRINT something or POKE something, it responded immediately to your command by doing the one thing you told it to do. Immediacy and single-mindedness are the two distinguishing features of BASIC commands.

BASIC commands are useful for accomplishing very simple, one-time-only tasks. But for most jobs, your computer is a much more effective tool if you tell it what to do with a program. A BASIC program is nothing more than a series of instructions that tells the computer to do something that can't be expressed in a single BASIC command: balance your checkbook, play a game, or whatever. As you'll see in this chapter, instructions like PRINT and POKE are the building blocks BASIC programmers use to create programs to do just such things.

This is another hands-on chapter—you'll get more out of it if you go through at least some of the examples with your computer. The programs in this chapter are short, so you won't have to wear your fingers out typing. Some are useful, some are just for fun. But the main purpose of these examples is to show you the fundamentals of computer programming without overwhelming you with details.

There's a lot of information in this chapter; don't try to memorize all of it. You'll probably want to take a break or two along

the way. (If you take your breaks when the instructions say to type NEW, you can turn the computer off without losing anything in memory that you'll need later.) By the end of the chapter, you should be able to read and write simple computer programs in BASIC, even if you don't understand every detail. Don't expect, though, to be programming your own video game or word processor by dinner time. The idea is just to give you a feel for programming so you'll be a more knowledgeable program USER. Total mastery takes time; in this chapter you'll simply be establishing a relationship with your computer.

If total mastery is your goal, you'll find more help in Chapters 4, 5, and 8, and many of the books listed in Appendix A. Or, if you aren't interested in programming, you can learn what you'll need to know about programs from the first few pages of this chapter.

COMMANDS AND STATEMENTS: THE BASICS OF BASIC

Let's start by looking once more at the PRINT command in action. Last chapter you learned how to use PRINT to display numbers and calculations on the screen. PRINT can also be used to display words, sentences, or just about anything else. To get you started, let's look at a PRINTed word. Turn your computer on and type this, punctuation and all, remembering to hit RETURN at the end of the command (If you make a mistake typing, use DEL to backspace to the error):

```
-----  
PRINT "HELLO!"  
-----
```

As before, you're telling the computer to PRINT something on the screen immediately. This time, though, you're telling the computer to print a group of characters (H, E, L, L, O, and !),

called a *string*. A string may contain any number of letters, digits, special characters, or even blanks. Everything between the quotation marks is displayed exactly the way you typed it. (Notice that the quotation marks weren't printed.)

(WARNING: When you're working with strings, the cursor-move keys don't always work the way you'd expect them to. For example, if you typed YE instead of HE in the example above, and you tried to correct it by moving the cursor back to the Y with the CURSOR-LEFT key, you'd just add strange-looking graphic characters to the string, for reasons that needn't concern us now. This silliness can happen even if the cursor isn't inside a string; if the computer only thinks it's inside one. (It gets confused sometimes when it sees quotes.) If you want to move the cursor backward in this situation, you have to either use the DEL key (which will erase characters as it goes), or terminate the string (real or imaginary) with another quotation mark or a RETURN, and THEN move the cursor).

Running your first program. In the last example, PRINT was used as a BASIC command: a simple instruction to be acted on immediately, but only once. PRINT can also be used as a *statement* in a *BASIC program* that can be used over and over. Programs are made up of statements, which are distinguished from commands by the STATEMENT NUMBERS in front. PRINT can be either a command or a statement, depending on whether it has such a number. So turn your HELLO! command into a one-statement program by retyping it with the number 10 in front of it, like this:

```
-----  
10 PRINT "HELLO!"  
-----
```

You just typed your first program. Nothing happened, right! All you did was enter the program into memory. It's sitting there right now, waiting for instructions from you. If you want your program to do something, you have to type,

```
-----  
RUN  
-----
```

This BASIC command tells the computer to RUN, or EXECUTE, the program currently stored in memory. Regardless of how it sounds, executing a program does NOT kill it. It's still sitting there in memory, waiting for you to type RUN again, or change it, or do something else.

(If the computer responded ?SYNTAX ERROR IN 10 when you typed RUN, it's because you didn't type line 10 exactly as shown here. You can fix your error by retyping the whole line and RETURNing. This replaces line 10 in the memory with a new version, which can now be RUN. Or you can cursor up to the line, edit it (a la Chapter 4), and press RETURN to signal the computer to look at that line again. The cursor should then be sitting on the line that says RUN, so you can just press RETURN again to tell the computer to run it again. If, on the other hand, when you typed RUN the computer responded with just SYNTAX ERROR, you probably misspelled RUN. Try again.)

This one-statement program isn't very exciting. So let's spice it up with some more statements. Each statement must have a unique whole number between 0 and 63999, so 20 and 30 will do. Type this, RETURNing after each line:

```
-----  
30 PRINT "WHEN YOU TOUCH ME LIKE THAT!"  
20 PRINT "I LOVE IT"  
RUN  
-----
```

Your computer should respond by displaying

```
-----  
HELLO!  
I LOVE IT  
WHEN YOU TOUCH ME LIKE THAT!  
-----
```

Notice how all three of the strings you typed in were printed back out, but not in the same order you typed them in. That's because of the First Basic Rule of Law and Order:



THE FIRST BASIC RULE OF LAW AND ORDER:

When a program is run, the statements are executed one at a time, in numerical order, until all statements have been executed.

Before you take this rule too seriously, consider the Second Basic Rule of Law and Order:

THE SECOND BASIC RULE OF LAW AND ORDER:

There are ways of getting around the first rule.

Listing your program. The easiest way to see this is by modifying the program that's currently in memory, which we can see by typing the BASIC command.

```
-----  
LIST  
-----
```

LIST displays the program on the screen, with the lines arranged in numerical order for convenience:

```
-----  
10 PRINT "HELLO!"  
20 PRINT "I LOVE IT"  
30 PRINT "WHEN YOU TOUCH ME LIKE THAT!"  
-----
```

(For future reference, you can use LIST to list just part of a program, as in these examples:

```
-----  
LIST 20  
LIST 20-  
LIST -20  
LIST 20-30  
-----
```

The first of these commands says to list line 20 only (handy if you set a SYNTAX ERROR IN 20 message). The second lists the program from line 20 on, the third lists everything up to and including that line, and the fourth lists lines 20, 30, and everything in between. These variations may seem pointless now, but they're helpful when you're trying to look at a program that's too long to fit on one screen.)

GETTING AROUND . . . AND AROUND, AND AROUND . . .

Your modified program will have the same line 10 and new versions of lines 20 and 30. It's not necessary to move the cursor to change those lines; you can just retype them where you are:

```
20 PRINT "GOODBYE!"  
30 GOTO 10
```

When you type these lines, they replace the old lines 20 and 30 in the computer's memory. Type LIST and see:

```
10 PRINT "HELLO!"  
20 PRINT "GOODBYE!"  
30 GOTO 10
```

Before you run it, look your program over for a minute. In the vernacular, you now have a program with a LOOP structure instead of just a SEQUENCE. When it's executed, the computer will methodically follow the instructions in statements 10, 20, and 30, in order. But the instruction specified in statement 30 is to GOTO (pronounced "go to") statement 10 next and continue from there. Type RUN, and you should see something like this fly by:

```
HELLO!  
GOODBYE!  
HELLO!  
GOODBYE!  
HELLO!  
GOODBYE!  
etc.
```

As you can see, there's a slight problem with this loop program: it doesn't know how to stop. This is what's known in the business as an ENDLESS LOOP, and it's your first example of a program with a BUG, or error, in its logical design. 🐾 Fortunately, there is an emergency exit from this loop: hit the unshifted RUN/STOP key on the left side of the keyboard to tell the computer to STOP what it's doing. Don't be alarmed when it responds with something like BREAK IN 20. You haven't

🐾 In 1945, a computer error in the historic Mark I was traced to a two-inch moth caught in a relay. The technicians joked about "debugging" the system, and the term stuck.

broken anything except the action. And you can start that up again where it left off by typing the BASIC command for CONTInue:

CONT

(Of course, you could always start it over from the top by typing RUN; for this particular program, there's little difference.)



Soon enough you'll see how to exit from loops more gracefully. But grace isn't everything, so let's play with endless loops for a while, and see the power of punctuation and graphic characters in PRINT statements at the same time.

If you're in a hurry, and you have no interest in computer graphics or computer play, feel free to skip the rest of this section and the next. If you aren't interested in programming, this is a good time to exit this chapter and move on to Chapter 4.

LIST your program so you can edit it. Then cursor up to the end of line 10, type a semicolon(;) after the final quote mark, and hit RETURN to signal the computer to replace the old line 10 with this modified one. Now cursor out to the end of line 20 and do the same thing, placing the cursor squarely on line 30. Your program should now look like this:

```
-----  
10 PRINT "HELLO!";  
20 PRINT "GOODBYE!"  
30 GOTO 10  
-----
```

If it does, type RUN. (Be sure to blank out all the other characters on the line before you hit RETURN. If you don't, you'll be telling the computer to RUNGOTO 10. Needless to say, RUNGOTO is not in your computer's vocabulary.) Your program will still type HELLO and GOODBYE! over and over, but this time it will fill every space on the screen with characters. That's because the semicolon in a PRINT statement tells the computer to print the item before it and anything that follows it without leaving any space between them (unless one is a number, in which case it'll leave one blank space).

If the screen seems too cluttered that way, STOP your program (with the RUN/STOP key), LIST it, change the semicolons to commas (don't forget to RETURN after each change), and RUN it again. The commas tell the computer to allow 10 spaces for each item, whether it needs it or not. (If a string is more than 10 characters long, the computer will allow 20, 30, or whatever is needed in a multiple of 10 spaces.) Typists can think of the comma in a PRINT statement like hitting TAB, with tab stops set every ten spaces. (Not to be confused with TAB and SPC, functions that can be included in a PRINT statement to tab over a number of spaces from the left margin or from the current cursor position, respectively.)

For one final variation on this theme, STOP your program, LIST it, and edit the lines 10 and 20 so that it looks like this:

```
10 PRINT
20 PRINT "GOODBYE!"
30 GOTO 10
```

As you'll see when you RUN this program, the PRINT statement on line 10 just "Prints" a blank line each time the program comes around to it; a handy feature for giving your output breathing room.

When you've seen enough, STOP the action on the screen and type

NEW

(Don't forget to RETURN.) This BASIC command clears the old program out of the memory to make room for a new one. So if you now type LIST or RUN, nothing will be listed, because your program is gone.

ENDLESS GRAPHICS (a picturesque diversion)

Before we leave the world of endless loops, let's make some simple moving pictures by printing graphic characters instead of words. Change the text display color to white by pressing CTRL and 2 simultaneously, if you haven't already, and type the following program. Each of the three graphic characters inside the quotation marks is made by pressing two keys at the same time. The combinations are (1) COMMODORE and Q; (2) SHIFT and plus sign (+); and (3) COMMODORE and W. Don't forget the semicolon.

```
10 POKE 53281,0
20 PRINT "┌+┐";
30 GOTO 20
```

RUN the program, and watch the show. As a bonus, because of imperfections in color television displays, you'll probably see more colors than just the black and white that the computer thinks it's displaying. You might want to try experimenting with different background and character colors (see Chapter 2) to see how they affect the pattern. (If it moves too fast for you, you can slow the action down by holding down the CTRL key.)

If you feel adventurous, experiment by changing this program to print other combinations of graphic characters. Suggestions: (1) SHIFT-M, SHIFT-V, and SHIFT-N; (2) COMMODORE-asterisk, SHIFT-pound sign, and COMMODORE-G (3) use your imagination! With some character combinations (e.g., number 2 here) you can get a surprising effect by pressing SHIFT-COMMODORE while the program is running to switch the screen back and forth between character sets.

For even more interesting effects, you can include reverse-video and color-changes in your displays. This can get confusing, so let's start by looking at an example outside of a program. Stop your computer, make sure it's in upper-case graphic mode, and watch the screen while you type the following five key-combinations on a line all by themselves.

COMMODORE-asterisk (*), CTRL-9, COMMODORE-*, SPACE, and CTRL-zero (0). The first COMMODORE-* made a triangle-shape in the top-right half of the character's box (▴). The CTRL-9 turned on reverse-video, reversing the foreground and background colors so that the next COMMODORE-* made a triangle in the bottom-left, instead (▾). (Notice how the two triangles make a rhombus (diamond) shape together.) In the reverse-video mode, the SPACE looks like a white box. CTRL-0 turns the reverse-video off, so that the next characters you type will look normal.

In that example, the reverse-video on and off commands took effect immediately when you typed CTRL-9 and CTRL-0, respectively. But if you type those keystrokes as part of a PRINTed string in a program, they won't have an effect until the program is run. Try it: LIST the program in the memory, cursor up to

the first quotation mark on line 20, retype the quotation mark, type those five key-combinations again (COMMODORE-*, CTRL-9, COMMODORE-*, SPACE, and CTRL-0), and type another quotation mark, a semicolon, and RETURN. This time, the CTRL-9 and CTRL-0 look like strange characters in the string. That's because you typed a quotation mark first, signaling the computer that everything that follows (until the next quotation mark or RETURN is typed) is part of a string. If your program listing looks like the one shown here, RUN it and see the effect of including the reverse-video characters in your program.

```
-----  
10 POKE 53281,0  
20 PRINT " | + | "  
30 GOTO 20  
-----
```

If this isn't spectacular enough for you, you can easily make this (or any) picture multi-colored by inserting two or more of the color-change key combinations (CTRL or COMMODORE with a number key between 1 and 8). For example, edit line 20 again, starting with the quote, so that the string now contains COMMODORE-*, CTRL-9, COMMODORE-*, CTRL-3, SPACE, CTRL-2, and CTRL-0 (close quote, semicolon, and RETURN, of course). Then RUN for fun!

Exercise your creativity. The possibilities for variety in this simple three-statement pattern-generating program put Rubik's Cube to shame!

USING VARIABLES FOR FLEXIBILITY

Last chapter you saw how T1\$ could be used to refer to a portion of the computer's memory that contains a clock. T1\$ is an example of a BASIC variable: a named memory area whose contents can be examined or changed by the program. T1\$ is a special variable in that it always contains a clock value; the designer of your computer decreed it so. By contrast, most BASIC variables are created by the programmer, who makes all of the decisions about what they will contain and how they will be used.

There are two main types of variables in BASIC, *numeric variables* and *string variables*. The easiest way to understand how they work is to think of them as boxes inside the computer, each of which can only contain one kind of value. String variables can only contain strings of text characters, like "HELLO!" and "XYZ\$123", and numeric variables can only contain numbers, like 44 and 123.456. 🐾

You can tell the type of a variable by its name. In BASIC, a numeric variable name must be a single letter (like Q or A) or a letter followed by either a letter or a number (e.g., Q1, HA, or MM). A string variable name is just like a numeric variable name except that it must be followed by a \$ (e.g., Q\$, NA\$, M1\$). Actually, variable names can contain more than two letters or characters, but the computer ignores all but the first two, so it wouldn't know the difference between a variable called TOTAL and one called TOP. Variable names musn't contain BASIC keywords like GOTO, because that would confuse the computer. With the exception of a few special built-in variables like TI\$, variable names are chosen by the programmer, following these rules.

In the last chapter you learned how to use the LET command to assign a value to the variable TI\$. LET works the same way for any variable, as you can see in this next program. If you still have a program in the memory from last section, type NEW (RETURN) to erase it so you can start fresh. Then type in this program, being especially careful about punctuation. (If you despise typing, you can leave out the keyword LET wherever it appears; the computer will know what to do when it sees the equal sign. Also, statement 50 won't fit on one line; just let it wrap around.)

🐾 In Commodore BASIC there are also *Integer Variables*, which can only contain whole numbers, but they're only important when speed is a concern, so they won't be discussed here.)

```
-----  
10 LET N1 = 5  
20 LET N1 = 3  
30 LET N2 = 7  
40 LET A = (N1 + N2)/2  
50 PRINT "THE AVERAGE OF"; N1; "AND"; N2; "IS"; A  
LIST  
RUN  
-----
```

When you RUN this program, the computer responds to line 10 by creating a box labeled N1, and placing the number 5 in it. At line 20, it places a 3 in the same box, completely oblivious to the fact that it's wiping out the 5 that was there before (there's only room for one number in each box). Line 30 tells it to create another numeric variable box labeled N2 containing the value 7. To execute statement 40, the computer must first calculate the value of $(N1 + N2)/2$, using the values currently stored in boxes N1 and N2, and then place the result in a new box called A. Finally, line 50 displays the values of N1, N2, and A on the screen, along with appropriate strings to make it more readable.

```
-----  
THE AVERAGE OF 3 AND 7 IS 5  
-----
```

You now have a program to calculate the average of two numbers and display the result. Each time you want to calculate the average of two new numbers, you can retype lines 20 and 30 and reRUN the program. OR you can add INPUT statements to your program so that it will ASK you to type in new numbers each time you run the program!

INPUT is like the LET statement, in that it causes a value to be placed in a variable box. In the LET statement, that value is built right into the statement, on the right-hand side of the equal sign. The INPUT statement takes the next thing you type on the keyboard as the value to be placed in the box. INPUT statements can be used with either numeric variables or string variables, as these modifications to the program you've been working on will demonstrate (don't type NEW, because you're going to be reusing lines 40 and 50 from the last program.

```
-----  
10 INPUT "WHAT'S YOUR NAME";N$  
15 PRINT "HI,"; N$; ". TELL ME A NUMBER."  
20 INPUT N1  
25 PRINT "TELL ME ANOTHER."  
30 INPUT N2  
LIST  
-----
```

When RUN, the INPUT statement on line 10 first prints a message on the screen prompting you to type your name, and then waits for a response to be typed into the string variable N\$. Everything that you type before the RETURN is put in the box labeled N\$. Line 15 prints a greeting, using the name typed into N\$, and tells you what to type next. Line 20 tells the computer to wait for you to type a number (followed by RETURN), and places the number in box N1. If anything besides a number is typed, the computer will ignore it, respond REDO FROM START, and wait for you to type the number again. Lines 25 and 30 do the same thing for N2, and lines 40 and 50 are the same as before. So a complete run of the program might look like this:

```
-----  
RUN  
WHAT'S YOUR NAME? KERMIT  
HI, KERMIT. TELL ME A NUMBER.  
99.9  
TELL ME ANOTHER.  
33.3  
THE AVERAGE OF 99.9 AND 33.3 IS 66.6  
-----
```

Of course, you can run the program as often as you like using different names and numbers each time.

THE AMAZING FOR/NEXT LOOP

You could use a pencil and paper, a pocket calculator, or a single PRINT command on your computer to calculate the average of two numbers almost as quickly as this program runs. But sup-

pose you're trying to develop a family budget, and you need to figure your average utility costs per month over a twelve month period. For each utility (electricity, gas, water, etc.) you're going to need to take the average of TWELVE numbers. Maybe you're the type who abandons the budget plan at the thought of all of those repetitive calculations. Take heart—your computer has a natural talent for repetitive calculations. And you can easily write a program with a loop to exploit that talent!



Instead of using the GOTO statement to make this next program loop, you'll use two other BASIC statements: FOR and NEXT. These two statements work together to make a loop that will repeat a specific number of times. To see how, type NEW (RETURN) to wipe the old program from memory, and type in this short program:

```
-----  
10 FOR N= 1 TO 7  
20 PRINT N  
30 NEXT N  
40 PRINT "ALL GOOD CHILDREN GO TO HEAVEN!"  
-----
```

The FOR statement is a very powerful statement; it's really several statements in one. In this example, it's saying, roughly: Create a memory box for a variable called N. Place an initial value of 1 in the box. Then execute all of the succeeding statements in this program until the NEXT N statement is encountered. (In this case, there's only statement 20 to execute.) At that point, change the value of N to the next highest whole number value, 2, and execute the statement(s) between FOR and NEXT again. Keep doing this until the N value is higher than 10. At that point, proceed to the statement after NEXT N and continue. Here's what you should see when you run this program:

```
-----  
RUN  
1  
2  
3  
4  
5  
6  
7  
ALL GOOD CHILDREN GO TO HEAVEN!  
-----
```

The FOR/NEXT loop doesn't have to count just by ones. LIST the program, change line 10 so that it looks like this, and RUN it again:

```
-----  
10 FOR N= 1 TO 7 STEP 2  
-----
```

(If you feel like experimenting, try other values in place of the 1, 7, and 2 here, to make the program count by halves, or count backward, or whatever.)

To use the FOR/NEXT loop to calculate an average utility bill, this next program will keep a running total in the same way you would if you were using a calculator to do the job. Even if you don't type it in and run it, you might want to pretend you're a computer and trace through it to see if you understand how it works. (Typing notes: You really don't need to type NEW

here, because you'll be replacing lines 10 through 40 with new ones anyway, but it's a good habit. Also, remember the keyword LET is optional.)

```
-----  
NEW  
10 PRINT "AVERAGE CALCULATOR FOR 12  
NUMBERS"  
20 LET T=0  
30 FOR M=1 TO 12  
40 PRINT "WHAT IS NUMBER";M;  
50 INPUT NO  
60 LET T=T+NO  
70 NEXT M  
80 LET A=T/12  
90 PRINT "THE AVERAGE IS",A  
-----
```

Since this program is a little more complex than any you've seen so far, a line-by-line description might be in order. Line 10 prints out a phrase to tell you what it's going to do. Line 20 sets the running total variable T to 0. Lines 30 and 70 are the beginning and end of the FOR/NEXT loop that will be executed 12 times, once for each month. The first time through, M is assigned a value of 1, so line 40 prompts you to enter the first number at the keyboard. Line 50 places that number in box NO, and line 60 adds it to the current T value of 0, and puts the result right back in T. So if you had typed in 38.25 (without a dollar sign, of course, since numeric variable like NO can't contain special characters), both NO and T would now contain 38.25.

Line 70 increments M to 2 and sends the computer back to the top of the loop (line 30), where it asks you for the second number. If you type in 49.50 this time, that value replaces the old 38.25 in NO. Line 60 tells the computer to calculate the sum of the values of T (38.25) and NO (now 49.50), and store the result back in T. Then line 70 increments M to 3 and starts the whole process again. This process of incrementing M, reading in a new value for NO, adding it to the current value to T, and storing the result in T, is repeated until M reaches 13, at which point the computer moves on to line 80, which calculates the

average and stores it in A. All that's left to do is print the result, which is exactly what line 90 does.

Here's a complete sample run of the program, so you can see exactly what the output will look like:

```
-----  
RUN  
AVERAGE CALCULATOR FOR 12 NUMBERS  
WHAT IS NUMBER 1? 38.25  
WHAT IS NUMBER 2? 49.50  
WHAT IS NUMBER 3? 45  
WHAT IS NUMBER 4? 39.10  
WHAT IS NUMBER 5? 28.88  
WHAT IS NUMBER 6? 29.90  
WHAT IS NUMBER 7? 26.01  
WHAT IS NUMBER 8? 23.11  
WHAT IS NUMBER 9? 22.22  
WHAT IS NUMBER 10? 28.82  
WHAT IS NUMBER 11? 33.33  
WHAT IS NUMBER 12? 44.44  
THE AVERAGE IS 34.0666667  
-----
```

Try running the program using these numbers, or a dozen of your own, or both.

This is a handy program, but it would be handier if it could average ANY number of numbers, rather than being limited to just twelve. You can easily change it so that it starts by asking you how many numbers you want to average, and proceeds accordingly. (If you like challenges, see if you can figure out how before you read on). Here are the necessary modifications:

```
-----  
10 INPUT "HOW MANY NUMBERS TO AVERAGE",Q  
30 FOR M= 1 TO Q  
-----
```

The rest of the program looks, and works, just like before. If you're skeptical, try it and see!

LOOPS WITHIN LOOPS (A COLORFUL DIVERSION)

While your computer is running a BASIC program like that last one, it has to translate and execute each statement. The computer is so fast that you normally don't notice the small amount of time it takes to execute a single statement. But that small amount multiplied by 1,000 is noticeable, as you can see if you watch how long it takes for the second message to appear on the screen after the first one is displayed.

```
-----  
NEW  
10 PRINT "TIME FLIES LIKE AN ARROW..."  
20 FOR T=1,TO 1000  
30 NEXT T  
40 PRINT "FRUIT FLIES LIKE BANANAS!"  
RUN  
-----
```

Since there are no statements between FOR and NEXT in this program, the loop is in effect just telling the computer to count to 1,000 before it proceeds with the program. Which wouldn't take long, except that the computer has to figure out what statements 20 and 30 mean EACH TIME it loops past them. This may seem kind of dumb, but it makes for a handy timing mechanism. You can use this loop as a delaying tactic anytime you want some time to elapse between two or more events. You can even NEST it inside another loop, if you want something to happen at regularly-spaced intervals.

For example, this next program causes the screen background to change colors every 1,000 counts until all 16 colors have been displayed. (You don't need to type lines 20 and 30 of this program, since they're the same as in the last one.)

```
-----  
10 FOR C=0 TO 15  
15 POKE 53281,C  
20 FOR T=1 TO 1000  
30 NEXT T  
40 NEXT C  
RUN  
-----
```


This program uses the POKE command that you first saw in Chapter 2, this time as a statement. Recall that when you POKE a numeric code between 0 and 15 into memory location 53821, the computer responds by changing the screen background to the color corresponding to that code. This program

CHANGING SCREEN COLOR



repeatedly POKES the value of the variable C into that location, with C assuming each successive value between 0 and 15 because of the FOR statement at line 10. Each time through that FOR loop, the computer sets the screen color using the value of C as its color code, and then pauses for a while to go through the empty loop at lines 20 and 30, giving you time to see the color before it changes to another the next time through the outer loop. Try running the program to see how it works.

If you want to see all possible combinations of screen border and background colors, you can put this whole program inside yet another loop (getting dizzy yet?) along with a POKE statement to change the border color. Here are the necessary modifications:

```
-----  
5 FOR B=0 TO 15  
60 POKE 53280, B  
70 NEXT B  
LIST  
-----
```

The idea here is the same: B (the border color code) is set to 0, that value is POKEd into the appropriate location, and the border turns black. It stays that way while the screen tries on all 16 colors, and then the whole thing starts again with B equal to 2. And so it goes, until B has gone through all 16 codes.

There's potential here for a nice light show, if we can remove the text from the screen and make the program continue indefinitely. You may remember that pressing SHIFT-CLR/HOME clears the screen. If you type SHIFT-CLR/HOME inside quotation marks in a PRINT statement, you'll see a reversed heart character in the string. But when that statement is RUN as part of a program, the screen clears. To make the program run indefinitely, we just need to put a GOTO at the end of the program to loop it around to the beginning to start over. Here are the changes (you can also change 1000 to some other number if you want slower or faster action):

```
-----  
1 PRINT "♥"  
80 GOTO 5  
LIST  
-----
```

Before you RUN this program, look it over for a minute. You're looking at a program that contains loops nested four-deep: a loop within a loop within a loop within a loop!

MAKING BASIC DECISIONS

So far you've seen how you can make your computer do calculations, move things around in memory, accept input values, and display output. There's only one other basic ability that it needs to make it full-fledged computer: it needs to be able to make decisions. You can try this program to see if your computer passes the final test. (RUN it twice, lying about your age once.)

```
-----  
NEW  
10 INPUT "HOW OLD ARE YOU";A  
20 PRINT "DON'T FORGET TO VOTE"  
30 IF A<18 THEN PRINT "WHEN YOU'RE 18!"  
40 PRINT "(THIS IS A PUBLIC-SERVICE  
ANNOUNCEMENT)"  
RUN  
-----
```

This simple program demonstrates one of the most powerful of all BASIC statements: the IF/THEN statement. Line 10 asks your age, and stores it in A. Line 20 then prints a string. Line 30 tells the computer to print the second string ONLY if the value stored in A is less than 18. If A has a value greater than 18, the computer skips the rest of line 30 and moves right along to line 40. Compare these two runs of the program:

```
-----  
RUN  
HOW OLD ARE YOU? 93  
DON'T FORGET TO VOTE  
(THIS IS A PUBLIC-SERVICE ANNOUNCEMENT)  
-----
```

```
-----  
RUN  
HOW OLD ARE YOU? 5  
DON'T FORGET TO VOTE  
WHEN YOU'RE 18!  
(THIS IS A PUBLIC-SERVICE ANNOUNCEMENT)  
-----
```

In general, the IF/THEN statement reads like this:

IF this is true THEN do this.

The IF part can check for several different conditions, illustrated in these partial examples:

IF B = 0 THEN. . .	(if B is equal to 0)
IF N > M	(if N is greater
THEN. . .	than M)
IF Y >= 1984	(greater than or
THEN. . .	equal to)
IF Q <= R	(less than or equal
THEN. . .	to)
IF A <> 5	(not equal to)
THEN. . .	

It can also check for two things at once, using the key words AND and OR. Here are some examples that include string comparisons:

```
IF A >= 18 AND SX$ = "MALE" THEN . . .
IF S$ = "HEARTS" OR S$ = "DIAMONDS" THEN . . .
```

Then what? If the part between the IF and the THEN is true, then the statement after THEN is executed. This isn't the whole story, as you'll soon see, but it's enough for this next program, which is a simple drill of the multiplication tables:

```
NEW
100 PRINT "MULTIPLICATION DRILL:"
110 INPUT "WHICH NUMBER DO YOU WANT TO
PRACTICE";M
120 FOR N= 1 TO 9
130 PRINT "WHAT'S";M;"TIMES";N;
140 INPUT A
150 IF A=M*N THEN PRINT "RIGHT!"
160 NEXT N
```

This program is fairly straightforward. It announces its purpose, asks which number you want to practice with, and stores the number in the variable labeled M. It then runs through values of N from 1 to 9, asking you to multiply each of these by M and type in the result, which is stored in A. If the value of A is, in fact, equal to $M * N$, the computer prints RIGHT! and goes on to the next N. If not, it says nothing and simply moves on to the next N. Here's a sample run:

```
-----  
RUN  
MULTIPLICATION DRILL:  
WHICH NUMBER DO YOU WANT TO PRACTICE? 9  
WHAT'S 9 TIMES 1? 9  
RIGHT!  
WHAT'S 9 TIMES 2? 18  
RIGHT!  
WHAT'S 9 TIMES 3? 28  
WHAT'S 9 TIMES 4? 36  
RIGHT!  
WHAT'S 9 TIMES 5? 45  
RIGHT!  
WHAT'S 9 TIMES 6? 52  
WHAT'S 9 TIMES 7? 62  
WHAT'S 9 TIMES 8? 70  
WHAT'S 9 TIMES 9? 99  
-----
```

Educational theorists wouldn't give any prizes for this program. For one thing, it doesn't tell the correct answer when you make a mistake. Here's a solution to that problem:

```
-----  
155 IF A <>M*N THEN PRINT "THE CORRECT AN-  
SWER IS"; M*N  
-----
```

Better still, the program should let you try again if you answer incorrectly. So an incorrect answer should cause the computer to branch back to line 130, making a little loop inside the bigger loop. Here are the changes:

```
-----  
145 IF A<>M*N THEN 130  
150 PRINT "RIGHT!"  
155  
-----
```

If you list the program it should look like this:

```
-----  
100 PRINT "MULTIPLICATION DRILL:"  
110 INPUT "WHICH NUMBER DO YOU WANT TO  
PRACTICE";M  
120 FOR N= 1 TO 9  
130 PRINT "WHAT'S";M;"TIMES";N;  
140 INPUT A  
145 IF A<>M*N THEN 130  
150 PRINT "RIGHT!"  
160 NEXT N  
-----
```

Notice how line 145 doesn't need to say THEN GOTO 130, but simply THEN 130. The effect is still to tell the computer to go back to line 130 and continue from there if A does not equal M*N. This loop is similar to the loops we made with GOTO statements at the beginning of this chapter, except that it isn't endless. The computer leaves the loop and goes on to line 150 as soon as it's given a value of A that will satisfy the test on line 145. Since that's the only way it can ever get to line 150, there's no need for the IF-test we had there before, and there's no need for line 155, either. Loops made out of IF's and GOTO's like this are very common—and very handy—in BASIC. But you have to be careful when you use them, because they're also very bug-prone.

As it stands, this program still won't threaten any teachers' jobs, but it illustrates one use of the IF/THEN statement. Before we look at another sample program, there's something else you should know about Commodore BASIC: the truth is that several statements can be included on one line, with one statement number, as long as they are separated by colons (:). An example:

```
-----  
10 FOR I= 1 TO 1000: PRINT"*";: NEXT I  
-----
```

In spite of the fact that it can make a program harder to read, this kind of statement packing is traditional among microcomputer programmers because it saves memory space. But saving memory space is only important if you run out, and with 64K of built-in RAM, the Commodore 64 has memory to spare for most home applications. So memory conservation at the expense of readability is a questionable trade-off.

But multiple-statement lines can be very handy with IF/THEN statements, because the computer will do every statement on the line after the THEN if the IF part is true. 🐾 This makes it possible to have several things done depending on the outcome of a single test. The next program will show you how.

This program will calculate gross pay (P), including overtime, when given the number of hours worked (H) and the hourly wage (W). If H is less than or equal to 40, the calculation is simply $P = W * H$. If H is greater than 40, then P is $40 * W$ (to cover the first 40 hours) PLUS time-and-a-half for all hours worked over 40. Here's the listing:

```
-----  
NEW  
110 PRINT "PAY CALCULATOR."  
120 INPUT "HOURS WORKED"; H  
130 INPUT "HOURLY WAGE"; W  
140 IF H <= 40 THEN LET P = H * W: GOTO 200  
150 LET P = 40 * W + (H - 40) * W * 1.5  
200 PRINT "TOTAL PAY = $"; P  
-----
```

The important thing in this program is line 140. At that point the computer decides whether the value stored in H is less than or equal to 40. If so, it calculates P according to the short formula on that same line, and then skips to line 200. Otherwise, it ignores the rest of line 140 and moves on to line 150 to do the pay-with-overtime calculation.

🐾 In this context, a line can be up to 80 characters long, as long as it contains no carriage returns.

To see the value of the multi-statement IF/THEN, consider what would happen if that GOTO statement weren't tacked on to the end of line 140, and you were calculating wages for a week without overtime. When H passes the test at line 140, the computer calculates P on the same line, and then goes on to the next line in the program, where it RECALCULATES P USING THE WRONG FORMULA! Of course, this bug could be eliminated without statement packing. But believe me, it's easier to do this kind of thing when you can put several statements on a line. Or better yet, try it both ways and see for yourself!

As a final trick, let's rewrite this program so we can calculate gross wages for any number of employees. It's easy to make it loop back from the end to the beginning (can you figure out how?), but we'll also need a way to terminate the loop when we're through. Three more statements should do it:

```
-----  
310 GOTO 120  
120 INPUT "HOURS WORKED (0 WHEN DONE)";H  
125 IF H=0 THEN END  
LIST  
-----
```

Line 310 makes the loop; line 125 stops it. Line 120 is changed to give more instructions. This modified program will keep calculating paychecks until you enter a zero for hours worked. The zero-value for H will pass the IF-test on line 125, causing the program to execute the statement after the THEN. That statement tells the computer to do just what it says: END the execution of the program immediately. Here's a sample run:

```
-----  
RUN  
PAY CALCULATOR.  
HOURS WORKED (0 WHEN DONE)? 40  
HOURLY WAGE? 5.00  
TOTAL PAY = $200  
HOURS WORKED (0 WHEN DONE)? 50  
HOURLY WAGE? 5  
TOTAL PAY = $275  
HOURS WORKED (0 WHEN DONE)? 0  
-----
```


That's enough programming for now. You've seen examples of all of the major operations that computers can do, and you've seen programs that combine these operations in a variety of ways. Of course, there's a lot you haven't seen. But the most important concepts were here; most of the rest is just detail.

There'll be other programs in this book, but most of them are long enough that you'll not want to have to type them in every time you want to use them. So next chapter will give you the information you need to save programs on disk or tape, among other things.

Before you move on, though, pat yourself on the back. You're a programmer now!

BEHIND BASIC: WHAT REALLY HAPPENS (for the curious)

Your computer doesn't REALLY understand you when you tell it to PRINT, POKE, or GOTO. The 6510 chip that serves as the computer's brain is capable of understanding instructions only in 6510 *Machine Language*. This Machine language doesn't look very much like a language to most humans; when displayed in its purest form, it's just a string of zeros and ones, representing patterns of bits, or on-off switches, inside the computer.

It's possible to actually write programs directly in machine language. But it's a tedious process at best, in part because of the readability problem. (This problem can to a large extent be solved by using *Assembly Language*, which is discussed in Chapter 8.) But most programmers write programs in high-level language, like BASIC, which is easier for humans to read and write.

How is it that the computer responds to BASIC commands, if it only understands machine language? It's all possible because of a couple of programs that reside permanently in

(continued)

ROM (read-only memory — remember?). One of these, the *operating system*, starts running as soon as you turn the computer on, and does all sorts of housekeeping and organizational tasks: organizing the memory, checking the characters you type and displaying them on the screen, and so on. When you type a line, the operating system calls on another program, the BASIC *interpreter*, to interpret, or translate, that line into machine language. If the interpreter can't find what you typed in its phrase book, it tells you that you have made a SYNTAX ERROR; otherwise it generates the machine-language instructions that correspond to your command.

Because machine language is primitive, a typical BASIC command translates into several machine language instructions. For example, the BASIC command

```
-----  
LET X = Y + Z  
-----
```

would translate into separate machine instructions to do the following: (1) load the first number from memory (the location with the symbolic name Y) into the *accumulator*, where arithmetic operations are evaluated, (2) add the second number in memory (location Z) to the contents of the accumulator, and (3) move the new contents of the accumulator to somewhere in memory (location X). In addition, each of those variable names would have to be translated into numeric addresses before the command could be processed. (There's even more; to do a simple operation like adding two numbers, the microprocessor has to combine them ONE BIT AT A TIME, following a set of rules called BOOLEAN ALGEBRA. But that's another story.)

All of this translation and looking up takes time, so BASIC programs don't run as fast as machine language programs. But they're certainly easier to write! Chapter 8 explores both these languages in a little more detail, along with others that offer some of the advantages of each.

CHAPTER 4

A Peripheral User's Guide

If you're going to put all your eggs into one basket, you'd better watch the basket.

—Mark Twain

In the last chapter, you typed a few full-fledged BASIC programs into your computer's memory. The programs were intentionally kept short to save your fingers. But even for short programs, typing takes time, and it's not much fun. Computers would not be much help if we had to type in every program each time we used it.

Canned Programs. Fortunately, you don't. For one thing, you can buy ready-to-run software in cartridges, on disk, or on tape. But how do you get it into your machine? If it's a cartridge, you just plug it into the back of the computer before you power up. When you're through with it, you slide it out **AFTER** the computer is turned off. A cartridge is nothing more than Read-Only-Memory with a program permanently etched in. So when you turn the computer on, the program is already there, waiting to do your bidding. Besides being instant, a ROM cartridge is relatively indestructible. About the only things you can do to ruin a cartridge are to plug it in or pull it out of the computer while the computer is turned on. (The flip side is that you can't make changes in a cartridge program, even if you want to.)

Not all commercial programs come in cartridge form, though; most are sold on diskettes and cassettes. Programs stored on these magnetic media have to be **LOADed** (copied) into the computer's memory before they can be run. This takes time (especially with tape), but it isn't hard. It's usually just a matter of issuing a BASIC command or two and waiting while the disk drive or datasette does the job. Commercial programs generally come with exact instructions for loading.

This chapter will teach you the commands that are generally used for loading programs from those peripheral devices into memory. You'll learn how to SAVE copies of programs on disk and tape, so that you can have a backup in case something happens to the original. These commands will also work for saving programs that you type yourself, so that you can recall and reuse them later without retyping them. The chapter closes with a short lesson in using the printer to make hard (paper) copies of program listings and such.

You won't learn everything there is to know about these peripheral devices from this chapter. You will learn the procedures you'll need to use them in day-to-day home operations. You'll know enough to cut through the jargon in the user's manuals that come with these devices when you need more information.

Of course, you'll need more than the computer to use this chapter. If you plan to use tape, you'll need a datasette and at least one cassette. The tape can be just about any brand or quality, but in general the very short tapes made specifically for data recording work best. If you're using diskettes, you'll need a disk drive and at least one 5-1/4-inch, single-sided, single- or double-density, soft-sectored diskette. The section on printers assumes you have a printer connected to your C-64 and an adequate supply of paper.

A SOUND PROGRAM FOR SAVING

The most important thing a beginning computer user needs to know about peripherals is how to load a program from tape or disk into the computer's memory. If you can do this, then you can take advantage of the software that's available on these two media. But in order to load a program from disk or tape into memory, you first have to HAVE a program stored on disk or tape. So you'll start by typing in another short program, and saving it on tape and/or disk. For the fun of it, you can use this program, which will show off your computer's voice.

Meet Sid. Last chapter introduced you to programming,

but you didn't get a chance to meet SID, your computer's sound interface device. Writing BASIC 2.0 programs to use this music synthesizer is no simple task, but it is possible to write short programs to make interesting sounds come out of your TV speaker (or monitor speaker, if it has one; or stereo system, if it's hooked up). This program, for example, simulates the sound that cartoon coyotes make when they fall off of cliffs:



```
-----  
1010 REM FALLING SOUND  
1015 FOR S=54272 TO 54296:POKE S,0: NEXT S  
1020 POKE 54296,15  
1030 POKE 54277,190  
1040 POKE 54278,248  
1050 POKE 54276,33  
1060 FOR F=255 TO 2 STEP -1  
1070 POKE 54273,F: POKE 54272,F  
1080 NEXT F  
1090 POKE 54276,128  
-----
```

Here's a shorter, silent-movie version of the program you can type instead if you're in a hurry:

```
-----  
1010 REM FALLING SOUND  
1020 PRINT "WHISTLE THUD"  
-----
```

The first program here is full of POKEs, which are impossible to understand without a map or description. Each one of those POKEs is sticking a value into a memory location controlling some aspect of the sound chip: line 1015 puts zeros in all of the sound chip locations to clear things out; line 1020 sets the volume; line 1030 sets the attack/decay rate; line 1040 defines the sustain/release; line 1050 chooses the waveform; the statements in line 1070 establish the frequency, or pitch (which goes down because of the FOR/NEXT loop); and line 1090 turns the note off with a percussive bounce.

Still confused, right? Don't worry about it; you aren't here to learn music synthesis, anyway. (See Chapter 8 for a shortcut if you are.) You just need a program to SAVE; you don't need to understand it.

REMARKS. There is one new statement that you should know about if you're interested in programming, and it's on the very first line of the program. REM stands for REMark, and it doesn't do anything at all! When the computer sees the keyword REM, it ignores everything else on the line. That

means that you can type anything you want there without changing the logic of the program at all. The purpose of the REM statement is to make programs more readable and understandable by humans. You can put REMarks in your program that will help you (or anybody else) to figure out or remember what the program (or part of the program) does, who wrote it, when it was written, and so on. This particular REM statement was included in the listing because the rest of the program contains virtually no clue as to what it does. Since human memories are more volatile than magnetically-encoded memories, REMarks like this are essential if you want to remember what your old programs do.

So type this program in, and RUN it, making sure the volume is turned up on your TV or monitor. When it makes noise for you, you're ready to SAVE it on tape and/or disk.

Filing programs. SAVEing a program is a lot like filing a copy of a letter or an article in a file cabinet for future reference. If you just stuff the letter into the drawer, you probably won't have much luck finding it when you need it later. You want to file it so that you can retrieve it quickly. One way is to put the letter in a manila folder, and put a label on the folder tab that identifies the letter with some unique key words, like "MOM 4/1/84."

Similarly, when you save a program, you should give it a name to file it under so you'll be able to easily find it when you need it. The name can be anything you like, as long as it is no more than 16 characters long—it can even contain numbers, spaces, or periods. Of course, it's best to choose a name that reflects the purpose of the program, so you'll recognize it later. For this program, a good name might be FALLING SOUND. That happens to be the same phrase that you typed into the REM statement at the beginning of the program—a good practice, but not a requirement.

Because of the similarity between saving programs and filing paper documents, a program that is saved on disk or tape is called a PROGRAM FILE. Stored programs are not the only things that are referred to as files. For example, if a program

produces output that goes to the datasette, the disk drive, or the printer, that output is called a DATA FILE. There are many similarities between the ways these two types of files are handled, but our immediate concern is with saving a program as a file.

If you plan to save the program on tape, the next section will show you how. If you're using disk, read the sections after that. The last part of this chapter deals with printers; it's there in case you're interested.

SAVING AND RECALLING PROGRAMS ON TAPE

Getting to know the datasette. If you have experience with a standard audio cassette recorder, you should have no trouble at all using the datasette as a storage medium for your programs. You'll just need to learn three new BASIC commands that the computer uses to control the datasette.

So start where you are: you have a program that you want to save on tape. (It doesn't matter which of the two versions of the falling sound program you typed in.) First, you'll need to open the datasette by pressing the button marked EJECT. Then place a blank tape cassette in it, with the exposed tape facing you (don't touch the tape when you're handling the cassette!), and close the lid. If you're using a new cassette, fast-forward it to the end (by pressing the button marked F.FWD) and then rewind it back to the beginning (REW), to make sure it's evenly wound. (This isn't absolutely necessary, but it's a good precaution to take.) In any case, rewind the cassette to the beginning, press the STOP button on the datasette, and push the tape COUNTER reset button to set the counter back to zero. If your tape has a leader on it (a short length of non-magnetizable tape at the beginning, usually white or transparent as opposed to brown), you should probably do a quick fast-forward past it before you try to save your program. (Tape leaders vary in length, but if you go forward 'til the tape counter says 003, and

then hit STOP, you should be past the leader; you can always EJECT the tape by hitting the STOP-EJECT button again and look if you're not sure.)

Saving on tape. Now you're ready to save it on tape, using the name FALLING SOUND as its label:

SAVE "FALLING SOUND"

The computer should respond by telling you to

PRESS PLAY AND RECORD ON TAPE

Do as it says. (On some datasettes it's not necessary to press PLAY when you're recording information—consult your manual or experiment.) The cassette should start moving in the machine and the screen should go blank while the program is being recorded. In just a few seconds (because this is a short program), everything should return to normal, telling you that the program has been recorded. Which is to say that a copy of the program is now on your cassette, just like the one you typed into the computer's memory. That one is still in memory, as it was before the SAVE. RUN it and see.

Now that you've saved your program on tape, do it again, without rewinding, using a new name:

SAVE "FALLING SOUND 2"

Why do it twice? Datasettes are not known for their reliability, so it's wise to make two copies of any program you want to save. 🐾 (For maximum security of long or important programs, you should put a DIFFERENT cassette in the machine and SAVE a backup copy there, too. This protects you against faulty or misplaced tape as well.) After the second copy has been

🐾Technically, you already have two copies stored on tape; the datasette always makes two copies when you ask it to save something so it can check for recording errors. But experience says that's not enough of a backup.

recorded, make a note of the number on the tape counter and press STOP on the datasette to release it from RECORD mode. (Pressing the same button again EJECTs the tape.)

Verify. You've now got two copies of the program on tape, but how do you know that you've got a good one? One way is to rewind the tape and type

```
-----  
VERIFY "FALLING SOUND"  
-----
```

This BASIC command tells the computer to locate and read the program on the tape called FALLING SOUND, and compare it to the one currently stored in the memory. (As before, the screen will go blank while the tape is being read.) If they match exactly, it will say that everything's OK. If not, the computer will tell you that you have a VERIFY ERROR. In which case, you can simply repeat the command, with the program name FALLING SOUND 2, to see if that matches. If it doesn't do either, you might have a bum cassette. Try saving the program again on a fresh one to see if that helps. If not, you may have a problem with your datasette.

Assuming that you successfully managed to save and verify a copy of the program, you don't need to keep it in the computer's memory any longer. If you think of the memory as a workplace like a desktop, you are now free to clear the workspace (by typing NEW), or load another program from disk or tape, or turn everything off and walk away, knowing that anytime you want you can resurrect the old program from tape. How?

Loading from tape. First you'll need to REWIND the tape to the beginning, where you stored the program, and press stop when it stops. Then type the BASIC command for loading a program from tape into memory:

```
-----  
LOAD "FALLING SOUND"  
-----
```

The computer will respond:

```
-----  
PRESS PLAY ON TAPE  
-----
```

Do it, and watch the screen go blank and the cassette turn as the computer searches for the program. When it finds it, it will announce

FOUND FALLING SOUND

If you find yourself waiting a lot longer than it took to save the program in the first place, it may have missed the first copy you made. If that happened, it should find FALLING SOUND 2, instead. But if it misses both copies, it will keep searching to the end of the tape unless you stop it by pressing the computer's RUN/STOP key. Then rewind the tape and try again.

Once the computer has found the program you're looking for on the tape, you may need to give it the signal to load the program into memory. So press the COMMODORE key to load it, and wait for the screen text to reappear and the tape to stop. (On some C-64s, the program will load automatically after 15 seconds even if you don't press the COMMODORE key.)

If everything went OK, you should have the original program back in memory. Type RUN and see. If your computer doesn't whistle, and you don't see the original program when you type LIST, something may be wrong with the cassette or the datasette. Then again, it may be that things just didn't work right that time. Rewind and try again before you panic. If you're going to use cassettes for your storage medium, you have to be prepared for that kind of thing.

The steps for loading a program are the same whether the program was created by you or by professional programmers—the only thing that's different is the program name. And as you'll soon learn, that's not even necessary! (Some commercial programs may have special loading instructions that are slightly different than these: when in doubt, read the directions.)

More tape tricks and tips. You know the basics now, but there are a few other things you want to try with your datasette. Here's one: rewind the cassette to the beginning, press STOP, and type:

LOAD "FALLING SOUND 2"

The computer will respond just like it did before, telling you which buttons to push, looking for a program on the tape, and telling you that it's:

FOUND FALLING SOUND 2

But since FALLING SOUND wasn't what you asked for, it will keep looking instead of loading when you press the COM-MODORE key. Eventually it should proudly announce:

FOUND FALLING SOUND 2

Since that's what you told it to LOAD, pressing the COM-MODORE key will tell it to complete the loading process, as before. Of course, you can abort the process by pressing RUN/STOP instead.

It's worth noting that you should be extremely careful with spelling when you tell the computer to load a program. If you had typed FALLING instead of FALLING in that last example, the computer would have glided right by the program you were looking for on the tape, as you stood helplessly by and watched. If this ever happens to you, hit RUN/STOP, rewind the data-sette just a smidgen, and retype the load command.

If your typing is terrible and your spelling is sporadic, there's still hope. Rewind the tape, and type:

LOAD

You're telling the computer that you want to load the first program stored on the tape, whatever it is. (Hopefully it will find the first program on the tape!) Except for not worrying about the program name, the computer will respond just like it did the first time.

Here's another trick for the I-don't-like-to-type crowd: rewind and STOP the datasette as usual, and then simply press SHIFT and RUN/STOP. This is an abbreviated way of telling the computer to locate the first program on the tape, load it into memory, and run it, all without further human intervention!

That's handy, especially for children and computerphobics who don't like to memorize long lists of commands to type. But there are lots of things about cassette storage that are anything but handy. Most of these were discussed in Chapter 1, so I won't go through them again here. What I will do is list a few tips for minimizing the hassles of cassettes:

- Use the shortest tapes you can find (C-10 is a good length), and don't store too many programs on one tape. Searching through a long tape for a particular program can be mighty frustrating, and tape is cheap.
- If you have one or more programs stored on tape, and you want to store your current program on the same tape, you can use the VERIFY command, with the name of the last program on tape, to position the tape head after the end of that program. Of course, the computer will tell you that you have a VERIFY ERROR, because the program in memory doesn't match the one on the tape. But no matter; the tape is positioned perfectly for adding the new program with a SAVE command.
- Even better: turn the tape over and store the program on the other side. You'll be able to find it faster later.
- Keep a log for each tape telling what is stored on the tape, and the approximate tape counter index of each program. The tape counter isn't 100 percent accurate, but it's better than no record at all.
- A tape can be reused when the programs on it are no longer needed. When you SAVE a program onto tape, the datasette automatically erases anything that was previously recorded on that section of tape. If you want to erase a tape without recording anything new on it, turn the computer on, put the cassette in the datasette, press PLAY and RECORD, and wait for the tape to run through to the end.

- If a cassette contains important programs, protect it from accidental erasure by breaking out the write-permit tabs on the back edge of the cassette. When these tabs are removed (with a pen, a chopstick, or a pointy fingernail), you can't accidentally record over the program, because you can't record on the tape at all. (If you later change your mind and decide you want to record something new on the tape, you can always replace the tabs with cellophane tape patches). Each tab controls one side of the tape—when you're looking at the tabs, the one on the right is for the side that's facing up.
- Save at least two copies of your favorite programs on different tapes; think of it as cheap insurance.
- Keep your tapes away from anything with a strong magnetic field: large electric motors, ringing telephones, subways, etc.
- Make sure that all of the buttons on the datasette are disengaged before you leave your computer; the rubber wheel that holds the tape in place can be bent out of shape if it's left pressed against other parts for long periods of time.
- After every few hours of use, clean and demagnetize the heads of your datasette, just as you would an audio cassette recorder. Cotton swabs and rubbing alcohol, or one of the commercially available cleaning products, should be used for cleaning. Ask your computer or audio dealer about head demagnetizers.
- Buy a disk drive, and move on to the next section!

SAVING AND RECALLING PROGRAMS ON DISK

Getting ready. Disk storage is far and away faster, more convenient, and more reliable than tape storage. But disk storage does require a little bit more work on your part before you can store your first program. Here's a rundown.



First you'll need to turn your disk drive on, using the switch on the back (assuming you've already plugged it in to the power and connected it to the computer). If you've got it plugged into a common switched receptacle with the computer (as discussed in Chapter 1), the disk drive will go on with the computer when you turn on the master switch. Otherwise, you should turn the computer on after the disk drive is on, so that it doesn't get confused later. Either way, when you power up the computer, the green and red lights on the front of the drive should go on while it makes a whirring sound. After a few seconds the whirring should stop and the red light should go off. In general, when the red light is on, the drive is doing something. (Unless it's steadily flashing, like a stoplight, in which case it's trying to tell you that something is amiss. More on that later.) The green light remains on whenever the drive is turned on.

Next, take a new single-sided, single-density, soft-sectored, 5-1/4-inch diskette, slide it out of its protective sleeve, and look

it over. You should be seeing several holes in the stiff paper envelope. The disk drive uses these holes to touch the sensitive disk surface; you shouldn't! You'll also find a square notch cut into one edge of this envelope. That's called a write-protect notch, and it provides insurance against accidental erasure of important programs. If the write-protect notch is covered by tape, the disk can't be written on by the drive. That makes you a lot less likely to accidentally write over the top of your favorite program on the diskette. So if the write-protect notch on your diskette is covered, you probably should be using another disk for this exercise.

Here's how to insert your disk in the disk drive: Gently press on the disk drive's door handle (that tab sticking out in front) to release the catch; the little door should slide easily up and out of the way (unless, of course, it's already open). (If there's already a disk in the drive, it will pop out like toast when you open the door.) Orient the disk so that the label is facing up and toward you, the write-protect notch is on your left, and the long oval hole in the paper envelope is pointing toward the front of the drive. Slide the disk carefully into the drive slot until it stops; it should stay in by itself. Now slide the door down to where you found it, and your disk is ready to go. Well, almost. . . .

A brand-new diskette contains no information at all; just a virgin magnetic surface waiting to be used. But your Commodore disk drive won't let you store any programs or data files on it until you **FORMAT** the disk. If you think of the diskette like a freshly-paved circular parking lot for programs, formatting the disk is a little like painting stripes and stall numbers on the pavement so the parking lot attendant will be able to easily park and unpark the files. 🐾

🐾The parking lot attendant in this case is actually a microcomputer-on-a-chip in the disk drive that coordinates communication between computer and disk and performs miscellaneous housekeeping functions associated with disk operation.

In the process of formatting the disk, you'll give it a name (up to 16 characters long) and an identification code (any combination of two letters and/or numbers). The name is stored along with some miscellaneous information in a special place on the disk called a **HEADER**. The ID is recorded there too, but it's also recorded in each of the "parking stalls" on the disk where information can be stored, so that the disk drive can always quickly tell which disk it's using. (Formatting a **USED** diskette has the effect of making it like a new one; all of the old information recorded on the disk is effectively erased.)

This process of formatting or **NEWING** the disk 🐾 requires three **BASIC** commands that are confusing if you aren't used to them. These commands will be explained later; for now just type them in on faith. To format a new disk, with a name **MY FIRST DISK**, and an **ID-A1**, start by typing

```
-----  
OPEN 15, 8, 15  
-----
```

If everything's OK, the computer will respond **READY**. If the disk drive isn't turned on, or it's not correctly connected to the computer or the disk isn't properly seated in the drive, the computer will say something like **DEVICE NOT PRESENT ERROR**. Check the connections, check the switch, and try popping the disk out and pushing it back in; then try again. If that doesn't work, it sometimes helps to turn the drive and/or the computer off, wait a few seconds, and turn it back on. (If you turn the computer off, though, you'll lose the program in memory!) When you get it to work, type

```
-----  
PRINT# 15,"N:MY FIRST DISK,A1"  
-----
```

The red light on your disk drive should go on and the disk should spin for a minute or so. Don't worry about the scraping and bumping noises or the flicker of the red light; those are all normal when the disk drive is working. When it's done, the

🐾 Computer engineers and sportscasters love to turn nouns and adjectives into verbs!

noises should stop and the red light should go off. If the light flashes at you, the drive is trying to tell you that something didn't work right; run through the troubleshooting steps outlined in the last paragraph and try again. No luck? Try a new diskette. If the same thing happens when you repeat the process with another disk, skip ahead to the end of the next section to find out how to test your drive for problems. When you get an "all clear" signal, type the last command:

```
-----  
CLOSE 15  
-----
```

Saving a program on disk. You're (finally!) ready to save your program on the newly-formatted disk. This part is easy; you just need to tell the computer that you want to save the program on disk with the name FALLING SOUND. This command should start the disk whirring:

```
-----  
SAVE "FALLING SOUND",8  
-----
```

You may have noticed that this is the same BASIC command that you'd use to save the program on tape, except for the 8 tacked onto the end. The computer doesn't care whether it saves your program on disk or tape; it just needs to know which you want. The way you tell it is by specifying a DEVICE NUMBER. For no apparent reason, the device number for the datasette is 1 and the device number for the disk drive is 8. So this command is saying to the computer, "Save the program currently in memory using the name FALLING SOUND, and save it on device 8, the disk drive." (If you don't specify a device number when you use the SAVE command, the computer assumes that you are referring to device number 1, the device number in this command.)

If the computer prints an error message on the screen, or if the disk drive's red light is winking at you, reset the disk and try again. Otherwise, an exact copy of the program in memory should be safely tucked away on your diskette. Disk drives are

so much more reliable than datasettes that most programmers don't bother to use the VERIFY command (explained last section) to check the accuracy of saved files.

But they DO generally make *backup copies* of important programs. This involves nothing more than saving a second copy of the program. Since there's already a program called FALLING SOUND on this disk, you'll have to use a different name for your backup:

```
-----  
SAVE "FALLING SOUND 2",8  
-----
```

(If you REALLY wanted to make sure you had a backup copy of this, or any, program, you'd be better off storing your second copy on ANOTHER disk, to protect you against the possibility of the first disk being lost, mutilated, or just plain defective.)

Loading a program from disk. So now you have two copies of your program on your disk. That means you can type NEW to clear the original from the computer's memory, or load a program from another disk or tape, or turn the computer off without worrying about losing the FALLING SOUND program forever. Because whenever you need it, you can copy it back into memory from disk by putting the right disk in the drive and typing

```
-----  
LOAD "FALLING SOUND",8  
-----
```

Once again, this is the same command you'd use to load a program from tape, except that you're specifying that the program is to be loaded from device number 8 instead of device number 1. You can check to see that it worked by typing LIST or RUN. You'll use this LOAD command over and over with different file names. It's the way you'll load commercially-produced, as well as homemade, programs from the disk.

There are a couple of variations on this command you're likely to see. For example, some programs need to be loaded into different parts of the memory than the BASIC program area. The command to do this looks the same, except that a comma and a 1 are tacked onto the end, like so:

LOAD "CHEESECAKE",8,1

Often programs loaded this way will run as soon as they are loaded without your having to type RUN.

Another variation uses the asterisk (*) in place of the filename:

LOAD "*",8

If the computer has just been turned on, this command will load the **FIRST** program it finds on the disk. If you've already been working with files when you type this, the computer will try to load the file you last referred to in a command.

You can also use the asterisk to replace just part of the file name, like this:

LOAD "FALLING*",8

This command tells the computer to load the first program it finds on the disk with a name that starts with **FALLING**, no matter what the rest of the name is. 🐾 This can not only save typing, but it can also help when you can't remember the whole name of the file. As you'll soon see, there's a less chancy way to deal with that problem.

KEEPING YOUR DISK IN ORDER

You now know the commands you need (1) to save programs you type in from books, magazines or your imagination, and (2) to load those programs, or professionally-produced software, into the computer. That should be enough to get you started with your disk. But there are a few other disk operations that you will likely have to perform from time to time: finding out

🐾 The first one it **FINDS** on the disk isn't necessarily the first one you put there. It's the first one listed in the disk **DIRECTORY**, which will be discussed directly.

what's on a disk, removing old files, changing the names of files, and other general housekeeping operations. This section discusses these procedures, in order of decreasing importance.

The disk directory. Suppose you're interrupted from your computing by a phone call, and you return to discover that you spaniel ate this chapter right out of the book. You've forgotten the names of the files you stored on the disk; how can you find out what they are?

On every formatted disk there's a special file called the DIRECTORY. This directory is like a table of contents of the disk—it's a list of all of the files currently stored on the disk. You can LOAD the directory into memory just like you would load a program, as long as you know what to call it. It goes by the seemingly arbitrary but easy-to-type name of "\$" (because it's so valuable?), so the load command looks like this:

```
-----  
LOAD "$",8  
-----
```

When the disk drive is done spinning, the directory should be loaded into memory, replacing any program that was there before. (So it's important to make sure you've saved the current program before you issue this command!) You can see the directory by typing

```
-----  
LIST  
-----
```

If you've been following along with this chapter, the directory that appears on the screen should look like this:

```
-----  
Ø "MY FIRST DISK" A1 2A  
1 "FALLING SOUND" PRG  
1 "FALLING SOUND 2" PRG  
662 BLOCKS FREE.  
-----
```

The disk name and ID number appear at the top of the directory, followed by a list of files stored on the disk. The abbreviation

PRG to the right of each file name indicates that the file is a program rather than a data file. The number to the left of each file name indicates how much space, in blocks, the program occupies on the disk. A block (in case you're curious) can contain up to 256 bytes, or characters, and a Commodore-formatted disk contains 683 blocks. (Only 664 of those blocks are available for your use; the rest are used by the disk-controller for various housekeeping purposes.) The number at the bottom of the list tells you how much space remains unoccupied on the disk. As you can see, there's room for plenty of programs on a disk!

Here's another tip for lazy typists: If you want to load a program from the disk, and the directory is displayed on the screen, you can move the cursor to the line that contains the name of the program in question, type the word LOAD and a comma, space over the the first quotation marks, cursor-right past the second quotation marks, type another comma and an 8, space out the rest of the line, and RETURN. It sounds complicated, but it's handy if you use long program names.

Opening, Closing and Formatting Revisited.

There are a few other important things you can tell your disk drive to do: erase a file from the disk, change the name of a file on the disk, eliminate incorrectly-stored information from the disk, and get ready to use a different disk. Commodore BASIC 4.0 contains simple commands to perform any of these operations. Unfortunately, the unadorned Commodore 64 contains BASIC 2.0 instead of BASIC 4.0, so doing these things is a little more complicated.

Since you can't instruct the disk drive to do these things directly with a BASIC command, it's necessary to open up another channel of communication. That's effectively what the OPEN command is for. You already used this command when you formatted your new disk; let's look at it again:

OPEN 15,8,15

One way to try to understand this cryptic command is to think of the computer as a corporate executive locked in his office with only a telephone to communicate with all of his underlings. Since our hypothetical boss has a bad memory, all employees are referred to by number rather than name. But even that doesn't help much, because he can't keep track of the employee numbers. He can only remember the telephone line numbers, because those are right in front of him. When the boss wants to talk to somebody, say employee number 8, he picks up the phone and tells his receptionist, "Put number 8 on line 15." Once that's taken care of, he doesn't have to worry about the name or number of the person he's talking to; he just needs to talk to line 15.

So the OPEN command shown here is telling the computer to open a communication line, or LOGICAL FILE, with a number 15, and connect it to an external device number 8, the disk drive. Once this is done, the computer can talk to the disk drive by sending BASIC commands to the logical file number 15. Like our executive, the computer needs to open a communication line (a logical file) anytime it talks to anything in the outside world: the disk drive, the printer, or whatever. (Sometimes this is done automatically, as when you told the disk drive to save a program.)

The first number after the keyword OPEN, then, is the logical file number. (It can be any number between 1 and 127, but 15 is usually used for the kind of things we'll be doing.) The second number, 8, is the device number, and it's chosen based on which device you want the computer to talk to. The OPEN command often just has these two numbers in it, but in this case it has a third. The third number is the *channel number*; 15 is the standard number for the disk drive's command channel—the channel through which the computer sends disk-controlling commands, besides LOAD and SAVE. By including this number; you're telling the computer that you don't want to send just any old information to the disk — you want to send one or more of these special commands.

If you're not confused by now, you're probably not paying attention. But it's OK—this works whether you understand it or not.

After the command channel is open, you're ready to send a command. When you formatted your new disk, the command you sent was (WARNING: don't type this if the disk in the drive contains anything you want to save!)

```
-----  
PRINT# 15, "NEW:MY FIRST DISK,A1"  
-----
```

PRINT# is a BASIC command that works pretty much like PRINT, except that it's used to send messages to files and peripheral devices instead of the screen. (And it can't be abbreviated as ?#.) Since you gave the command channel of the disk drive a file number of 15, and 15 goes after the PRINT# keyword, effectively saying "Send this message through the command channel of the disk drive." (Recall our shut-in executive, who simply talks to a numbered phone line.) The message being sent is in quotation marks after the comma: NEW, followed by the name and the ID to be assigned to the new disk.

The tidy way of doing things is to close whatever you've opened before you move on, so you closed file 15:

```
-----  
CLOSE 15  
-----
```

Any time you need to format a new disk, you can follow this procedure, using a different disk name and ID number. (The disk name can be up to 16 characters long, and may include letters, numbers, spaces, and/or periods. The ID can be any two letters and/or numbers. It's important that all your disks have different IDs, so that the computer doesn't accidentally get confused about which disk is in the drive.) This next example illustrates how you can abbreviate NEW as N, and how you can format more than one disk while you've got the command channel open. (Of course, you'll need to replace the disk in the drive between PRINT# commands.)

```
-----  
OPEN 15,8,15  
PRINT# 15, "N:GAME DISK 11:11"  
PRINT# 15, "NØ:PERS LETTERS,L1"  
CLOSE 15  
-----
```


(Note: the \emptyset after the N in the second PRINT# command stands for the DRIVE NUMBER, which is not the same as the device number, because it's possible to have one device containing two drives. It's not necessary to include this number unless you have more than one disk drive. On systems with two drives, the drive numbers are 0 and 1, and the device numbers are 8 and 9. If you have more than one drive, be sure to include a drive number after the command on all of the commands that follow.)

Open me first. So now that you know how to open and close the command channel (or at least you know how to look it up when you need it), there are some other handy messages you can send between opening and closing. If you want to try these, put your MY FIRST DISK back in the drive, and open the command channel by typing

```
-----  
OPEN 15,8,15  
-----
```

Deleting disk files. It's nice to be able to eliminate clutter from a disk after you've been using it for a while. Maybe you have several slightly different versions of the same program on disk filed under different names, and you want to use the space occupied by the older ones for something else. Once the command channel is open and the disk is in place, you can SCRATCH, or delete, the program called FALLING SOUND 2 by typing

```
-----  
PRINT# 15,"S:FALLING SOUND 2"  
-----
```

(Variations: You could use the complete word SCRATCH instead of the abbreviation S; you could also include the drive number \emptyset after the S or SCRATCH—a necessity if you have more than one drive.)

If the red light went out when the disk stopped moving, the program should be gone from the disk. If you want to be sure, check the directory:

```
-----  
LOAD "$",8  
LIST  
-----
```

Renaming a disk file. Your directory should now contain only one program: FALLING SOUND. Most programmers put numbers after their program names, so that they can keep old copies of programs for backup, but still remember which version is most current. (A program, like a painting, is never really done.) If you want to establish this convention, you can RENAME your program "FALLING SOUND 1.0." (to reflect its status as the first version of the program), by sending this command down the still-open channel:

```
-----  
PRINT# 15, R:FALLING SOUND  
1.0 =FALLING SOUND  
-----
```

(As usual, you could spell out RENAME rather than just typing R, and you could include a drive number after the command.)

Renaming a disk file is handy when you want to save your current program using a file name that's already been used on the disk.

Initializing a disk. Sometimes the disk drive gets confused as to what disk it's working with, and it refuses to obey your commands. If you've just switched disks and the drive won't do what it's told, it might help to INITIALIZE the disk with this command, assuming that the command channel is open:

```
-----  
PRINT# 15,"I"  
-----
```

It never hurts to issue this command, so some cautious folks do it every time they change disks. But it doesn't do much good either, unless you're working with two or more disks with the same ID. I usually wait until there's a problem.

(WARNING: If you ever switch to another brand of computer, be aware that initializing a disk may mean what formatting means to Commodore: wiping it clean and starting over!)

Validating a disk. This is a good thing to do when a disk has been around collecting programs and data for a while. The

VALIDATE command tidies up the disk, eliminating sections that for one reason or another shouldn't be there and squeezing everything else together so there's room for more new files on the disk. It looks like this (if the command channel is open):

```
-----  
PRINT# 15,"V"  
-----
```

Close me last. When you're done issuing disk commands, don't forget to close the command channel:

```
-----  
CLOSE 15  
-----
```

A little shortcut. If you're only going to issue one disk command while the channel is open, you can include the message in the OPEN command, instead of putting it in a separate PRINT# command. For example, you could reformat a disk with the following two commands:

```
-----  
OPEN 15,8,15,"N:FINANCIAL RECORDS, 84"  
CLOSE 15  
-----
```

The DOS wedge—a bigger shortcut. If you do a lot of work with the disk drive, you'll get tired of these awkward ways of telling the disk what to do. The people at Commodore anticipated that, and created a program to make it all easier. It's called "C-64 WEDGE" (or "DOS 5.1", or "C-64 SUPPORT") and you'll find it on many Commodore software disks, including the test/demo disk that came with your drive. It effectively "wedges" itself between the computer and the disk drive, acting as a communication facilitator so that you don't have to work so hard. To use it, all you need to do is insert that disk into the drive, load the program using the appropriate name), and run it:

```
-----  
LOAD "C-64 WEDGE", 8  
RUN  
-----
```



The computer will respond with a brief message identifying the program, and then say the usual READY. The important parts of the program are now tucked away in a remote corner of memory, waiting for you to use one or more of your new, abbreviated disk commands. Here's the new vocabulary:

COMMAND	MEANING
↑ (followed by program name)	load and run the program
/ (followed by program name)	load the program
%	load the program into a special memory area
← (followed by program name)	save the program
@ or > (followed by a disk command)	send the command to the drive (no need to OPEN or CLOSE anything!)
@ or > (followed by \$)	display the directory

(continued)

COMMAND

@ or > (alone)
disk drive error

MEANING

(without disturbing
the program in
memory!)
read and display the
channel

(This last one is handy when the disk drive's red light is blinking at you and you don't know why. It displays a message number and a brief explanation.)

Here are a few examples, most of which you've already seen in their longer form. (Note: no quotation marks.)

FALLING SOUND
/FALLING SOUND
FALLING SOUND
@N:MY FIRST DISK,A1
@S0:FALLING SOUND 2
@R:FALLING SOUND 1.0=FALLING SOUND
@I
@\$
@
@Q

The last command in this set is one you haven't seen; it tells the wedge to quit. Without this command, the wedge just keeps on working as long as you leave the computer on. (There are exceptions to this blanket statement: a few programs use the same corner of memory that the wedge uses; if you run one of those it will probably wipe out the wedge, making all of those cute commands unrecognizable to the computer again. The only solution is to rerun the wedge program.)

As you can see, the wedge can be a real time-saver when you're disk-diddling. Many programmers load and run this program first whenever they power up their computers. Others use it occasionally or never. It's all a question of taste.

A driver's test. By the way, there are lots of other interesting programs on that same test/demo disk. An especially valuable one is the "PERFORMANCE TEST," which runs an exhaustive set of tests on your disk drive, using a blank (or erasable) disk as its scratch pad. (Don't take one set of test results as the final word; sometimes the drive will fail the test because the disk is bad or the drive doesn't like the color of your shoelaces. Try it again.)



DISKETTE CARE

There are few things in life more exasperating than finding that disk damage has destroyed the program or data file that you spent hours (or months, for that matter) creating. Diskettes, like phonograph records, deteriorate rapidly if you don't take care of them. Here are some tips:

- Don't touch the disk surface; your skin's natural oils are much better for your body than they are for the disk.
- Don't smoke in the presence of a disk; the little smoke particles are as bad for a disk surface as they are for your lungs.
- Store your disks in a dust-free environment. Several types of boxes are designed with this in mind, or you can use zippered sandwich bags.
- Don't write on the disk sleeve while the diskette is inside.
- When you write on the disk label, tread lightly with a felt-tip pen. Don't erase. Don't use a ball point pen!
- Don't attach notes to the diskette with paper clips or rubber bands.
- Store disks upright, and avoid placing heavy objects on them.
- Don't fold, spindle, or you-know-what.
- Don't expose the disk to direct sunlight, excessive heat, magnetic objects, static electricity, or radiation.
- Don't leave a disk in the drive when you turn the drive or the system on or off; you may put a magnetic glitch on it.
- Protect important disks by covering the write-protect notch with tape. (It's usually supplied with the diskette.)
- Just in case, make backup copies of all important programs, and store them in a safe place. (There may be a program on the disk that came with your machine that will make a backup copy of an entire disk.)

If after all this, you find that a program won't load from a disk, it may help to wait a while and try again. Or, you

(continued)

can try to load the program on somebody else's disk drive, and if it works, save it on another disk.

Your disks will last longer if you take good care of your disk drive. Don't bang it around any more than you have to, and keep the head clean. (You can buy special head cleaning disks at most computer stores.)

USING THE PRINTER: A PRIMER

For a computer user (as opposed to a computer programmer) the printer is generally used in conjunction with a canned program, such as a word processor. The program takes care of all of the details of opening and closing communication channels and such without bothering the user. But there are two things that even a casual user might want to know how to do on her own: Print a paper copy of the disk directory, and print a listing of a program.

The BASIC commands for using the printer are a lot like the commands for using the disk drive. So if you found those unfathomable, you probably won't be able to make much sense out of these, either. On the other hand, if you were able to successfully use the disk commands, you should have no trouble with these.

To do anything with the printer, it's necessary to open a communication channel to the device, by associating a logical file number with a device number. The normal device number for the printer is 4, so for simplicity most programmers use 4 as the file number for the printer, as well. It's not necessary to list a separate command channel when opening the printer, so the complete command looks like this (make sure the printer is turned on):

```
-----  
OPEN 4,4  
-----
```


But you aren't ready to use the printer yet. You need to wake up the printer with the BASIC command CMD:

```
-----  
CMD 4  
-----
```

This command tells the computer to redirect output from the screen to the printer (which is associated with logical file number 4) until further notice. (The Commodore Printer manual says the printer is now "listening.") This means that if you now type

```
-----  
LIST  
-----
```

the program or directory in memory will be listed on the printer instead of screen.

Once you've completed your printer business, you'll need to do a couple of things to get things back to normal:

```
-----  
PRINT 4  
CLOSE 4  
-----
```

The PRINT# command tells the computer to print nothing in particular on the printer, but it has the effect of closing the local file. (Commodore calls it "unlistening.") Even so, it's important to formally close everything up with the CLOSE command.

And that's about it. To summarize, here's a recipe for printing the disk directory, so you have a paper copy to tape to the diskette sleeve:

```
-----  
LOAD"$",8  
OPEN4,4:CMD4:LIST  
PRINT#4:CLOSE4  
-----
```

Printing a program listing works exactly the same way, except that the program should be in memory instead of the directory.

Of course, there's plenty more that can be done with the printer, but not without a more detailed study of files than there's space for here.

SOUNDING OFF!

Many of the commands in this chapter can be used inside programs to do all sorts of fancy things, but it's not necessary to know about that now. You're now Commodore-literate enough to write and use your own simple programs, or at least take advantage of the world of already-written software for the Commodore 64.

Congratulations! You just completed your BASIC training! Give yourself a hand—or let this program do it for you:

```
-----  
1100 REM THE SOUND OF ONE CHIP CLAPPING  
1110 S=54272  
1115 FOR L=0 TO 24: POKE S+L,0: NEXT L  
1120 FOR C=1 TO 20  
1130 POKE S,240: POKE S+1,33  
1140 POKE S+5,8: POKE S+22,104  
1150 POKE S+23,1: POKE S+24,79  
1160 POKE S+4,129  
1165 FOR T=1 TO 90: NEXT T  
1170 POKE S+4,128  
1180 NEXT C  
-----
```

(Final exam: Put MY FIRST DISK back in the drive, or a cassette in your datasette, and save this program as CLAPPING 1.0. If you're following the programming parts of the book, you'll be able to use it next chapter. If you can't remember how to save a program, remember the crib sheet in back.)

CHAPTER 5

Software Shopping, Swapping, and Scrounging

*My grandfather said that of those he could hire,
Not a servant so faithful he found;
For it wasted no time, and had but one desire,
At the close of each week to be wound.
And it kept in its place, not a frown upon its face,
And its hands never hung by its side.
But it stopped short, never to go again, when the old
man died.*

—“*Grandfather’s Clock,*”
(*Traditional song by Henry Clay Work*)

Writing your own personal computer software is a little bit like sewing your own clothes. It’s possible to make it through the life looking very respectable without knowing a thimble from a bobbin, as long as you have the cash to pay someone else to construct and repair your clothes. But needles, thread, and a little bit of sewing knowledge can be liberating: you can replace buttons, patch knees, and make other emergency repairs; you can make minor alterations to your factory-made clothes to make them fit just right; you can even save yourself money by buying thrift-shop clothes and customizing them to suit your needs. It’s a big step, involving lots of study, practice, and equipment, from that point to the point where you can custom-make your entire wardrobe yourself. Few people liberate themselves from the clothing industry to that extent; most are comfortable buying professionally-made clothes.

Software is like soft wear. (Soft where?) You can use your personal computer like an intelligent appliance, depending on the experts to write your programs. But with a little bit of programming knowledge, you can make modifications in commercially-written programs so that they better meet your needs; you can type programs into your machine from books and magazines, and not feel helpless when they don’t run the first time; you can even write your own small programs from scratch. Or,

you can invest a little in software tools, and develop your skills for a year or so, so that you can write your own high-quality programs—maybe even become a professional programmer yourself!

The first part of this chapter is written with a “computer-as-home-appliance” approach; it talks about the process of choosing and using professionally-written software. Later on, the discussion moves to the thrift-shop software you can find in the public domain collections, books, and magazines. The chapter ends with another lesson in BASIC, so that, if you’re so inclined, you can repair and customize software to meet your own needs. Chapters 6 and 7 will discuss in detail the specific types of software that are available for the Commodore 64; Chapter 8 discusses the art of writing your own professional-quality software.

INTELLIGENT SOFTWARE BUYING

Buying the best software isn’t easy. Software cost is difficult to evaluate before you buy, and is often disappointing after you buy. Everybody seems to be getting into the software business lately: computer manufacturers, book and magazine publishers, multinational corporations, and 12-year-old kids. There are virtually no standards in this young industry, so it’s hard to know what to expect when you lay your money down.

But there’s no need to be terrified by the software jungle; a little planning and questioning can make most software safaris successful.

First off, figure out what you want from the program. This may sound trivial, but it’s crucial to know what you’re trying to do. Maybe your goal is something simple like “I want a game to entertain me this weekend.” On the other hand, if your answer is something like “I want a word processing program,” then you haven’t really answered the question. Why do you want a word processor? What problem(s) are you trying to solve with a word processor? (A lack of productivity? Low grades? Fear of typing?) What do you plan to do with it? (Write letters? Term

papers? Books? Junk mail?) If you can't answer these questions before you shop, then you're likely to be overwhelmed by the myriad of choices.

Next, figure out how much you have to spend. Don't assume that you'll get better software just by spending more money. The correlation between price and quality is weaker in software than it is in most other consumer goods, so you can sometimes buy a quality product that meets your needs exactly for less than you'd expect. 🐾

How to find that quality bargain? For starters, look through the last half of this book. Then ask around—fellow computer users, user group members, software and hardware salespeople—anyone who might have experience with the products you're considering. Read reviews in books and magazines (listed in Appendix A), keeping in mind that (1) the reviewers are just people with informed, but subjective, opinions; and (2) many periodicals shy away from negative reviews because they want to protect their precious advertising revenues.

You can also learn about a software product by looking over the *documentation*—the written instructions that accompany the product. Most computer stores will let you examine the documentation in the store before you buy. As you study it, ask yourself these questions:

- Will this program work with my equipment? Make sure the program is written for the Commodore 64; software for other machines generally won't run on the C-64. (Exceptions are discussed in the last section of this chapter and in Chapter 9.) But that's not the only equipment check you should make. Some programs require a disk drive to work properly. Others require a datasette. Still others assume you have joysticks or other paraphernalia. And some expect a printer. If the program works with a printer, make sure it will work with your brand of printer.
- Is the documentation easy to read and understand? The doc-

🐾 In part, this is because software prices are rapidly changing—usually in a downward direction. Which is why you won't find any exact price quotations in this book.

umentation will probably have to serve as both your teacher and your reference guide for the program; it should be designed to meet both these needs. If you can't understand it in the store, it probably won't help much at home. If it lacks examples, an index, and clear organization, then it may be difficult to use when you need it.

- Do the program instructions look simple and straightforward? If not, the program itself might be complicated and hard to use. The care that goes into the documentation often (but not always) reflects the care that goes into the software.
- What happens when something goes wrong? Commercially produced software can be incredibly complex, and even the best software is seldom completely bug-free. If you use a new program often, with enough different kinds of data, there's a good chance that it will do something unexpected: print garbage on the screen, refuse to save your file, freeze the keyboard so you can't type anything . . . you name it. Sometimes this is annoying; occasionally it's disastrous.

Since computer scientists haven't yet developed foolproof methods of debugging large software products, it's not realistic to expect a new piece of software to perform perfectly under all circumstances. But you should expect the manufacturer of the program to back their product when it misbehaves. The documentation should contain a phone number you can call for help or information. It should also state the company's replacement policy if the program doesn't load or run properly. (Alternatively, make sure your dealer will allow you to exchange it.)

- What about backups and upgrades? Nobody likes to be ripped off, and programmers are no exception. They're especially vulnerable to rip-offs though, because computers make it so easy to copy their products onto other disks or tapes. (You learned how last chapter!) So, to protect their maker's paychecks, many programs are altered so they are difficult to copy, which, in itself, is quite reasonable except that when your legitimately-purchased copy of a copy-proof program bites the dust, gets struck by lightning, or dies of old age, you don't want to have

to buy a whole new software package. So make sure that the documentation that comes with the product includes a provision for getting a backup copy when and if the need arises. Most reputable software firms allow bona-fide purchasers of their products to replace dead disks or tapes for a few dollars. They usually make a similar offer for trading in your copy when a new, improved version of the program (an upgrade) is released.

- Is the program written in BASIC or in machine language? Sometimes the documentation will say something like “all BASIC” or “100 percent machine code.” This may seem irrelevant, but it can make an important difference in several ways. BASIC programs run much slower than their machine language counterparts, because the translation from BASIC takes time. For normal data processing programs, that may not matter, but for many applications (especially video games, and to a lesser extent word processors, spreadsheets, and the like) speed can be crucial. On the other hand, a program written in BASIC is much easier to customize, debug, and copy than a machine-language program.

There are lots of things to look for when you’re checking out the documentation. But there’s no substitute for actually using the program. So, if at all possible, take the time to sit down in the store and get to know the product before you buy. It may seem awkward at first; many quality programs do. But with a good manual and a little help from a salesperson, you should be able to work it enough to see if it suits you well.

MAKING FRIENDS WITH STORE-BOUGHT SOFTWARE

Hopefully, most of the programs you buy will be so easy to understand and use that these tips will seem unnecessary. But at some point you’re likely to encounter a program that plays hard-to-get-to-know. Maybe it was designed by an engineer who didn’t remember what it felt

(continued)

like to be a casual computer user; maybe the user's guide was thrown together too quickly; or maybe it's just a complicated program with lots of features to learn.

It's up to you to make up for the shortcomings in the design of the program or the documentation. If the user's manual for a complicated program doesn't have a summary page similar to the one in the back of this book, you'll save yourself lots of time by making one as you read the manual. If it doesn't have an index, use stick-on index tabs to highlight important pages. If it lacks examples, take notes on your own trials and errors, so you don't get caught in the same traps twice.

Many programs take advantage of the four FUNCTION KEYS on the right side of the keyboard. These powerful keys can mean whatever the programmer tells them to mean, and a good program can use them to great advantage. But it's not always easy to remember what these keys do when you're using the program without some kind of reminder labels, like these made-to-be-photocopied prototypes, which are just the right size to be cut out and strategically positioned on your keyboard. Stick-on note pads are handy for this, too. (Some otherwise fine programs take the liberty of redefining other keys on the keyboard so they don't do what they say they do; if you use one of these programs, be generous with reminder notes.)

Finally, take your time. It may be very frustrating to have to take four or five hours to learn to use a program that's supposed to save you time, but that's just the way those kinds of things work. You didn't get there faster the first time you rode a bike; don't expect instant results with software. If you did your shopping wisely, your training time will be repaid over and over.

PUBLIC DOMAIN PROGRAMS: SOFTWARE FOR FREE

It's easy to spend money on software. It's also easy to get software without spending money. Ever since Commodore sold its first PET microcomputer back in 1977, people have been writing programs for Commodore PET computers. Many of these programs were written by professionals for copyright and sale; others were written without the slightest thought of profit; the authors were delighted to share their works with anyone who wanted them. These programs are legally in the *Public Domain*, just like the works of Shakespeare. Over the years, some dedicated PET user groups have carefully collected and cataloged thousands of these programs, and made them available to PET users for a nominal disk-copying charge. The largest such library is maintained by the Toronto Pet User Group, and is described and partially cataloged in the *Whole Pet Catalog* (see Appendix A).

The Commodore 64 is a far cry from the early PETs; it has lots more memory, full color display, sophisticated sound capabilities, and a more advanced internal design. Fortunately, the two machines are similar enough that many PET programs will run on the 64 with little or no modification. So the large library of public-domain PET programs is potentially of value to Commodore 64 users.

The Ontario Public Schools and Commodore have converted hundreds of these programs so that they will run on both the PET and the 64, and have made them available for little more than the cost of the disks they're stored on, through Commodore dealers and user groups everywhere. While most of these programs are educationally-oriented, the collection also include lots of games, financial programs, and utilities (programs that are designed to make it easier to program or use the computer).

These programs represent just the first wave of public-domain programs for the 64. New ones are being written and donated all the time to user's groups. Many are translations of programs originally written for other computers; others are original works.

How good are these public-domain programs? Some are worthless; a few are priceless. The vast majority are good, workable programs that do one thing pretty well, but don't set the machine on fire. How do they compare with their commercial counterparts? In general, they aren't as slick or fancy. The graphics are often crude or dated; the designs usually aren't as flexible and powerful; documentation is generally sparse; and the programs sometimes contain troubling errors (which are even more bothersome with public-domain software, because there's no dealer or manufacturer to support the product). But there are many exceptions. And many seasoned computer users prefer public-domain programs because they often have more "personality," in the same way that small-town newspapers are more folksy than *Newsweek* and *Time*.

To find out more about public-domain software, ask your dealer, or get involved with a user group. Commodore publishes a list of active user groups in their *COMMODORE* magazine (a few of the larger ones are listed in Appendix A of this book). If you can't find one in your area, you can always start one by sending an announcement to that magazine and posting signs at local computer stores. 🐾 If the public-domain software bug bites you hard, you should consider adding CP/M to your system; Chapter 8 has the details.

ALMOST-FREE SOFTWARE: BOOKS AND MAGAZINES

If you have more time than money to invest in software, you can take advantage of the myriad of programs that are available in computer books and magazines. And in many cases, the only skill you need is the ability to type.

There are hundreds of books and magazines in print that contain complete listings of BASIC programs: games, educational programs, household helpers, business tools, programming utilities—you name it! (Many of these books and magazines are listed in Appendix A.) Of course, type-in programs generally aren't as fancy as the expensive commercial packages; if

🐾 See "Starting a User Group," in the May, 1983, issue of *Commodore Magazine*.

they were, they'd take ages to type. But many of these programs are of surprisingly high quality, considering the price.

(If you're not interested in doing all of that typing, you still might be able to take advantage of these published programs. Many books of programs are sold with optional diskettes that contain the already-typed programs.)

Some of these published programs are written specifically for the Commodore 64, and can be run as soon as they are typed in—assuming, of course, that you made no typing errors. And that's a big assumption. Even a world-class typist would find it challenging to type a long program listing full of mysterious numbers and unfamiliar symbols without making a single typo. And a mistyped number or misplaced punctuation mark can make all the difference between a working program and worthless collection of symbols.

If a program doesn't work when you type it in, you have three choices: punt; compare your listing with the original, character-by-character (and hope that the listing in the book or magazine is error-free!); or play Sherlock Holmes and try to logically deduce what went wrong. Detective work combined with liberal doses of listing-comparison can usually get a program up and running in time. And this kind of debugging can teach you a lot about programming, if it doesn't drive you crazy first! But to pull it off, you need to understand the features of BASIC that are used in the program. The next two sections of this chapter will inventory most of the remaining BASIC keywords that you're likely to encounter in a program listing, so that you'll be better equipped to debug the programs you type into your computer.

Many BASIC programs printed in books and magazines are not written for the C-64, but they're written in a generic form of BASIC that will run with little or no modification on your machine. Others would require so much tinkering to make them work for you that it would be easier to start over. With a little bit more knowledge of BASIC, you'll be able to tell from the listing whether a published program is potentially compatible with your C-64, and what it needs to make it work for you. The rest of this chapter should help.

MORE BASIC: SUBROUTINES, FUNCTIONS, ARRAYS, ODDS AND ENDS

If you took the long way through Chapter 3, you're familiar with the basics of BASIC. But there's more to BASIC than what you've seen there. Whole books are devoted to the subject of programming in BASIC (see Appendix A), and this half-chapter won't make you an expert. But it will give you a brief survey of the BASIC keywords you haven't seen, so that when you're transcribing a program from a printed listing, or when you're trying to modify a public-domain program, you'll have a better understanding of what's going on.

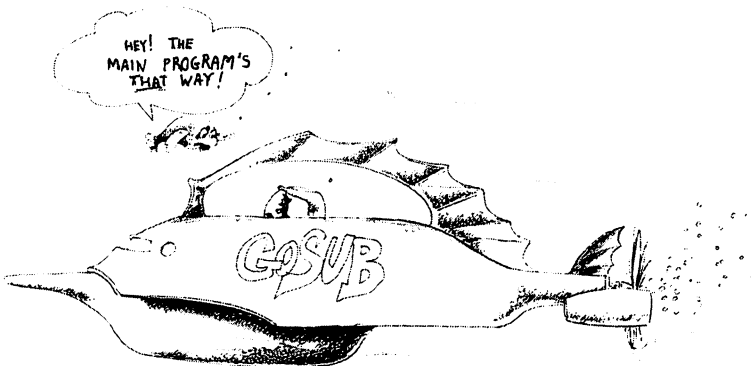
If you're so inclined, you can treat this material like a tutorial, typing in the programs and tracing through the examples. Or you can skim it quickly so you'll know what's here in case you need it later. Or, if you're a casual computerist, you can skip it altogether. Suit yourself.

GOSUB and RETURN. Chapter 3 introduced you to the GOTO statement, which acts like a one-way ticket to some other statement in the program. GOSUB is the round-trip version. Like GOTO, it tells the computer to transfer control to another statement. But before it goes to the destination statement, the computer makes a note of where it is in the program, so that it can come back and continue where it left off when it encounters a RETURN statement. This makes it possible to build a large program by stringing together several smaller SUBROUTINES or SUBPROGRAMS.

Let's look at an example. (Try it on your computer, if you like.) Suppose we want to modify our multiplication drill program from Chapter 3 so that it claps when the student gets a right answer. You may already have a program that makes a clapping sound SAVED on your disk or tape as CLAPPING 1.0. If so, you can save yourself the trouble of typing it by loading it into memory. We'll use that program as a subroutine; our multiplication program will CALL that subroutine with a GOSUB whenever applause is appropriate.

The next step is to type in the program from Chapter 3 without clearing the memory. Here's what you should have when you list it:

```
-----  
100 PRINT "MULTIPLICATION DRILL:"  
110 INPUT "WHICH NUMBER DO YOU WANT TO  
PRACTICE";M  
120 FOR N=1 TO 9  
130 PRINT "WHAT'S";M;"TIMES";N;  
140 INPUT A  
145 IF A<>M*N THEN 130  
150 PRINT "RIGHT!"  
160 NEXT N  
1100 REM THE SOUND OF ONE CHIP CLAPPING  
1110 S=54272  
1115 FOR L=0 TO24: POKE S+L,0: NEXT L  
1120 FOR C=1TO 20  
1130 POKE S,240: POKE S+1,33  
1140 POKE S+5,8: POKE S+22,104  
1150 POKE S+23,1: POKE S+24,79  
1160 POKE S+4,129  
1165 FOR T= 1 TO 90:NEXT T  
1170 POKE S+4,128  
1180 NEXT C  
-----
```



If you run the program right now, it'll do the multiplication drill just like before, and then clap when the drill is over, regardless of how you did. While this arrangement may be appropriate, it doesn't demonstrate subroutines. To make the clapping part into a subroutine that gets called for each correct answer, make the following changes:

```
-----  
155 GOSUB 1110: REM CLAPPING SUBROUTINE  
1105 REM CLAPPING SUBROUTINE  
1190 RETURN  
-----
```

Here's an abbreviated listing of what the program should look like now:

```
-----  
100 PRINT "MULTIPLICATION DRILL:"  
110 INPUT "WHICH NUMBER DO YOU WANT TO  
PRACTICE";M  
120 FOR N= 1 TO 9  
130 PRINT "WHAT'S";M;"TIMES";N;  
140 INPUT A  
145 IF A<>M*N THEN 130  
150 PRINT "RIGHT!"  
155 GOSUB 110: REM CLAPPING SUBROUTINE  
160 NEXT N  
1105 REM CLAPPING SUBROUTINE  
1110 S = 54272  
  3  
  ↓  
1180 NEXT C  
1190 RETURN  
-----
```

It works just like it did in Chapter 3, except when the correct answer is entered. When that happens, making $A = M * N$, the program still says RIGHT! at line 150. But the GOSUB at line 155 transfers control temporarily to line 1110, which is the first real line of the clapping routine. (You'll remember that REM statements are always ignored by the computer; they're for people.) The computer proceeds from line 1110, merrily clapping right through line 1180. When it gets to the RETURN statement at 1190, it goes back to line 160—the next statement after the GOSUB —and takes up where it left off.

Since there's no limit to the number of subroutines you can add to a program, they're an extremely powerful tool for converting simple little programs like this one into sophisticated big ones! (Speaking of sophisticated, you could always add a subroutine to this program so that it gives the raspberries for wrong answers.)

There is a subtle bug in this program, which you've probably noticed by now if you typed it in and ran it: after the computer goes through all of the numbers in the multiplication table, it charges right on past line 160 into the clapping subroutine. This would be OK (you might want the computer to clap when you're done), except that it complains when it encounters the RETURN without a matching GOSUB. There's an easy way to stop it before it gets there.

END and STOP. END tells the computer to stop in its tracks. It can be used as the last statement in a program, although it really doesn't do anything useful there. It is useful when you want the program to stop without going all the way to the last statement, as in our multiplication program. To stop the program before it falls through to the subroutine, type

```
-----  
200 END  
-----
```

Alternatively, you could use the keyword STOP instead of END; the only difference is that when the computer encounters a STOP statement, it announces the line number of the statement, by saying BREAK IN 200. This serves no purpose here, but it's handy if you have several STOPS in a program and you want to know which one did the job.

(By the way, if you like the idea of having the program clap when you finish, you can add the GOSUB statement shown here.

```
-----  
170 PRINT "  HOORAY!"  
180 GOSUB 1110  
-----
```

The PRINT statement is just for fun. It clears the screen, moves the cursor down and over, and prints HOORAY! The reversed characters in the string before the HOORAY are made by typing SHIFT-CLR/HOME, CURSOR DOWN four times, CURSOR RIGHT 15 times and CONTROL-9.)

RND and INT. Chapter 2 introduced you to SQR and several mathematical functions. Commodore BASIC has other functions, some of which turn up in lots of magazine and book programs. One of the most important of these is RND, the RANDOM NUMBER function. RND is the computer's equivalent of flipping a coin or throwing a die. Each time this function is called, it returns a random number between 0.0 and 1.0. Try typing this command:

```
-----  
PRINT RND(0)  
-----
```

Now cursor up to that same PRINT line and RETURN again. Notice how a different number gets printed each time you repeat this process. 🐾

A random value between 0 and 1 may not seem very exciting, but it can be used to add an element of surprise to an educational drill, shuffle a deck of cards, or put variety in a video game, among other things. It works especially well with INT, a function that chops the decimal part off whatever number is passed to it inside parentheses. Try these:

```
-----  
PRINT INT(5.9)  
PRINT INT (RND(0)*9) + 1  
-----
```

The first example should cause 5 to be displayed. The second one first picks a random number between 0 and 1 (the computer always starts at the inside of nested parentheses like these):

🐾RND actually returns a PSEUDO-random number, because the different calls to RND don't technically return statistically independent values. This distinction is important to statisticians and philosophers, but it shouldn't make any difference to you (unless the number in parentheses after the RND is negative, in which case you'll get exactly the same sequence of numbers each time you turn the computer on).

We can use this trick in the multiplication drill program to randomly choose values to be multiplied, so that the program doesn't always go through the tables in order. The only changes needed are shown here:

```
-----  
125 N = INT(RND(0)*9) + 1  
120 FOR I = 1 TO 9  
160 NEXT I  
-----
```

The new line 125 calculates a random number between 1 and 9, and puts it in the N box. Since it's illegal in BASIC to change the value of a FOR-loop counting variable inside the loop, we need to change the N in lines 120 and 160 to something else, like T. The changes you just made should look like this inside the program:

```
-----  
100 PRINT "MULTIPLICATION DRILL:"  
110 INPUT "WHICH NUMBER DO YOU WANT TO  
PRACTICE";M  
120 FOR I = 1 TO 9  
125 N = INT (RND(0)*9) + 1  
130 PRINT "WHAT'S";M;"TIMES";N;  
140 INPUT A  
145 IF A<>M*N THEN 130  
150 PRINT "RIGHT!"  
155 GOSUB 1110: REM CLAPPING SUBROUTINE  
160 NEXT I  
-----
```



ON . . . GOSUB and ON . . . GOTO. Before we leave this program behind, let's make it more interesting by adding some variety to the responses it gives when the correct answer is given. We'll use our trusty random-number generator and a new statement: ON . . . GOSUB. Here are the new lines:

```
-----  
150 R=INT(RND(0)*3)+1  
152 ON R GOSUB 300, 400, 500  
300 PRINT "THAT'S EXACTLY RIGHT!": RETURN  
400 PRINT "PERFECT!": RETURN  
500 PRINT "YOU GOT IT!": RETURN  
-----
```

Line 150 uses RND and INT to choose a random integer between 1 and 3 and puts it in R; that number is used in line 152 to determine which of three responses will be given. The ON. . .GOSUB statement works just like GOSUB, transferring control temporarily to a subroutine. The difference is that this statement gives us a choice of subroutines, depending on the value of the variable R. Line 152 can be read like this: "If the value of R is 1, then GOSUB 300; if R is 2, then GOSUB 400; if R is 3, GOSUB 500." Whichever path the computer chooses, it will execute a one line subroutine praising the answer and then RETURN to 155, the line after the GOSUB. Try it! (It can't happen in this program, but for the record, any other non-negative value of R would have caused the computer to go on to the next statement, and a negative R would have given an error message.)

There's another BASIC statement—ON. . .GOTO—which works like ON. . .GOSUB, sending the computer to a line number depending on the value of the variable that sits between the two keywords. The difference is that no return ticket is included with the ON. . .GOTO transfer.

SYS andUSR. There are two more control-transferring reserve words that you should know about if you're going to be snooping around in other people's BASIC program listings: the statement SYS, and the functionUSR. Both SYS andUSR transfer control to a machine language routine stored in the computer's memory. That routine may be a built-in subroutine that does something simple like clear the screen and return to your program, or (in the case ofSYS) it may be a large machine-language program that takes the helm away from the BASIC

program forever. Hopefully, there'll be a REM in the listing to clue you in.

String functions. Commodore BASIC has a number of functions that are specifically designed for working with strings of characters. The easiest way to understand these is to look at some examples:

```
-----  
LET SS="MONKEY BUSINESS"  
PRINT LEFT$(SS,1),RIGHT$(SS,2),MID$(SS,8,3)  
PRINT LEN(SS)  
-----
```

LEFT\$ in this example returns the leftmost one character in the string, or M. RIGHT\$ here returns the rightmost two characters, SS. MID\$ returns three characters, starting with the eighth: BUS. and LEN returns the number of characters (including spaces) in the string, 15. These functions often turn up in word processors and other character-oriented programs.

PEEK. There are a few other functions that any good program scrounger should know about. Probably the most widely used of these is PEEK, the functional opposite of POKE. PEEK is like a window through which you can examine the contents of any memory locations in your computer. Example:

```
-----  
POKE 40000, 100: PRINT PEEK(40000)  
-----
```

This POKE statement pokes the value 100 into memory location 40000, the PRINT statement tells the computer to PEEK into that same location and display whatever it finds there. PEEK and POKE are used a lot together in programs. PEEK is usually used for checking special memory locations to find out such things as whether two characters in a video game collided.

CHR\$ and ASC. Try this:

```
-----  
PRINT CHR$(147), CHR$(65), CHR$(218)  
-----
```

Surprised? You just told the computer to print three characters, as you've done many times before. But this time, instead of

typing the characters yourself, you used the CHR\$ function to specify the numeric codes of the characters you wanted to print. The number 147 is the numeric code for the SHIFT-CLR/HOME; when you “printed” that, the screen cleared. 65 is the code for the letter A, and 218 is the code for the diamond. Every key-stroke combination that you’ve used, including cursor moves and other editing facilities, is represented in the computer as a numeric code. The code used for these characters is ASCII (pronounced ass-key; stands for American Standard Code for Information Interchange), and you’ll find a complete table of ASCII code values in Appendix F of your *Commodore 64 User’s Guide*.

The opposite of CHR\$ is ASC. This function returns the numeric code associated with a character. To try it type this (type SHIFT-CLR/HOME for the inverse heart, and SHIFT-Z for the diamond):

```
-----  
PRINT ASC("♥"), ASC("A"), ASC("◆")  
-----
```

This command should print 147, 65, and 218. (Compare it with the last example.)

There are other functions in BASIC; you can even invent new ones to include in your programs (using DF to Define the Function and FN to call it). But you’ve seen the important ones, and you can look the others up in the *Commodore 64 User’s Guide* or the *Commodore 64 Reference Manual* if you need them.

Arrays. You’ve probably noticed by now that a function name in BASIC programs is almost always followed by at least one number or variable in parentheses. This is referred to in the vernacular as a *parameter*, and it’s used to send a value to the function.

It’s unfortunate that the designers of BASIC chose a similar notation to represent a very different concept: the *array* and its *subscript*. An array is a mechanism in BASIC that allows us to refer to several different memory locations with the same name. We use a subscript to distinguish the different *elements* of the array. Here’s a program that uses an array in tallying the votes for a three-way election:

```

-----
NEW
100 REM TALLY VOTES FOR 3 CANDIDATES
110 DIM C(3)
130 REM INITIALIZE VOTE COUNTERS TO 0
140 FOR N=1 TO 3: C(N)=0: NEXT N
150 REM READ AND TALLY VOTES
160 INPUT "ENTER VOTE (0 WHEN DONE)";V
170 IF V=0 THEN 210
180 C(V)=C(V)+1
190 GOTO 160
200 REM DISPLAY FINAL COUNTS
210 PRINT "CANDIDATE VOTE"
220 FOR N=1 TO 3: PRINT N, C(N): NEXT N
-----

```

The DIM statement in line 110 sets up the dimension for our array: it says that we will have three boxes, all using the name C. We'll refer to them as C(1), C(2), and C(3). (Actually, there's a fourth box, C(0), but we won't use that one in this program. Also, in Commodore BASIC, the DIM statement is only necessary when you have an array with more than 10 elements, so we really don't need it here. But the program is easier to read and modify if we include it.)

We're going to use that array to store the vote count for each of our three candidates, so we have to start by putting zeros in all three boxes. That's what the FOR/NEXT loop in line 140 does: by varying the value of N from 1 to 3, it assigns a value of 0 to C(1), C(2), and C(3).

The next part of the program is another loop that reads votes from the keyboard and adds them to the appropriate counters. Each vote is read into the box V at line 160; then the program checks to see if a vote of 0 was entered on line 170. Let's say the first vote was a 2, for candidate 2. Since that's not 0, we fall through to line 180, which says to add one to the value stored in C(2) (because V has a value of 2), and put the new value right back into box C(2). So C(2) now has a 1, and the other 2 C boxes contain 0. Line 190 sends the computer back up to line 160 to read another vote.

And on and on, until a vote of 0 tells the computer to jump out of the loop because the voting is done. Line 210 then prints headings to label the results, and the loop on line 220 prints each candidate's number and vote-count. Try it and see!

This program can be easily modified to accommodate any number of candidates by changing all of the 3s to some other number. And we can modify it so that it prints candidate names instead of numbers by storing those names in a `STRING ARRAY`:

```
-----  
110 DIM C(3), NA$(3)  
115 REM READ CANDIDATE NAMES  
120 FOR N= 1 TO 3  
123 PRINT "ENTER NAME FOR CANDIDATE";N  
125 INPUT NA$(N)  
128 NEXT N  
220 FOR N= 1 TO 3: PRINT NA$(N), C(N): NEXT N  
-----
```

But this example doesn't begin to illustrate the power and versatility of arrays. For one thing, arrays can have more than one subscript, with each subscript representing a different dimension. That means that we can use arrays to represent things like the squares on a checkerboard, or a grid of characters on a screen, or the cells of a complex accounting spreadsheet. 🐾 There's no room here for more examples, but there are lots of good BASIC books that can help if you want to delve further.

MORE BASIC INPUT AND OUTPUT

Input and output are traditionally among the most difficult topics for a beginning programmer to master, because there are so many miniscule matters to attend to. Chapter 3 showed you how `INPUT` and `PRINT` can be used in BASIC to get information into and out of the computer. As it turns out, there are lots of other ways. This section will give you an introduction, without getting bogged down in the messy details.

GET. You're likely to see `GET` in lots of programs. Like `INPUT`, it reads characters that you type on the keyboard. The difference is that `GET` looks at each character as soon as you

🐾 A two-dimensional array is often called a `MATRIX` or a `TABLE`.

type it, instead of waiting for you to hit carriage return. GET is favored over INPUT by most programmers because it gives them more control over the input operation, and allows them to make their programs more error-resistant. Here's a commonly-used GET statement:

```
10 GET CH$: IF CH$ = "" THEN 10
```

The GET statement tells the computer to get a character from the keyboard (or a signal from the joystick port) and put it in the memory location called CH\$. If nothing was typed, then CH\$ is empty, which makes CH\$ = "" (since "" is NULL, or empty, STRING). In that case, the computer goes back to 10 and tries again.

READ and DATA. Sometimes it's inconvenient to have to type in data values every time you run a program. Consider, for example, your computer's special moveable sprite characters. These amazing critters can be defined to look like anything from the space shuttle to Miss Piggy, and then moved around the screen at will, interacting with you and each other. But to give a sprite its shape and colors, you need to POKE 63 numbers into some special memory locations. Some of the tedium of this process can be removed by storing these numbers in special statements in your program called DATA statements, like this:

```
NEW
1010 DATA 1,160,0,1,168,0,1,192,0
1020 DATA 13,192,0,13,240,0,13,242,128
1030 DATA 13,254,160,13,252,0,61,255,0
1040 DATA 61,255,0,61,255,192,61,255,192
1050 DATA 61,255,240,253,255,240,253,255,
124
1060 DATA 253,85,85,1,0,68,85,85,85
1070 DATA 85,85,84,21,85,80,5,85,64
```

This is not the time to explain what these numbers mean. 🐾

🐾 Suffice it to say that they are decimal representations of binary numbers which, when loaded into the proper memory locations, tell the computer which parts of the sprite character should be left blank and which should be colored in. Many of the books listed in Appendix A, including the *Programmer's Reference Guide*, can fill you in on the details.)

The important question is, what can we do with them? For starters, we can use the BASIC READ statement to read them into variables in our program.

The READ statements works just like INPUT, except that it reads values from DATA statements instead of from the keyboard. The first READ statement reads the first number in the first DATA statement into a variable. (If the READ statement contains more than one variable name, then it reads enough DATA values to fill all of the variables, moving through them in order.) The next READ statement encountered in the program reads data starting where the first one left off, and so on. When all the data in one DATA statement has been read, the computer moves on to the first value in the next one, and so on. (Unless a RESET statement is executed. RESET tells the computer to start over reading from the beginning of the first DATA statement again with the next READ.) The rest of our sprite program shows the READ statement in action:

```
-----  
100 REM SAILBOAT SPRITE  
110 PRINT "♥": REM SHIFT-CLEAR/HOME  
120 POKE 2040,13: REM ESTABLISH MEMORY  
AREA FOR SPRITE 0 DATA  
130 V=53248: REM VIDEO CHIP MEMORY AREA  
BEGINS HERE  
140 POKE V+21,1: REM TURN ON SPRITE 0  
150 POKE V+28,1: REM SET MULTICOLOR SPRITE  
MODE  
160 POKE V+39,4: POKE V+37,0: POKE V+38,1:  
REM SET COLORS  
195 REM READ 63 DATA VALUES INTO SPRITE  
MEMORY AREA  
200 FOR N=0 TO 62: READ D: POKE 832+N,D:  
NEXT N  
205 POKE V+1,229: REM SET SPRITE VERTICAL  
POSITION  
208 REM  
210 POKE V+16,1: MX=91: REM RIGHT SIDE  
220 GOSUB 310  
-----
```



```

230 POKE V + 16,0: MX = 255: REM LEFT SIDE
240 GOSUB 310
250 GOTO 210: REM ENDLESS LOOP
260 REM
300 REM SUBROUTINE FOR MOVING SPRITE
310 FOR X = MX TO 0 STEP -1
320 POKE V,X: REM SET SPRITE HORIZONTAL
POSITION
330 FOR T = 1 TO 15: NEXT T: REM TIMER
340 NEXT X
1000 REM 63 SPRITE DATA VALUES READ BY
STATEMENT 200
-----

```

This program listing is chock full of REM statements to give you a few clues as to what's going on with all of those POKEs. The only part of this program that's relevant to our discussion here is the READ statement embedded in line 200. Each time the computer hits that statement it reads into variable D the next number from the DATA statements that start at line 1010. The value is then POKEd into a special memory location, and the FOR/NEXT loop starts the READ-POKE process over again. Since that loop causes the READ statement to be executed 63 times, our DATA statements need to contain at least 63 values. They do, so we can go happily sailing off into the sunset. (It's up to you to program the sunset part!)

Data files. There's another way of storing data that's a lot more versatile than the DATA statement: data files. You've already been briefly introduced to the concept of files in Chapter 6, when you learned how to save and load programs as files on disk and tape. Of course, programs aren't the only things that can be stored on magnetic media; you can also save data. For example, a checkbook program might write the end-of-month balance and all of the outstanding checks onto a disk or tape file, so that next month it could read those values in automatically, instead of requiring you to type them in. A word processor should allow you to save a copy of your text as a disk or tape file, so you can retrieve it later to make changes or print out a copy. And a database management program's sole

purpose in life is to store data, whether recipes or mailing addresses, in an easy-to-manage form. In addition, any program that communicates with the printer or other external device does so by sending and receiving files.

This section won't tell you how to write programs to do any of these things; entire books have been written with that in mind. The idea here is to show you what the file-related commands look like so you'll recognize them when you see them in programs.

If you read the section on disk operations in Chapter 4, you've already encountered some of the most important file commands. OPEN, you'll remember, is the computer's equivalent of opening the file cabinet so that it can put information in or take it out. OPEN does a lot more than that. It also associates a logical file number (the first number after the keyword) with a physical device (symbolized by the second number). It specifies whether we'll be READING data from the file or writing data ONTO the file. (If you're using cassettes, this is done with a numeric code; if you're using diskettes, the codes R and W are used.) OPEN also associates a name with the file, for your convenience. The syntax varies for the OPEN command depending on whether you're opening a file on disk, tape, or some other device like the printer. Consult the *Programmer's Reference Guide* for details.

CLOSE (also discussed in Chapter 4) is used to close any files that were OPENED when they're not needed any more by the program. The CLOSE command doesn't have to say anything except which file is being closed, so it only has one number after the keyword: the logical file number. That number always matches the first number of the open command, for obvious reasons.

Opening and closing files would be of little value if the program didn't do something with the file in between. There is one BASIC keyword, PRINT#, that allows you to write data onto an open file, and two, GET# and INPUT#, that read data from a file. Except for the trailing tic-tac-toe symbol, these commands look like the keyboard and screen commands you've already

seen. And for the most part, they work like their familiar namesakes. The major difference is that each of these keywords must be followed by the logical number of the appropriate file.

There's one other keyword associated with I/O (computerese for input/output): STATUS. This function tells the computer the completion status of the last I/O operation. In other words, a program checks STATUS after a read or write operation to find out if there were any problems (bad disk, out of tape, etc.)

The manuals that accompany the disk drive, the datasette, and the printer contain details on how each of these commands works with those devices.

RECOGNIZING GENERIC BASIC: A TRANSLATOR'S GUIDE

BASIC is spoken by more computers than any other language. But BASIC, like English, is a language of dialects. The BASIC that one computer understands may be very different from the BASIC of another, making it difficult to write a program that will work on both computers without some changes.

Fortunately, there are more similarities than differences between different BASICs. Most of the commands and statements you've seen are more or less standard from one machine to the next. If you look at any of the major BASIC textbooks, you'll see a sort of generic BASIC that includes only keywords that have relatively standard meanings. Programs written in this textbook BASIC (sometimes called MINIMAL BASIC) can be typed and run on most computers, including Commodore 64s, without major modifications.

This means that you can take advantage of the hundreds of generic BASIC programs that are available in textbooks, computer hobbyist books, and magazines. But you may run into some problems trying to make these programs work on your computer unless you're aware of a few quirks that Commodore BASIC has. This section will discuss some of those quirks, and show you how to make most generic BASIC programs run on

your computer. The chapter will close with a discussion of converting programs specifically designed to run on other computer models so they'll run on the Commodore 64.

Converting generic Basic to Commodore Basic. One constraint that may prevent a textbook BASIC program from working properly on the Commodore 64 is the 64's 40-column screen. Many published programs assume that your computer has a screen width of 72 or 80 columns, and their output displays reflect that. This difference won't affect the internal workings of the program in the least, but the wrap around can turn an easy-to-read table of results into a confusing jumble of numbers and words. If you can't get used to a display like that, the only solution is to rewrite the PRINT statements so that no line is longer than 40 characters. Fortunately, most microcomputers used in schools and homes today have the same 40-column limitation, and most modern BASIC books have been written with that in mind.

There is another possible source of output problems if the program you're converting contains PRINT statements with commas in them. You'll recall that in your BASIC, a comma between PRINTed items means that a zone of 10 spaces is to be allowed for each item. Since other computers have different zone widths built into their BASICs, a textbook program may not assume a zone width of 10. 🐾 So to avoid the possibility of garbled output, check out the listing before you type it in.

You may run into another minor problem with output: the nonstandard display of numbers. You'll recall how your Commodore puts an extra space before and after a numeric value when it prints them out on the screen, so that if you type this:

```
-----  
DY = 31: MN$ = "MARCH": YR = 1978  
HR = 8: MN = 33: AP$ = "AM"  
PRINT "BORN";DY;MN$;YR;"AT";HR;" ";MN;AP$  
-----
```

🐾 For example, Apple and TRS-80 computers assume a zone width 16, IBM-PC assumes 14, and Atari assumes 10.

the computer will display this:

```
-----  
BORN 31 MARCH 1978 AT 8 : 33 AM  
-----
```

This example shows the positive and the negative side of the automatic spacing feature: the date is spaced properly, but the time looks funny. Standard BASIC doesn't insert both those spaces around a number, so the output from these statements on many computers would look like this instead:

```
-----  
BORN31MARCH1978AT8:33AM  
-----
```

If you want the spaces in the date using standard BASIC, you'd have to include them in the PRINT command, like this:

```
-----  
PRINT "BORN ";DY;" ";MNS;" ";YR;" AT ";HR;" : ";MN;  
"TO";AP$  
-----
```

A statement like this would produce extra spaces if you used it as is on your Commodore. Consequently, if spacing is important, you'll have to systematically remove the extra blanks in PRINT statements when you encounter them in generic BASIC programs. It's not so easy to take the blanks out of the time in this example. If you were so inclined, you could print the line without the colon and sneak that in later with another PRINT and some fancy cursor control, but it's usually not worth the bother. If you're not a perfectionist, just leave out the colon altogether.

Speaking of colons, the multiple assignment statements in that last example aren't really standard either. Some BASIC use of backslash (/) to separate multiple statements on a line. Others combine two assignment statements into one, as in this statement for setting X and Y to 0:

```
-----  
110 X=Y=0  
-----
```

Neither of these options will work on your Commodore, but that's OK. You know how to convert them if you see them, and

most generic BASIC books avoid multiple statements on a line, anyway.

String variables are another area where different systems tend to disagree. While most BASICs (and therefore, most textbook BASIC programs) handle strings in much the same way your Commodore does (except for a few functions), you should be aware that some BASICs require the programmer to specify the maximum length of strings with a DIM statement. This can cause great confusion when translating programs. Consider the statement

```
-----  
210 DIM N$(11)  
-----
```

On your machine, this sets up an array of 12 strings (to store the names of the twelve months, for example). On some other machines, this statement establishes just one string with a maximum length of twelve characters. Luckily, you won't find many programs that assume the second meaning. It's worth checking out before you spend your afternoon typing a program in, though. (While you're checking, make sure all of the string-handling functions in the program work the same way they do in Commodore BASIC; your User's Guide has a complete list.)

Then there's the problem of how the computer handles special cases. An example: if the variable in an ON . . . GOTO statement has a negative value, your Commodore computer will display an error message. Some books assume the computer will simply move on to the next statement in the program, as if nothing happened. Ideally, you won't run across any programs that allow this situation to happen in the first place. But in the real world, it's best to be ready for anything.

The other big subject that's likely to cause translation problems is the data file. The problem here is that there really is no standard way of handling all of the details of opening, using, and closing files. Unless you feel adventurous, it's best not to try to copy a program using files that aren't expressly written for a Commodore computer.

Translating programs written for other computers. You've got the information you need to convert most generic programs into Commodore BASIC. But what about programs written for specific computers besides the Commodore 64? Is it possible to make a magazine program work on your 64 if it was written for an Apple //e or a TRS-80 or a VIC-20? Sometimes, but this kind of thing can get technical very quickly, so it's important to know what you're getting into before you're over your head. Here are a few guidelines so that you can better judge whether the translation is worth the trouble.

First guideline: Avoid trying to translate programs that contain lots occurrences of PEEK, POKE, SYS, and USR. These four keywords are always accompanied by a reference to a specific memory location in the computer. Because no other brand or models of computer has exactly the same memory layout as the Commodore 64, any reference to a specific location in another computer is meaningless to your computer. If a program contains just one or two POKES, it might be worth the trouble to find out what those locations mean and use a memory map to find to corresponding location in the C-64 (if there is one). But unless this kind of detective work is fun for you, you're best off avoiding foreign programs that use these keywords.

Second guideline: Avoid programs that contain many keywords you don't recognize. Many computers have special home-brewed BASIC statements for manipulating special features of their computers; Atari's SOUND statement is a good example. Unless you're intimately familiar with how those statements work, you'll have trouble duplicating their effects on the C-64.

Third guideline: Don't try to translate programs that have fancy graphics and sound features unless you know what you're doing. This is really a restatement of the first two guidelines, because most sound and graphics effects are done with PEEKs, POKES, and special keywords.

Fourth guideline: Stay close to home. Commodore PET programs can often be easily translated to run on the C-64, because of the similar design of the two machines. (PET BASIC 4.0 has a few commands that aren't available in an unadorned C-64,

but these don't often appear in program listings. 🐾 Many programs written for the VIC-20 are easy to translate for the 64, with a hitch: the VIC has a 22-character wide display, so the output from VIC programs often looks strange on the 64. But neither the PET nor the VIC have the same memory layout as the 64, so you'll still have to watch for PEEKs and POKEs. (One other advantage of using programs written for Commodore computers is that all Commodore machines use the same disk and tape format. That means that you can take a program saved on disk or tape by a VIC or a PET and load it directly into your C-64 without retyping it.)

Programs written for the Apple II, the Atari, and the TRS-80 can often be translated without major rewrites, as long as they don't get fancy with graphics and sound.

Final guideline: If you really want to take advantage of software written for the PET and the Apple, Chapter 9 has some tips on giving your machine a whole new personality!

🐾 See *Commodore Magazine*, March, 1983, page 41, for translation details.

CHAPTER 6

THE MARVELOUS TOY: RECREATIONAL AND EDUCATIONAL SOFTWARE

*No compulsory learning can remain in the soul...
In teaching children, train them by a kind of game,
and you will be able to see more clearly the natural
bent of each.*

—Plato, The Republic, Book VII

Everything is educational. Whether we're aware of it or not, we're learning all of the time, whatever we do. We're built to react to a changing outside by changing inside. The question isn't whether children (and adults) are learning something from computer games; the question is, what are they learning? The answer isn't always as obvious as it seems. Even programs designed to teach specific skills have hidden lessons the programmer never thought of, for better or for worse.

It's just as true that any educational program has an aspect of entertainment to it. You don't have to be glued to the edge of your chair in suspense, or fall off of it in stitches to learn something, but you should be awake and involved. So the line between education and entertainment is sometimes a fuzzy one.

This chapter will present some of the best of the entertainment programs and educational programs currently available for the Commodore 64, including many hard-to-classify, borderline cases. Of course, a chapter like this can only look at the tip of the iceberg, and no doubt this iceberg will have grown tremendously between the time these words went to print and the time you read them. But at least it's a start, and there are plenty of periodicals to tell you about the hot new items that hit the racks every day (See Appendix A).

If you're interested in your computer as an educational tool, you'll want to also look at the material in the next three chapters from that point of view. Chapter 7 describes word processors and spreadsheet programs, which can be powerful tools for developing skills in writing and number manipulation, among other things. Chapter 8 discusses program design, and profiles two programming languages that were designed specifically as educational tools. Chapter 9 looks at timesharing utilities, which can be used as electronic reference libraries (and long-distance game machines). Chapter 9 also peeks at some hardware attachments, present and future, which can make your computer into a musician's instrument, an artist's canvas, or a voice in a platonic dialogue!

A TAXONOMY OF COMPUTER GAMES

Video Games. Back in the days when most of us thought of computers as the things that messed up the phone bills, hackers in the computer centers of large universities were using the multimillion dollar computer systems to play what may have been the first animated computer video game: Space War. In this game, the player piloted a space ship through three-dimensional space (the video screen serving as the porthole of the ship), trying to defend his vessel from attack by others. Not long after the birth of Space War, bars and restaurants across America were being quietly infiltrated by quarter-eating Pong-machines that pitted people against each other in fast-action bouts of knob-turning. These early scouts apparently found Earth habitable, because they've been multiplying like mad ever since.

Today it's next-to-impossible to find a pong game anywhere—we're too sophisticated now to be happy beeping a little light around with a flat paddle, and our computers are, too. The Commodore 64 has incredible potential as a video game machine, and software manufacturers are realizing that potential,

most of them, anyway. There are still lots of games being sold that are not fit for public consumption. Some have mediocre graphics and sound; others are poorly designed; still others have errors in their programs that cause them to abort at inappropriate times. When you buy one of these games, you can't help but wonder if anybody tried it out before they sold it.

Computer video games aren't cheap, and it's definitely not fun to spend your money on a bad one. So it would be nice if I could list all of the good ones for the Commodore 64 here. But video games are as transitory as hit records. Any list of favorites is bound to be out-of-date a few months after it's compiled, as new titles edge their way into the charts. So I'll start by discussing what to look for in a game. Later in the chapter you'll find a list of some of the most trustworthy of the game manufacturers, with descriptions of some of the current titles that are likely to pass the test of time.

Video games and hit records are both designed, above all else, to appeal to that part of us that's willing to spend money for fun. So a quick look at the record industry can give us some insights into what makes a good video game. To break into the top 40, a record by an unknown performer must have a "hook"—a melody, lyric, or rhythm that reaches out and grabs listeners the first time they hear it on their car radios. (A million-dollar promotional campaign doesn't hurt either.) Hooks are usually simple, because we're most responsive to new input when it's simple. We tire of simple hooks quickly, though, so to stay on the charts a record needs to have something more. The ideal hit record, then, is simple enough to grab our ears immediately and yet complex enough to keep our interest after repeated listenings.

The same psychological principles apply to the marketing of video games. A new game probably isn't going to catch on if it takes four hours of practice to learn the basic skills needed to play. It has to grab our imaginations immediately if it's going to take our money and run. On the other hand, once a game is mastered, it's not much fun. (Remember tic-tac-toe?) So a good game should be easy to enter but hard to conquer. One way to resolve this dilemma is to have many levels of difficulty,

starting with a learner level. This is the classic arcade technique: the longer you play, the tougher your electronic opponent becomes. Some of the best games are really many games in one, with each new level demanding a new set of skills and strategies. Home game designers, who aren't bound by the constraints of the arcade economy, are free to improve on these techniques by building into their products features like these:

- An optional learner level that allows you to learn the ropes in a simplified, no-pressure setting;
- A choice of entry levels, so that the experienced player can skip immediately to the real challenges;
- An interrupt button that allows you to freeze the action while you answer the phone or eat your lunch. (This might be considered a safety feature; there's been at least one reported homicide because of an interrupted arcade game!)

If a video game is going to offer continuous entertainment to different members of the family, it should have some or all of these options. Many companies offer these features routinely in all of their games, making it easier to shop by brand name. Of course, none of these features can make a boring game interesting, and the game's the thing. Brand-name shopping still isn't out of the question, though, for two reasons. First of all, the best companies never put their logo on games that haven't been thoroughly user-tested. And secondly, a really good game is typically cloned by several companies, with just enough changes to keep the copyright lawyers away.

Finally, when you're choosing a video game, there's another hard question you may want to ask: what does a player come away with? This is really two questions, one of values and one of skills. The values question can take countless forms, depending on what's important to you. Some examples: is the player being conditioned to react to problems by pulling triggers, or to think of women as empty-headed sex objects, or to accept nuclear holocaust as a way of life?

The question of skills may be easier to deal with. A video game can be a great escape into a world where eye-hand coordination is the only thing that counts, and the action is so fast-paced that the troubles of the real world don't exist. Or it can be an intellectual challenge demanding a tremendous amount of strategic thinking. There's something to be said for — and against — both extremes, but the best video games are usually somewhere in between.

Strategy Games While there's no clear line between video games and strategy games, strategy games generally place less emphasis on fast action and dazzling graphics, and more on logic. Many of these games are screen versions of games from the pre-computer era, like chess, poker, and mastermind. Others are designed to simulate sports, war, stock exchanges, and other human institutions where planning and strategy are important. The classic computer strategy game is *Star Trek*, which appears in hundreds of versions on computers of all sizes. This low-graphics high-strategy game makes the player into a starship captain whose mission is to rid the universe of the dreaded Klingon menace. Computer strategy games are generally easier to program than arcade-style games, and they've been around longer, too. Consequently, there are good ones available with very reasonable price tags. Some of the best public domain programs (Chapter 5), including several versions of *Star Trek*, fall into this category.

In order to avoid being disappointed by a game of this type, it's important to know what to expect. If you want graphics and sound, make sure the product you're considering has those, preferably through first-hand experience. A surprising number of strategy game programs have very limited graphic displays; some are all words. That may be all you need, but then again, it may not be what you want. Another question to ask is whether the game allows multiple players. Are you playing against the machine, or against other humans? The best games usually let you choose. Finally, many of the questions raised in the section on video games are relevant here too.

Adventure and Fantasy Games A look at computer games wouldn't be complete without some discussion of those games that have come to be known by the generic name of



adventure. These games are descendants of a game by the same name that was developed on mainframe computers during the golden age of hacking. The classic adventure game is a kind of interactive fantasy mystery, in which the player types simple English commands to the computer, to maneuver his way through the caverns and tunnels of a labyrinthine underground world. This world is full of hidden treasures, hazardous passages, mischievous trolls, fire-breathing dragons, and assorted medieval surprises. It can take hours or days to puzzle all the way through a good adventure, and many people find the experience hopelessly addicting.

To feed this addiction, programmers have cranked out hundreds of adventure-style games for microcomputers. Many of today's adventure games are considerably more sophisticated and varied than their ancestors in their ability to understand English. ZORK I, one of the best of the modern adventures uses only prose to guide the adventurer.

Many adventurers claim that the lack of graphics in this kind of game is an asset for two reasons. First, pictures take a lot of memory, so a prose-only game has more room for complicated structure. And more importantly, the pictures in your imagination can be more vivid than any a computer could produce. Still, graphics are important to many consumers, and many companies are responding with adventure-style games that have graphics and even animation. Often, the graphics are little more than top-view maps of portions of the maze; others are approaching the dazzling graphics of a good video game.

Since most adventure games are impossible to complete in one sitting, good programs typically allow you to freeze a game in mid-adventure and save it on disk or tape for completion at a later time. This feature also allows a crafty player to save the game before a crucial decision or battle, so that he has a second chance if things don't go his way!

Adventure games aren't limited to mythological metaphors. Some take place in outer space or on other planets; others are classic pulp murder mysteries in which you play the detective. The common qualities of all these games are the spirit of exploration and experimentation, and the reliance on head rather than hand as the chief gaming tool.

GAMES BY NAME: A SAMPLER

Of course, the best directive for choosing any game is to play before you pay. If you can't do that, you might be able to benefit from this list of recommended companies and products. (Games with a more educational slant and games for the very young are also discussed in the next section.)

● **Artworx Software Company, Inc.** Artworx makes a variety of games for a variety of computers. Their most notable title for the Commodore 64 is BRIDGE 4.0, a computerized contract bridge simulation. The program serves as the dealer, your partner, and your two opponents. You can let the computer deal randomly from a shuffled deck, or redeal the previous hand as in duplicate bridge. The program follows most standard bidding conventions, plays well, and almost never kicks you under the table.

● **Avalon Hill Game Company.** Avalon Hill has been producing board, strategy, and (especially) war games for over 25 years, and has recently ventured into computer games in a big way. Some of their early software entries were simply computerized transcriptions of their more popular board games, without sound or graphics. But their more recent products include several video games and some adventure and strategy games with graphics and sound.

One of their best is TELENGARD, a classic *Dungeons-and-Dragons* style game which allows you to look down from above on part of the 50-level labyrinth. Unlike many adventure-type games, this one requires you to think fast to avoid the many hazards. The Commodore 64 version of TELENGARD has some fine sprite graphics and sound effects not found in other versions of the game.

● **Broderbund Software.** Broderbund produces a large and varied line of popular recreational software for many micros. Many of their best titles for the Apple have been translated to run on the Commodore 64, with virtually every detail the same as in the originals. When it comes to game design and graphics, that's a plus—these games shine in those respects. But it sure wastes the C-64's sound chip to have it simply mimic the primitive sounds that an Apple is capable of making.

Broderbund's best-selling game is CHOPLIFTER, a save-the-hostages joystick game which makes you a helicopter pilot on a midnight rescue mission. Points are scored, not for shooting down attacking planes, but for returning rescued hostages. If your first chopper is shot down, a second one moves into action,

with a third waiting in the wings. The game ends when you've been shot down three times or when the last of the 64 hostages has been killed or (hopefully) rescued. So it's possible to play this game without feeling like you finished in failure. Graphics are excellent: the little captives running across the desert almost seem alive.

Another winner from Broderbund is DAVID'S MIDNITE MAGIC. If you've ever dreamed of having your very own pinball machine, this might be the game for you. DAVID'S MIDNITE MAGIC is an extraordinary realistic keyboard-controlled simulation of a pinball machine, complete with all of the features that make pinball so seductive. The program allows up to four players per game, and saves the ten best scores on disk between games. It's a good thing the C-64 has a sturdy keyboard; you're likely to give the slipper buttons a beating with this one.

One of Broderbund's most unusual games is SERPENTINE, a sort of mating of Pac-Man and Frogger. The joystick controls a blue snake as he slithers through one of many mazes occupied by three green snakes and an occasional frog. I won't go into the details of the reptile-eats-reptile plot, which involves births as well as deaths. Suffice it to say that it's easy to get hooked on this one.

Finally, LODE RUNNER, a chutes-and-ladders game for creative gamers. On the surface, LODE RUNNER looks like a slightly inferior copy of Epyx's JUMPMAN (see below), a multi-screen sci-fi game with roots in the arcade machine DONKEY KONG. But a closer look reveals that LODE RUNNER has 150 different screens to play. And if that isn't enough for you, the program includes an easy-to-use game editor that allows you to create your own variations on a separate data disk, even if you know nothing about programming. A ladder here, a gold chest there — have it your way. After you've designed a screen, you can play-test it and fine-tune it until it's perfect. As you might imagine, there's lots of education hidden in this process — education that goes way beyond the three R's. LODE RUNNER is one of the first of a new breed of computer games that lets the player be a creator.

(Broderbund also markets Bank Street Writer, discussed in the next chapter.)

● **CBS Software.** Even CBS has its big eye on the software market. Their first game for the Commodore 64 is a whodun-it called MURDER BY THE DOZEN. It's really 12 interactive mysteries in which you play detective and try to uncover evidence and put facts together to solve crimes. You can do this alone, or as part of a team, or in competition with up to three other sleuths. This isn't a prose mystery like Infocom's Witness (see below); instead of talking to the computer in English, you respond to multiple-choice menus. The computer is just one of the tools you'll use in your work; the package includes a book of clues, a pad of worksheets for note-taking and map reading, a booklet containing solutions to the 12 crimes, and a transparent red "decoder" that lets you read those solutions.

● **Counterpoint Software.** Counterpoint specializes in educational games for children, some of which are discussed later in this chapter. Their first game for the older generation is a trivia-lover's delight called QUIZAGON. Counterpoint has packed the game program with over 6,000 questions and answers on two disks — questions about science, geography, sports, entertainment and arts. Your skill at answering these questions determines how well you do in the game. One to four players can play at a time, but playing solitaire isn't as much fun as playing in groups. This is a good game for introducing the computer to keyboard-phobics; the space bar is the only key they'll ever need to press.

Quizagon is like a computerized board game. The screen is divided into hexagons with a question hidden under each one. Players move around the screen, answering questions and trying to catch all four Quizards, indicated by flashing hexagons. If you like answering questions like "In TV's 'I Love Lucy,' what is Lucy's maiden name?" (McGillicutty) you'll enjoy Quizagon. (Counterpoint promises additional Quizagon data disks with questions on various subjects in case you memorize the answers to the first 6,000.)

● **Creative Equipment, Inc.** Creative Equipment has a small but growing line of games for the C-64, all in convenient cartridge form. TSI MAZE MAN is a single-maze Pac-style

game with consistently fast action and a couple of twists on the familiar plot. TSI LAZER CYCLES is an embellished version of the inside-the-computer cycle scenes from the Disney film *Tron*. (You have your choice of racing against the not-very clever computer or another human, clever or otherwise.)

But the most interesting—and frustrating—game from this company is TYLER'S DUNGEONS, a unique cross between Pac-Man and *Dungeons-and-Dragons* with excellent graphics to boot. You're looking down at an angle on a maze at the start of this game. Your joystick maneuvers a three-legged creature through the maze, which by itself isn't too tough. But you're looking at one of 250 interconnected maze-rooms containing giant snakes, collapsing walls, elevators, and deadly lurks that prowl the corridors incessantly in search of you. Your mission is to avoid the hazards long enough to locate eight treasures in this giant labyrinth. The reward? A message telling you how to enter a random drawing of winners for \$500 on April 1, 1984. Better hurry!

● **Creative Software.** Creative Software (not to be confused with Creative Equipment) specializes in software for the VIC-20 and, more recently, the Commodore 64.

One of their first cartridge products for the C-64 is also one of the most creative new games I've seen for any computer. Even the title screen is unique: "moondrop ships" lazily trace the name of the game, MOONDUST, leaving trails of tinkerbell sparkles behind, while strangely hypnotic music echoes the motion on the screen. The title fades into a menu of playstyle options: beginner, evasive, freestyle, and spinsanity. Whichever you choose, your joystick will move a cherub-like "spacewalker" among the moondrops as they leave swirls of colorful moondust. But your joystick also controls the ships, so any move of the spacewalker is gracefully echoed in different forms by everything on the screen. Even the electronic soundtrack is controlled by the joystick, going up or down with spacewalker; changing modes with sideways moves. You may get so caught up in creating exotic patterns of color and sound this way that you'll forget about the game. There is a game which involves dropping seeds for the ships to smear into circular energy fields

while avoiding collisions with those same ships. It can be as challenging as you like, depending in part on which playstyle you choose. But the game feels more like ballet than combat, and the mood is more like *Fantasia* than *Tron*. Moondust is a computer game for people who don't like computer games.

Creative Software also markets TRASHMAN, another Pac-alike game in which your joystick-controlled garbage truck travels through a maze picking up litter dots while avoiding four deadly flies. This cartridge game has only one maze, but several levels of difficulty, starting with a very slow speed for young munchsters.

Another popular game from Creative is SAVE NEW YORK, in which you're charged with stopping the waves of alien monsters before they devour the entire skyline of the city. This would be just another shoot-the-mutants game, except that this particular game allows two players to play at the same time, encouraging cooperation in the face of destruction.

(Creative Software's PIPES is reviewed later in this chapter.)

● **Datasoft, Inc.** Datasoft is another software company that's beginning to release games for the C-64. Their first 64 title is MOON SHUTTLE, a multi-level shoot-the-asteroids-and-aliens game that's technically well done. In the works is a 64 version of the popular arcade game POOYAN, which pits a mother pig armed with a bow and arrow against a pack of hungry wolves in hot air balloons. Probably based on a National Geographic TV Special.

● **Epyx Software.** Epyx has a reputation for producing high-quality adventure games for many brands of micros. Their TEMPLE OF APSHAI series includes several challenging graphic adventures that are extremely popular among *Dungeon*-lovers. But more recently, Epyx has branched out into producing arcade-style games and educational software.

The care that goes into their products is obvious in JUMPMAN, a platforms-and-ladders game vaguely reminiscent of DONKEY KONG, the arcade classic. JUMPMAN is the current favorite of our local user group, and with good reason: it's really

30 games in one, with seemingly endless variations on the simple jumping theme to keep you interested. Mastering JUMPMAN involves more than just joystick agility; many of the screens are tricky logical puzzles that require strategy, forethought, and practice to conquer. JUMPMAN players (from 1 to 4) can choose their speed and the difficulty level (beginner, intermediate, advanced, or random). High scores are saved on disk and displayed after each game. The different screens are stored on disk, too, so you have to wait a while for each succeeding level. But you'll wait.

If you don't like waiting, or if you don't have a disk drive, or if you've mastered all 30 Jumpman screens, you'll be happy to know that there's a cartridge-based sequel called JUMPMAN JUNIOR, with twelve more screens to challenge you.

When you tire of jumping around, you can come down to Earth with PITSTOP, Epyx's road-race game. Up to four players can race (one at a time) on any or all of six different racetracks. A joystick is used for steering and acceleration. The screen shows a blimp's-eye view of the whole track in the corner. The unusual part of this game is that you have to occasionally do a pit stop to change tires and refuel, using the joystick to move the members of the crew around. This necessary stop slows the action down, but it adds an element of strategy (Can I make it around again without blowing out that right-front tire?) to what would otherwise be just another eye-hand coordination game.

● **Human Engineered Software (HesWare).**

HES Produces a wide range of software and hardware products for Commodore machines (see Chapters 7, 8, and 9), including some first-rate video game cartridges.

One of their most popular HesWare games is GRIDRUNNER, a multi-level shoot-the-invaders game that looks a little like Centipede, the arcade hit. This battle supposedly takes place on a solar collection grid orbiting Earth. But forget the plot—fast action is the name of this game. (The programmer who created Gridrunner has developed a sequel for advanced players called ATTACK OF THE MUTANT CAMELS. Sounds like a candidate for a TV network mini-series.)

(Be sure to read about HesWare's BENJI'S SPACE RESCUE in the next section, too.)

● **Infocom.** Infocom produces some of the best-selling games in the industry, and they do it without the aide of sound or graphics! The ZORK trilogy sets a new standard of excellence for prose-style adventure games. Built around a special language developed by MIT artificial-intelligence students, any of these disk-based programs can almost make you believe that the computer knows English. Until you ask a simple and obvious question it can't understand, reminding you that you're playing with a dumb machine, after all. It takes persistence, cunning, logic, a little bit of luck, and a lot of time to complete one of these games. And if that's not enough, you can get hints and maps from the Zork User's Group, an independent club of Zork fanatics.

If trolls and treasures don't tickle your imagination, you might get hooked on one of Infocom's participatory science fiction thrillers (such as STARCROSS and SUSPENDED) or murder mysteries (like DEADLINE and WITNESS) built around the same prose-generator. Whichever you choose, you have to stay constantly alert. Witness this scene from WITNESS, a pulp mystery in which you play the detective:

Linder sits down in the carved chair.

>TELL ME ABOUT YOUR WIFE

"It's still too Painful for me to talk about, I'm afraid."

"If you'll just take a chair, I'll explain what this is all about."

The rain outside is falling heavily now.

>WAIT

Time passes. . .

Linder says, "I repeat: if you'll just take a chair, I'll explain what this is all about."

Do you want to keep waiting? (Answer YES or NO.) >NO

It's now 8:19 p.m.

>SIT DOWN

(What do you want to sit down on?)

>THE CHAIR

(Which chair do you mean, the wooden chair or the carved chair?)

>THE CARVED CHAIR

Linder glares at you. "I meant that you should sit in the customer's chair, not my lap!" You are on your own feet again.

(Infocom has an agreement with Commodore by which many of their games, including all of those mentioned here, are marketed under the Commodore logo.)

● **Screenplay** Screenplay does all kinds of games: arcade games, adventure games, simulation games, and educational games. Their early releases for the Commodore 64 are high-quality products. Some are packaged with Atari and Commodore versions on opposite sides of the disks or tapes.

WARRIORS OF RAS is yet another *Dungeons-and-Dragons*-style game series. DUNZHIN is the first — and easiest — in the series. It features view-from above graphics, game-save and character-save options, and a maze that changes every time you play.

POGO JOE is a creative variation on the bounce-around-changing-surface-colors-while-avoiding-other-bouncers theme popularized by the arcade game Q-BERT. Instead of a pyramid, POGO JOE bounces around on any of more than 60 different screen patterns of increasing difficulty. This game has excellent graphics and music, and offers players lots of options: number of players (one or two), starting screen, Joe's speed, monster's speed, etc.

KEN USTON'S PROFEE\$SIONAL BLACKJACK is aimed at a different kind of audience, with the monikers "intelligent statements" and "Grown-Up Gameware" on the package. This is no simple play-cards-with-the-computer program, although you can use it that way if you want to. This is a program for folks who take their blackjack seriously. For each game, you specify which casino's house rules will prevail (you can design your own if you like), how many players (between one and seven, any of which can be real people or computer stand-ins with different strategies and bankrolls), and how much help you want (anywhere from none to on-screen prompts telling you exactly what to do in each situation). The program has drill-and-practice options for developing your skills at card counting and playing particular kinds of hands. It's packed with a 48-page booklet explaining different blackjack strategies, and a coupon for Mr. Uston's big book, *Million-Dollar Blackjack*. (The package says the book is free; the coupon says it's \$2.50.)

● **Sierra On-Line** Sierra is one of the biggest software publishers in America, producing all kinds of games, a growing line of educational software, and a few business products. Their games have long been popular with owners of many different computers, and several titles have recently been released for the C-64.

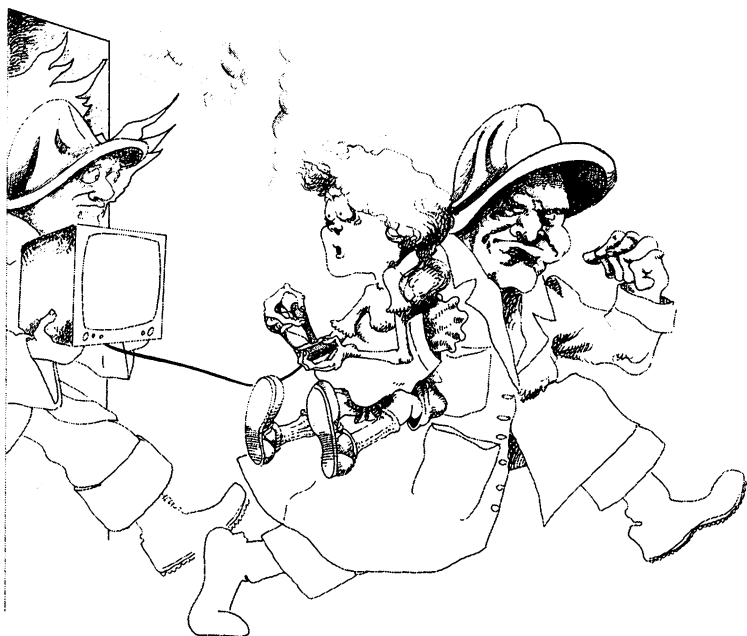
FROGGER is a faithful recreation on disk of the arcade classic in which you maneuver frogs across a busy highway and a river full of hazards, picking up an occasional hitchhiker along the way for extra points. Even the lighthearted music from the coin-op version is here. This version has three levels of difficulty, keyboard and joystick control, a choice of speeds, and a control for turning off the music without losing the sound effects.

JAWBREAKER is a variation on the Pac-Man theme aimed at young gamers with sweet teeth. The joystick steers a set of chattering teeth around a screenful of munchable dots, sweet treats, and rolling happy-faced jawbreakers that can instantly shatter the teeth—unless, of course, they've been energized. But instead of a maze, the playing field is a series of five rows separated by walls with constantly-moving gates. The player can choose the game's speed, from teddy-bear to speed-demon, and turn the carnival music on or off. (Parents, take note: there's a brief interlude with a toothbrush after each screen is cleared.)

Joystickers who aren't thrilled or challenged by JAWBREAKER might prefer SAMMY LIGHTFOOT. This one-or-two-player game captures the cartoon spirit and graphic style of Donkey Kong without being a simple donkey clone. Sammy is a joystick-controlled acrobat who maneuvers his way through three different screens full of trampolines, rope swings, fire pits, giant circus balls, moving platforms, and flying carpets, among other things. If he makes it through all three screens, he starts over with more obstacles than before. Instructions are sparse, because part of the fun of this game is figuring out what makes Sammy run, jump, climb, swing, fly, and dance.

Finally, if your arcade tastes run toward pyramid hopping with Q-BERT, you might try MR. COOL. In this cartridge game, the

loose-tongued Q-BERT creature has been replaced by Mr. Cool, an ice cube that can be instantly melted by passing fireballs and hot springs (bouncy, not wet), except when he turns super-cool. Like Q-BERT, MR. COOL hops around on a triangle of colored faces, changing their colors until they match. Each success brings a new round; 25 successful rounds brings a faster level with 25 more rounds, and so on.



● **Sirius Software.** Sirius has a large variety of games for many computers. They have several popular games for the 64, ranging from SNAKE BYTE, an unusual game with a keyboard-controlled snake and so-so graphics, to REPTON, a Defender-like space shoot-em-up with excellent visuals.

One of their most popular titles is SQUISH 'EM, in which you use your joystick to help a cute little guy climb to the top of a 48-story building to retrieve a suitcase full of money. Along the way, he encounters all kinds of not-very-friendly creatures,

which he may either avoid (if he can) or stomp flat. Stomps give extra points, but stomped creatures that pop back up (they all do, in time) can't be re-squished. The game starts easy enough that a 5-year-old can play. Joystick junkies can skip the easy levels and go straight to the tough stuff.

BLADE OF BLACKPOOLE is Sirius' first adventure game for the C-64. It's a graphic adventure, in which each move brings a new view onto the screen, as if you were taking still photos to record your trip. If **ZORK** is too wordy for you, give this one a try.

(Sirius' excellent **TYPE ATTACK** is described later in the chapter.)

- **Spinnaker Software.** Spinnaker specializes in learning games, including games for the very young. Many of their programs are discussed in the next section.

- **SubLOGIC, Inc.** SubLOGIC is in the process of translating their most popular Apple and TRS-80 games so they'll run on the 64. Their first translation is **NIGHT MISSION**, another excellent Pinball simulation with graphics comparable to Broderbund's **MIDNITE MAGIC**. As a bonus, this program lets you go inside the machine and change the color scheme, ball speed, table tilt, and all kinds of other parameters. (If you're looking for a non-violent game, don't stop here. **NIGHT MISSION** has a bombs-away theme.)

SubLOGIC also plans to release a C-64 version of their popular **FLIGHT SIMULATOR**, which makes your computer screen into the control panel and windshield of a Piper 181 plane on practice flights or WWI battle missions.

- **Synapse Software.** Synapse makes games to satisfy even hard-core arcadians. Their first games for the C-64 are all packaged like pulp science fiction novels, with flashy battle pictures on the front and comic book hype on the back. The games themselves are definitely not pulp. They're clever programs with impressive graphics and sound effects, challenging scenarios, and plenty of variety. (And plenty of violence, too.)

One big Synapse hit is FT. APOCALYPSE, a sci-fi shoot-em-up vaguely resembling the arcade Defender. You pilot a futuristic helicopter in this one, defending yourself against all kinds of dangers and rescuing prisoners as you descend level-by-level into the depths of an underground cave-fortress. Graphics and sound effects in the upper levels are impressive. I can't judge the lower levels, because I've never been able to get that far, even with the game set to its easiest play level. (Must be my joystick.)

If piloting a helicopter through caverns makes you claustrophobic, you might be more comfortable with SURVIVOR, which takes place in the wide open space of space. As surviving pilot of a small round star cruiser, your job is to destroy four gigantic enemy space forts. This game is unusual in that you can play it solo or as a team of two, pilot and gunner.

A completely different kind of program is SHAMUS, a graphic adventure-style game that requires arcade-style reflexes. The hero is piloted and defended by joystick as he goes from room to room looking for trouble, which isn't hard to find.

● **Timeworks, Inc.** Timeworks makes a variety of programs for the C-64, including many games, educational and otherwise. Two of their entries, DUNGEONS OF THE ALGEBRA DRAGONS (a math education adventure) and WALL STREET (a stock market simulation) are described in the next section.

● **UMI (United Microwave Industries, Inc.).** UMI has a number of good games for Commodore computers on the market. Two of their best for the C-64 are computerized recreations of ancient board games.

Chess players in search of opponents can find satisfaction in GRAND MASTER, an excellent chess simulation with several skill levels, from beginner to expert. The program has several features that make it an excellent learning tool, including an option for taking back moves and another for asking the computer for hints.

Another mind-challenging game from UMI is RENAISSANCE, an electronic version of the thousand-year-old game that's marketed today under trade names like OTHELLO and REVERSI. Like GRAND MASTER, this program is designed with a variety of skill levels and features to make it interesting and educational. (One particularly handy feature lets you change places with the computer when things look bleak on your side.)

● **Commodore.** Who? With all of the quality games being produced by software specialists, it's easy to forget that Commodore sells games, too—and usually at very low prices. But the Commodore logo on software doesn't give much of a clue as to the type or quality of the product, because Commodore buys its software from a variety of suppliers, in addition to producing some of its own. So you'll find the big C on adventures like ZORK (made by Infocom) as well as exciting arcade-style cartridges like LE MANS, a view-from-above race car simulation in which you use a game paddle to steer your car along a fast track with ice, tunnels, and other hazards. Unfortunately, you'll also see it on several mediocre products, most of them rushed to market to fill the initial software void. Test drives are recommended.

● **Public Domain.** Don't forget the public domain programs when you're hunting for games. Many of the classics are available for little or no cost: strategy games like Star Trek, Pong-style games like Breakout, word games like Hangman, and logic games like Mastermind. Since they're (mostly) written in BASIC, don't expect fast action or dazzling graphics. But the price is certainly right!

(A problem with some of the Commodore Public-domain arcade-style games is that they were originally designed to use the Pet's numeric keypad as a pseudo-joystick. Since the 64's number keys are all lined up in a line, it's very confusing to try to move a player around on the screen with them. If you want to run a game that uses this keypad gimmick on your 64, you might want to rewrite the program so that it looks to the joystick instead of the number keys for directions. If you have no joystick, you could use any block of 9 adjacent keys, such as T-Y-U-G-H-J-B-N-M, to serve the same purpose. Or, if program

surgery doesn't appeal, you can buy a numeric keypad attachment for your 64; one is listed in Chapter 9.)

AUTOMATIC LEARNING: A SURVEY OF POSSIBILITIES

Whatever else is said about them, arcade-style games can improve communication between hand, eye, and brain. Strategy games and simulations help develop logical thinking, planning, and patience, and can broaden a player's perspective. Fantasy and adventure games can do all of that and stimulate the imagination, as well. But these are side effects of programs that were designed for fun. Let's turn our attention to software that intentionally teaches us something, with fun as a side effect.

There are hundreds of educational programs available for the Commodore 64. While many of these programs are outstanding teaching tools in the classroom, most are of little interest on the home front. The great majority are some form of drill and practice, which, in its simplest form, means electronic multiple choice. In school, this kind of thing is an exciting alternative to workbooks. At home, where there are more choices, it's not likely to hold anybody's attention without being jazzed up considerably. And, unless it's a flexible program, it probably won't be of much use beyond the first couple of sessions. In short, would you pay 50 dollars for a program of drills in elementary geographic vocabulary? And if you did, would your kids use it more than once? 🐾

But there are a growing number of educational programs that are well-suited for use in the home. What sets these programs apart from the rest? For one thing, they're designed to teach without the aid of a teacher to set things up, fill in the holes, evaluate progress, and motivate. A good home educational program is easy to use, self-contained, and FUN! In addition, it should stand the test of repetition—is it still entertaining AND educational after the first session? Many school programs, designed to be used once by each student, are neither. Other

🐾There is a place for this kind of use-it-once program outside of the schools: Public libraries.)

programs may be fun to use again and again, but they teach very little after the first five minutes. A good example is Commodore's Tooth Invaders, an arcade-style battle between Plaqueman and D. K. Germ (approved by the American Dental Association).

Finally, when you're looking for educational programs, remember two things: (1) kids almost universally prefer graphics and animation to text, and active involvement to passive reading; and (2) we tend to learn what we're rewarded for learning. These two facts can be used together to design powerful learning tools. But an amazing number of supposedly educational programs were apparently designed by programmers who never studied educational psychology. Consider, for example, the public domain program called ENGLISH MONSTER. The program is designed to teach word association via multiple choice. At the beginning of the game, the screen shows a pair of innocent-looking children on one side of a wall, and a sinister-looking giant labelled "teacher" on the other. When the student types a correct single-letter response, the computer responds dryly with a pair of exclamation points. If the student chooses a wrong answer, a piece of the wall disappears. Finally, if the student perseveres and makes enough mistakes, he gets to watch the animated monster gobble up the children! Class dismissed.

The programs listed here are among the best currently available for the Commodore 64. With better programs being released all the time, some of these pioneering efforts will likely be overshadowed or replaced in the future. Keep an eye open. (There are more educational tool/toys in the last two sections of this chapter and in Chapter 8.)

PROGRAMS FOR THE VERY YOUNG

If the idea of software for preschoolers seems bizarre to you, you're not alone. Until recently, there was no such thing. But parents, educators, and software writers have discovered that little kids take to computers at least as easily as their older brothers and sisters — if the programs are right. It just takes a

few minutes watching a 4- or 5-year-old playing with one of these programs to see the amazing educational potential of the computer. Sesame Street can only talk to kids; computers can listen, too. If any preschoolers live at your house, you should try at least one of these programs.



● **EARLY GAMES FOR YOUNG CHILDREN**, from Counterpoint Software. If your kids are REALLY little, this BASIC program is a good place to start, because it doesn't require any reading. The menu is a series of simple pictures on the screen, each picture representing one of the nine games. When the child sees the picture representing the game she wants to play, she presses any key, and the game starts. Pressing any of the tan colored function keys brings back the menu. What could be simpler?

In two of the games, Match Letters and Match Numbers, the object is to find and press the key that matches the figure on the screen. In Alphabet, the object is to find the NEXT letter in the alphabet, with prompting, if necessary. Count, Add, and Subtract use colored blocks to teach very simple counting and arithmetic skills. names teaches the child to type her name, with a little help from Dad or Mom and the computer. Compare Shapes is a "which shape is different" game. And Draw lets the child use the keyboard to draw simple pictures on the screen by moving the cursor around (pressing any key at the top moves it up, etc). And CTRL-P can be used to save masterpieces on disk!

The graphics and sound in this program, while adequate, don't push the limits of the machine; don't expect sprites, high-resolution drawings, or musical offerings. But educationally and psychologically, it's hard to find fault with these thoughtful games. The package says they're for ages 2 through 6, but some 5- and 6-year-olds may find them a little too simple.

(Other Early Games dealing with specific topics are available from Counterpoint, including Early Games Music, discussed later in this section.)

● **LEARNING WITH LEEPER**, from Sierra On-Line, Like EARLY GAMES FOR YOUNG CHILDREN, this program is really a collection of simple games with a picture menu so the child can choose between activities without adult supervision. Here the child uses a joystick to move Leeper, a frog-like creature, to the picture on the screen representing the game of choice. The four games are Dog Count, a simple counting exercise (How many bones do I need to feed this many dogs?); Balloon Pop, a six-level find-the-matching-shape game (where shapes range from letters and numbers to abstract patterns); Leap Frog, a simple maze game that develops the eye-hand coordination needed later for writing and playing Pac-Man; and Screen Painting, a computer coloring book containing several line drawings that can be filled with colors chosen from the bottom of the screen (or embellished with additional lines).

In LEARNING WITH LEEPER, all interaction with the computer is done with the joystick and two function keys. Pressing f3 ad-

vances to the next level when there is one, or to the next picture in Screen Painting. F1 brings back the picture menu, so kidlets go back and forth between games all by themselves.

- **KINDERCOMP**, from Spinnaker Software. Spinnaker produces some fine educational software, and their preschool series leads the pack. KINDERCOMP is conceptually similar to Early Games, aimed at a slightly more mature child. The menu lets the child choose by number any one of six named games; the f7 key returns the menu. (Children who can't read will quickly learn what the numbers mean.) Draw is fancier than the Early Games version, allowing the child to control the cursor and colors with a joystick. Unfortunately, pictures can't be saved. Scribble fills a line with a typed character. In names, the child types her name (or any short phrase) and watches with delight while it darts around the screen and makes crazy patterns. Letters gives animated rewards for finding displayed letter on the keyboard. Sequence does the same for finding the NEXT number in a sequence. In Match the child finds matching patterns of shapes. You'll be amazed at your child's attention span with this program!

- **FACEMAKER**, also from Spinnaker. On the surface, this looks like an electronic version of the classic Mr. Potato Head game, where children build cartoon faces from a collection of noses, eyes, mouths, ears, and hairpieces. But there's more. Once a face is made, it can be programmed to go through an animated sequence of winks, frowns, smiles, cries, tongue-sticks, and ear-wiggles. As if that weren't enough, there's a memory game where the face goes through an ever-growing sequence of motions and the child reproduces the sequence.

The thoughtful *Note to Parents and Educators* that accompanies FACEMAKER describes it as "an educational program disguised as a game," and explains how it teaches the rudiments of programming and develops visual and auditory sequential memory, skills that are necessary for learning to spell. Even if your 5-year-old isn't interested in developing those skills, he'll love this game!

- **HEY DIDDLE DIDDLE**. Another winner from Spinnaker, this one build around classic and obscure Mother Good Rhymes.

The very young will enjoy watching the pictures and listening to the delightful music while (optionally) Mom, Dad, or somebody reads the rhymes. Kids who can read like playing the Rhyme Game, where the lines of the rhymes are scrambled, and the joystick or function keys are used to put them in order. The game can be played with one or two players, with or without scorekeeping. My 5-year-old son Ben loves it. My only gripe with this program is that there's a lot of dead time while the pictures are pulled from the disk. That will be taken care of when somebody produces a faster disk drive for the C-64.

- **ALPHABET ZOO**, from Spinnaker. This one will tackle tiny Pac-maniacs while it develops their spelling skills. The game screen shows a picture of a simple object and a simple maze with a few letters scattered around inside it. One or two players joystick their characters through the maze until they find the first letter of the name of the picture. Older kids can play the advanced version of the game, in which their maze-runners must spell the whole word. ALPHABET ZOO is both fun and educational, but the Commodore 64 version has one minor flaw: the characters don't move easily through the maze gates in response to joystick movements. Some youngsters might not have the fine motor skills necessary to make this game easy.

Reading, Writing, and Arithmetic. There are plenty of programs out there to teach the basic skills, including hundreds in the public domain (Chapter 5), many of which are very good. There aren't too many commercial programs aimed at the home market in this category — yet. Exceptions include HEY DIDDLE DIDDLE and ALPHABET ZOO, described earlier, the BANK STREET WRITER, described in the next chapter, and these:

- **PRIMARY MATH TUTOR**, from Comm*Data. This package of four programs is designed to teach basic addition and subtraction skills through drill and practice. The four programs cover beginning and intermediate addition and subtraction drills. In each program you can choose to have visual counting aids accompany the numbers; in the more complicated problems these aids demonstrate with simple animation the processes of carrying and borrowing. Sound and graphics are simple but effective.

● **MONKEY MATH**, from Artworx. This is an arithmetic drill-and-practice program artfully disguised as an arcade style game. The learner uses a joystick to control a monkey working on an assembly line of numbers. Below the monkey sits an arithmetic problem. The monkey's job is to hit the correct answer as it goes by on the belt. You can set the game to produce problems in simple counting, addition, subtraction, multiplication, division, or a mixture, at any of three different skill levels. The animation and sound are on a level with many arcade-style games, and the game is fun. Of the arithmetic drill programs listed here, **MONKEY MATH** is probably the one that most kids would choose to use.

● **PINBALL MATH**, from Taylormade Software. This is a good program for developing more advanced arithmetic skills. It lets the student choose between addition, subtraction, multiplication, and division, and then one of three levels of difficulty. Points are awarded based on the time it takes to produce a correct response, and errors are pointed out by a sprite character as they happen. For more difficult problems involving several digits (and, in the case of multiplication and long division, several steps), the program leads the learner digit-by-digit and step-by-step toward the answer. At the end of each practice set, the program gives percent-right feedback. Don't expect arcade-style action with this one, in spite of the name. But **PINBALL MATH** does a good job of teaching complicated mathematical processes. 🐾

● **FLOWER POWER**, from Softwave. This is one of the most versatile of the arithmetic drills I've seen. It's probably not the best program to start the novice arithmetic tinkerer on, since there are no counting pictures to illustrate the concepts of addition. But it's a well-designed drill-and-practice program with simple but pleasant graphics. There are lessons in addition, subtraction, multiplication, and division of whole numbers, fractions, and decimals, plus a lesson on converting decimals to/from fractions.

🐾 Some think computers and calculators will make long division skills unnecessary. (Can you remember the last time you needed it?) Whether this comes to pass or not, at least programs like this make it easier to learn.

As in good arcade games, difficulty is based on how well the student is doing. Problems keep getting tougher as long as she gets them right; wrong answers lower the built-in skill level. Graphically, the screen is a garden. Each right answer grows a flower, and each wrong one grows a weed (which is replaced by a flower when the correct answer is given). Score is kept based on speed and accuracy of response. Not as exciting as an arcade game, but definitely more engaging than worksheets.

● **FRACTION FEVER**, from Spinnaker. FRACTION FEVER is a genuine hybrid — an arcade-style educational game, where arithmetic skills, strategy, eye-hand coordination, and quick reflexes all play a part. The player moves a perpetually-bouncing pogo-stick along the ribbon-like floors of a 20-story abstract structure. Each floor has picture fractions scattered around, and a numeric fraction displayed at the top of the screen. Points are scored by pressing the fire button while the pogo is bounding over pictures that don't match the fraction. Firing the matching fraction activates an elevator to the next floor. This is tricky business: you can fall through holes that you make while zapping bad fractions, or fall off the end of a floor, or run out of time.

Even if you've got your fractions down pat, you won't master this game immediately. It takes practice to develop the right touch with the joystick, and strategy to maximize points. It's not always easy to tell at a glance which little picture is, say $7/13$.

The graphics and sound effects may not measure up to the likes of FORT APOCALYPSE, but they're better than you'll find in many arcade-style games. But what does a kid get from this program? Besides developing coordination and strategic thinking, the game gives solid visual representations of fractions, so that kids can see that $1/2$ is the same as $4/8$, and that $5/11$ is less than $1/2$. Not enough to guarantee admission to law school, but not bad for an arcade game.

● **DUNGEONS OF THE ALGEBRA DRAGONS**, from Timeworks Personal Computer Software. A sly way to get the *Dungeons-and-Dragons* freak in the family to work on his algebra,

this is an animated adventure-style game with above-average graphics. The trapped player must maneuver through a multi-level maze of rooms, passages, ladders, and trap doors; avoid or stomp on treacherous giant spiders; and find the magic keys that unlock the dungeon. The wrinkle is that the dungeon is populated with dragons that can add to or subtract from our hero's supply of gold pieces based on his ability to solve algebra equations. There are four skill levels geared to ages 14 and up, but younger kids might be motivated to tackle algebra early with this one.

● **ALGEBRA ARCADE**, from WEPCO Software (Wadsworth Electronic Publishing Company). In spite of its name, ALGEBRA ARCADE is not a fast action shoot 'em up with a little math dressing. It is an entertaining and challenging game of strategy that teaches concepts from advanced algebra and pre-calculus analytic geometry. The game field is an X-Y coordinate grid populated with ten little creatures called Algebroids and a ghost. Each player (one or two can play at a time) tries to plot a line or curve that allows the whirlwind to hit Algebroids and avoids the ghost. If your memory for Cartesian coordinates is getting rusty, this game will shine it up in a hurry.

The carefully-designed program gives you lots of control. For example, you can change the coordinates, play with or without a timer, and play with different families of equations. Where was this program when I studied this stuff?

● **WORD RACE**, from Don't Ask Computer Software (distributed by Tronix). This is a competitive vocabulary-building game in which players compete against themselves and the clock to choose correct definitions for words that are lashed on the screen. There are no fancy graphics or sound effects, but the game should be fun for anyone who enjoys crossword puzzles and similar word games. The four different levels of WORD RACE range from pre-teen to walking dictionary. The WORD RACE package also includes two name games: a historical figure quiz called Claim to Fame, and a Sports Derby trivia contest.

● **WORD ATTACK**, from Davidson and Associates. This is a more versatile vocabulary-building program than **WORD RACE**: it has options for studying words with definitions and sample sentences, doing straight multiple choice drills in either direction, or playing a simple shoot-the-right-word game. The data disk contains 675 words categorized in nine different levels from age 8 to adult. There's also an edit feature for creating your own data disk with words of your choosing. The sound and graphics of this program won't excite the arcade set, but the program does a good job at vocabulary drilling and the documentation is very complete and well-written. Still, I'd like to see software that teaches true "word attack" skills, so the student can use clues of context, root words, prefixes, and suffixes to figure out meanings of words without so much memorization.

● **SPEED READER II**, from Davidson and Associates. The Davidson folks also do a program to teach speed reading. Its design is similar to a typical rapid-reading course, with exercises to develop several related skills: reading different things in different ways; focusing on phrases and ideas rather than words; and developing new eye-movement patterns. Exercises display letters, words, and whole articles with different formats and speeds, and there's an option to customize the exercises. The thorough documentation and menu format make this program easy to use.

Typing Tutorials. Until voice-input devices are perfected for computers (don't hold your breath), typing is an important skill for any computer-literate child or adult to have. Fortunately, there's a wealth of software designed to help you learn touch-typing in the privacy of your own computer room. Here's a sampling:

● **TYPE RIGHT**, from Commodore. This diskette contains a series of 25 lessons devised by a professional typing teacher and a computer programmer working together. The lessons start by showing you the proper finger placements, and proceed through a series of graduated drills in typing sequences of letters and words, with errors displayed as they occur, and speed and accuracy feedback after each lesson. Every few lessons a word game is thrown in for fun and vocabulary-building.

There's even an optional lesson here on typing BASIC programs. The booklet that accompanies the disk contains suggestions for modifying each of the lessons (written in BASIC) to meet your personal needs. There are no flashy sound or graphic special effects here to motivate the Star Wars set, but the teaching methods are sound and the lessons are very thorough.

- **TOUCH TYPING TUTOR**, from Taylormade Software. Another popular typing tutorial for the C-64. This one shows the keyboard on the screen, and optionally puts a flashing cursor on the key you're supposed to hit; a nice feature for beginners who don't want to keep peeking under their fingers. Its lessons aren't quite as methodical and complete as TYPE RIGHT's (capital letters aren't covered except in exercises), but it has options for practice of pseudowords and English text. A good program.

- **TYPE ATTACK**, from Sirius Software. This is a first-rate arcade game that teaches typing along the way. The people of Lexicon have been invaded by a falling alphabet, and the only way you can save them is by typing those letters before they hit the ground. If you succeed, you have to deal with the invading words. Each level of the game methodically introduces new letters, until you've mastered the keyboard. Great graphics in a well-designed learning game. There's even an option for creating your own lessons.

If you already know how to type, this might be what you've been looking for: a video game you can beat the kids at! (For a while. . . .)

- **MASTERTYPE**, from Scarborough Systems, Inc. "Learn to type or be blown to bits!" trumpet the ads for this arcade-style typing teacher. Your spaceship is surrounded by enemy ships painted with letters and words. It's up to you to type those labels before you get zapped. In spite of its game-like appearance, this is a carefully-designed typing tutorial with eighteen lessons that gradually introduce new characters on the keyboard and more difficult words. There's plenty of room for flexibility here, with provisions for controlling the speed, adding the shift key to any lesson, and creating new lessons with personalized word lists.

● **LETTER MAN**, from Behavioral Engineering, Inc. The program was designed in part by one of the developers of the psychological theories of neuro-linguistic programming. It's another Pac-clone, with a maze populated by letters instead of dots. You move around by typing those letters, while monsters prowl at speeds chosen by you. The concept is great, and the game is fun, but it's rough going if you haven't done some basic training with one of the other programs here. Rather than starting you out with the home keys and building, this one throws the whole alphabet at you early on. (Fortunately, there's an option for creating your own mazes, so you could develop a simpler version for beginners, or a personalized game with player names.) Arcadians in the family will probably be disappointed with the mediocre graphics and lack of sound, but they should enjoy the game anyway.

Odds and Ends. Memory, music, astronomy, economics, programming, and plumbing, to be exact. Plumbing?

● **PIPES**, from Creative Software. Pipes is a most unusual game/learning experience for children. The plot is simple: a joystick-controlled plumber connects the houses (from one to five) in the neighborhood to a water tank, using pipes from the pipe factory located on the right side of the screen. There are all kinds of pipes in the factory—straight ones, elbows, and junctions. But quantities of each type are limited, and so are the plumber's financial resources. It takes logic, planning, spatial visualization, and patience to put together a system that won't leak when the plumber turns on the main valve. But the clever animation and sound effects help to motivate even preschoolers to stick with it. Pipes is proof that a game doesn't need monsters, ray guns, and fast action to appeal to kids.

● **THE EINSTEIN MEMORY TRAINER**, from Einstein Corporation. All learning involves remembering. This package is designed to teach you techniques and tricks to improve your ability to memorize information, so that learning details later will be easier. The catch-22 is that you need to memorize these techniques in order to make them work for you! But the lessons in this package are designed to make this process about as simple and painless as it could be.

There are lessons here for learning to memorize names and associate them with faces (with graphic faces for practice) long lists (such as shopping lists) numbers, important dates and phone numbers. There's nothing new about the techniques taught here. One—the method of loci—was used by ancient Greek orators to remember their speeches. What is new is the packaging of this information, in a set of computerized lessons and drills. This package is professional and thorough in its approach. It comes with an excellent book of supplementary information, and it even allows for creating customized lessons.

● **EARLY GAMES MUSIC**, from Counterpoint Software. This little package of games is designed to teach basic music skills and theory to children between ages 4 and 12, but the younger ones will probably need some guidance. The rotating picture menu lets the learner choose from simple drills that associate letter names to the notes on the piano keyboard, the treble clef, or the bass clef, with difficulty levels increasing as correct answers are given. Other options allow simple melodies to be played using the number keys, recorded (temporarily in memory or permanently on disk) and then played back later. The Melody Tutor teaches simple tunes using a “Simon Says” feedback approach. Kaleidoscore displays colorful graphics on the screen to accompany the melodies played on the keyboard. (There's more on music at the end of this chapter and in Chapter 9.)

● **SPACE RESCUE**, from the Benji Discovery Series of Human Engineered Software (Hesware). Commodore has a cartridge on the market for the C-64 called the **VISIBLE SOLAR SYSTEM** that is designed to familiarize students with our five closest planetary neighbors. A noble effort from Commodore, but I suspect more kids will learn the solar system with this educational arcade game designed for anybody over 8. The Benji logo might not appeal to everybody, but the game itself is well-designed to keep the interest of even hardened vidiots and teach them a thing or two along the way.

When you load this program, your screen becomes a combination control panel/viewscreen for spaceship Star Woof, and you become Benji, whose mission is to rescue scientists held

captive on one of the nine planets. That's no simple task: it takes some thumbing through the 15-page manual, and some practice to learn how to make your ship do what it needs to do. First you need to find the hostages, and that involves scanning maps of the nine planets (The graphic representations of the planets are beautiful, and the accompanying data are based on the latest NASA information.) Next, you have to plan a trip through the solar system, making sure you allow adequate fuel and rations for the journey. When you reach the target planet (if you don't hit an asteroid on the way), you have to carefully maneuver your ship to the surface and beam the hostages aboard while defending your ship from enemy attackers.

There are seven levels to this game, each one demanding more skills from the player: memory, planning, and coordination are all important. Graphics and sound are outstanding, and the game is fun to play, with much less emphasis on firepower than most space-style arcade games. An entertaining way for Trekkies to learn about our corner of the galaxy!

- **WALL STREET**, a competitive game of financial speculation from Timeworks Personal Computer Software. In the tradition of Monopoly, this game allows one to four players to speculate on the stocks, real estate, mineral exploration, precious metals, the money market, special high-risk investments, and an IRA. Each player starts with one million dollars, and the computer acts as broker and manages all accounts. An entertaining way to develop business savvy and learn how the rich get richer.

- **GORTEK AND THE MICROCHIPS**, from Commodore. If there's anything that microcomputers ought to be able to teach us, it's how to program microcomputers. And there are lots of packages on the market for the C-64 that claim to do just that. Unfortunately, most of these "tutorials" don't do much more than display poorly-written text on the screen that you could have just as easily (and for less money) read a book. GORTEK and THE MICROCHIPS is an exception, using arcade-style games to gradually introduce computer concepts to kids from keyboard familiarity to simple programming. This reasonably-priced package, originally written for the VIC, has a few errors and omissions in the manual. Also, the converted programs don't do

justice to the graphic capabilities of the C-64. But kids who aren't too hardened by arcades will enjoy the games and probably learn to program along the way.

Freebies (almost). Seekers of educational software shouldn't overlook the wealth of public domain material (Chapter 5) that Commodore is selling for little more than the cost of their packaging. Each disk has about a dozen programs on a particular subject: business, computer science, English, geography, history, mathematics, science, technology, and miscellaneous games. A typical English diskette includes word games, parts of speech, poetry, and definitions. The quality of the programs is, of course, mixed. But odds are you'll like a couple in each package. And if you don't, you can always reformat the disk and fill it with something else.

There are other public domain programs that haven't yet been marketed this way, including aids for learning music theory, sign language, Swedish, Latin, and French (especially French, because of the popularity of Commodore computers in Canada). Your dealer or user group might have these programs, or you can contact Torpet (see Appendix A.)

THE TURTLE AS TEACHER: A LOOK AT LOGO

Most of the educational software discussed so far is designed to teach one or two skills using some form of simple drill and practice. These programs are included here because they do a good job of teaching what they set out to teach, whether it's arithmetic, typing, or something else. But one educational theorist and computer scientist, Seymour Papert, says programs like these make the computer a tool for programming children, which is exactly the opposite of the way it should be!

Papert's educational philosophy is grounded firmly in the work of renowned Swiss developmental psychologist, Jean Piaget. According to Piaget, children have a natural gift for learning on their own—they learn to talk, get around, and think without any formal training. Papert says kids are natural builders of

ideas and concepts, and that the raw materials for those ideas come from their surroundings. So a child growing up in France learns French effortlessly, because his environment has the necessary materials. In Papert's vision, the computer can provide building blocks for powerful ideas in even the youngest children. He sees a world where children can learn to communicate easily and naturally with computers, so that learning mathematics, science, and even the arts can be as effortless as learning French in France.

With his colleagues at MIT, Papert developed a computer language called LOGO specifically as a tool for developing children's minds. LOGO is based on LISP, a LIST-processing language used in artificial intelligence research. 🐾 Logo has powerful features not found in BASIC, yet it's easier to use; children can write impressive programs in it as soon as they're old enough to read and write a few simple words—sometimes even sooner!

In one sense, LOGO is not simpler than BASIC to use. The Commodore 64 can read and run BASIC programs as soon as you turn it on because it contains a built-in continuously-running program for translating BASIC commands and statements into the machine's native language. There's no such built-in program for LOGO, so you have to load one from disk and run it to make your computer LOGO-literate and ready to teach.

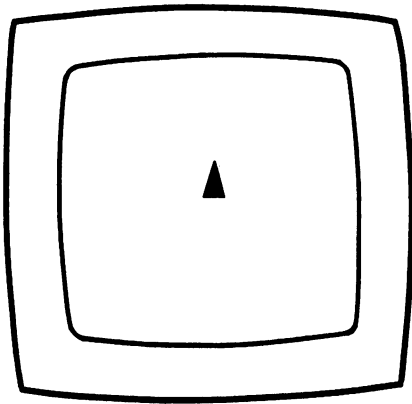
Rather than teaching a child by lessons and tests, LOGO creates environments for natural learning. The most famous of these LOGO environments is the world of TURTLE GRAPHICS. The first turtles were small mechanical robot-turtles that moved around on the floor, leaving a trail with a pen. A child could make one of these turtles go with Logo commands like these:

🐾 Artificial intelligence is, in a nutshell, the field that attempts to make computers do things that would be considered intelligent if people did them.

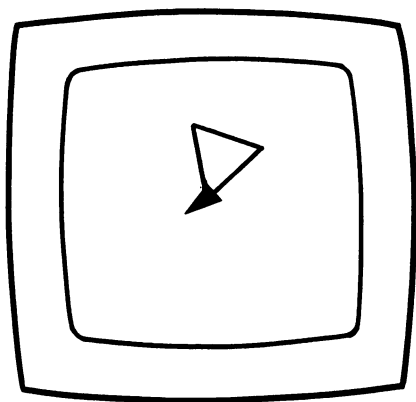
```
FORWARD 50  
RIGHT 120  
FORWARD 50  
RIGHT 120  
FORWARD 50
```

This sequence of commands tells the turtle to move forward 50 tiny steps, turn right 120 degrees (1/3 of a circle), move forward another 50 steps, turn right another 120, and move forward another 50. Imagine the child's excitement as he watched the robot obey his commands and scoot around the floor in a triangle-shaped path! But that child learned some high-powered geometry to make that happen: angle, degrees, triangle, equilateral—how old were you when you mastered those ideas?

While it's still possible to hook a mechanical turtle to a computer, today's LOGO turtle is typically a little triangle-shaped creature that lives in the middle of a TV screen:



Those same five LOGO commands that caused the mechanical turtle to triangulate the floor would cause a screen-turtle to trace this pattern:



Notice how the turtle is facing a different direction than it was when it started, because we didn't tell it to turn again after the last move. (The obedient terrapin doesn't do anything without being told.) Let's start fresh and draw the triangle again, this time with the final turn included. Starting fresh means clearing the screen and putting the turtle back in its home position:

```
CLEARSCREEN  
HOME
```

Drawing the triangle will be easier this time, because we'll take advantage of the fact that we're repeating the same set of actions 3 times:

```
REPEAT 3 [FORWARD 50 RIGHT 120]
```

REPEAT, FORWARD, and RIGHT are all commands that the LOGO turtle knows from birth. Talking to the turtle would get tiresome if every interaction had to be reduced to primitive commands like these. Fortunately, the little critter is eager to learn new words, if we're willing to teach it. To illustrate, let's teach it to draw a triangle like this one whenever we say TRIANGLE. To do that, we just need to type

```
TO TRIANGLE  
REPEAT 3 [FORWARD 50 RIGHT 120]  
END
```

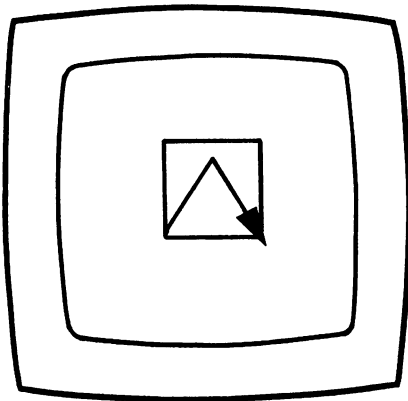
This is called defining a **PROCEDURE**, but it's nothing less than adding a new command to the turtle's vocabulary. Now that we've defined this procedure, the turtle will know exactly what to do when we say **TRIANGLE**. Let's teach it how to draw a square:

```
TO SQUARE  
REPEAT 4 [FORWARD 50 RIGHT 90]  
END
```

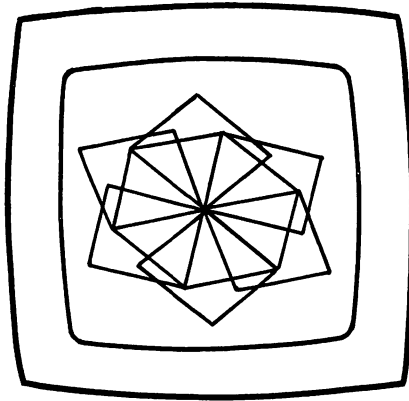
The turtle doesn't do anything when we type this definition; it just learns the word. To make the square happen, we have to use that word, alone or with other commands:

```
SQUARE  
TRIANGLE
```

Here's what the turtle draws for us when we type these commands:



One child might start with this and make it into an abstract design:

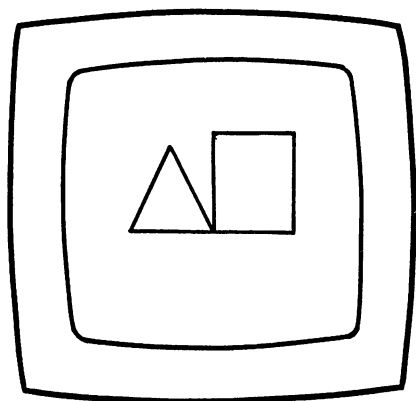


```
TO PINWHEEL  
REPEAT 6 [SQUARE TRIANGLE RIGHT 60]  
END
```

Another kid might look at it and think, "I could make a house with those pieces." And after some thought, trial-and-error, or turtle-role-playing, he might write a procedure like this:

```
TO HOUSE  
SQUARE  
FORWARD 50  
TRIANGLE  
END
```

The turtle will draw this when the child tells it to HOUSE:



Still not exactly right, but that's OK. In LOGOland, kids learn quickly that mistakes are part of the process of learning, and that progress is made by studying mistakes rather than trying to forget them. By looking at this latest picture and thinking about the procedures that drew it, the child will eventually see a way to put his house in order. These kinds of lessons—learning about learning and thinking about thinking—are at least as important as the geometry lessons the LOGO turtle teaches. LOGO provides kids with all kinds of problems that they can relate to, understand, work with, and solve.

LOGO isn't just a toy for little kids, though. It's one of the most powerful languages available for microcomputers today. In addition to the standard language features—variables, PRINT statements, IF statements, loops, and branches, LOGO has features that make BASIC look anemic: procedures for extending the vocabulary of the language; parameters for passing data into and out of procedures; and recursion, which lets a procedure call itself! Recursion is especially interesting, because it illustrates the power of LOGO to make abstract ideas concrete. Many a computer science graduate student has wrestled for days with the concept of recursion without gaining a real understanding of it. But grade school kids working with LOGO can easily grasp a procedure like this one:

```
TO CIRCLE  
FORWARD 1 RIGHT 1  
CIRCLE  
END
```

In short, this says “to draw a circle, go one step forward, turn 1 degree to the right, and repeat ALL of these instructions.” (If you’re confused, pretend you’re the turtle and try it!)

Unlike many so-called educational programs, LOGO isn’t a one-trick pony that gets boring in a day or two. Your Commodore-64 LOGO is especially rich in features for maintaining high levels of interest and learning. When you and the kids have had enough of black-and-white line drawings, you can experiment with different color schemes. Or you can use LOGO’s sprite commands to develop animated scenes or joystick-controlled video games. LOGO also has commands for playing melodies or making sound effects. Or experiment with the computational and list-processing capabilities of the language that can make child’s play out of complicated word, number, and logic programs.

If you’re concerned that all of this sounds too technical and complicated for the family to handle, relax. LOGO is designed to be easy to learn, and the Commodore version (designed by Terrapin, Inc.) follows in this tradition. The Commodore language extensions for working with Sprites and sounds are much simpler and more fun than the BASIC poking you’d have to do to create the same effects. These language extensions are stored on a LOGO Utility Disk that comes with the language disk. The utility disk is packed with LOGO procedures to simplify the LOGO programming process and demonstrate language features and programming techniques. Many of these programs are fun in their own right: an adventure game, an animal guessing game that illustrates computer-learning, and several fancy graphic demos. There’s even a simplified LOGO that allows non-readers to talk to the turtle with single-letter commands. Finally, the giant book that comes with these two disks is as friendly and complete as the language itself. It’s easy to

use as a self-teaching guide, a reference manual, and a source-book for programming exercises and examples.

But beware: LOGO is highly addictive. If you're a BASIC programmer, a couple of weeks programming in LOGO may convert you forever. If you're not a programmer, try LOGO. It's a perfect place to start. LOGO is for learning!

MUSIC AND ART: CREATIVITY SOFTWARE

This chapter ends with a look at a few programs that are definitely fun, but not strictly games; they're certainly educational, even though they probably weren't designed to teach; they could even be called tools, but you're not likely to see them on the shelf with word processors and spreadsheets. For lack of a better word, these programs are toys, because they're so much fun to play with. Even the most serious adult will probably find something in this collection to wake up the kid inside, get the creative juices flowing, and cause a smile or a giggle. Betcha.

If your interest is peaked by these programs, be sure to read the "What Next?" section of Chapter 8 — there are more music and art programs there.

- **MUSIC MACHINE**, from Commodore. Just plug this cartridge into the back of your computer, and you've got a musical instrument! Each button in the QWERTY row of the keyboard is now a white key of a piano keyboard, and the number keys are the blacks, ready to sing for you. Each time you play a note, it's displayed on a musical staff on the screen. Touching the cursor keys moves the pitch up or down an octave or six, and two other keys can fine-tune the pitch to match the other instruments (or recordings) in your house.

There's more. You have a choice of four waveforms (sawtooth, triangular, square, and narrow pulse) that represent different sound qualities (e.g., brassy). You can choose between letting the notes decay (like they do on a piano), sustain (like an organ), or hold (even after you release the key). You can give notes a

vibrato quality, or make them slide into each other, or add other special effects. And you can add a second or third voice to produce rich harmonies. You'll be amazed at the different sounds you can get by combining these effects in different ways!

Even if you have no musical training, this cartridge is a lot of fun (and in its own subversive way, it is musical training). The sound is great, especially when piped through a good music system. But if you're looking for a professional-quality music synthesizer, keep looking. The C-64 keyboard just isn't designed to play music; fingering is awkward, and it's impossible to produce solid three-note chords — you have to stagger the note attacks a little. The “rhythm accompaniment,” little more than the sound of two very bored mechanical hands clapping, is best when it's turned off altogether. Even so, you're likely to have arguments over who gets to play with this one next.

- **MUSIC COMPOSER**, also from Commodore. If you expect a tool for professional musical composition, you're going to be disappointed with this cartridge. This is a tool/toy for building songs and sonotas using a simple notation for specifying voices, notes, octaves, time signatures, attack/decay, filtering, and special effects. You can play back your masterpiece, and change it until you're happy with it, and then store it on tape (alas, no disk option?) for future generations. It takes a while to get used to the notation, which looks a little bit like machine language program (a simple song in the not-very-clear instruction book begins “0005(1V104CEFWGV2QR—EV3”). But if you persevere, you'll be able to make some fancy sounding tapes, and learn a little about music and programming along the way. (There's also an option in this program for playing music with the C-64 keyboard, but it doesn't have the features of the MUSIC MAKER cartridge. Unfortunately, it doesn't even use the same part of the keyboard.)

- **DANCING FEATS**, from Softsync, Inc. Dancing Feats is a different kind of musical toy that turns your joystick into a musical instrument and your computer into a back-up band. You can tell the computer what kind of bass, beat, style, tempo, and ending you want by responding to simple multiple-choice menus. There are lots of musical options, ranging in style from jazz to rock. But whatever you choose, it's up to you to provide

the melody by moving your joystick around. You can't control the attack, decay, sustain, release, or waveform of your instrument—just the notes. If even that scares you, relax. This program has been designed so that even if you know nothing about music, you'll be able to improvise solos without fear of hitting sour notes. As a visual bonus, each note you play splashes onto the screen as a multi-colored bar labeled with the note name. The program has options for turning off the back-up band; recording and playing back your compositions, and saving and recalling them with disk or tape.

● **MUSICOMP**, from Computer Alliance. If you want a more serious musical tool, consider Musicomp. This is really a musical language editor that lets you type BASIC-like statements into your computer to create music. To give you an idea of the power of this package, Musicomp comes with a "jukebox" full of music created with the Musicomp editor. While the jukebox is playing a tune, the musical notation and lyrics (if any) are displayed on the screen so you can sing along with your C-64. If you understand basic musical concepts like "dotted quarter note" and "key of F," you should have no trouble learning how to type songs (from sheet music or your head) into the Musicomp editor. (In case you don't, the 50-page manual has a chapter explaining musical notation.) The Musicomp language is easy to learn, and the editor works like the one that's built into Commodore BASIC. Once your song is typed in, you can play it back, list it, modify it, save it on disk, add it to the jukebox, or even build it into your own homemade programs!

There's a lot of flexibility in this program, with 16 possible instrumental sounds and an eight octave range for each of the three voices. The last part of the manual has a tremendous amount of technical information for musical hackers who want even more freedom to create. And as a bonus, the disk includes a much-improved version of the piano program that's listed in the back of the book that came with your computer. A real musical feast.

● **DOODLE**, from Omni Unlimited Computer Marketing Services. This feature-packed program is designed for drawing high-resolution pictures using a trackball (a control device that operates on the same principle as a roll-on deodorant bottle), but

it works well with a joystick, too. When you start the program, there's a cross hair marking a spot in the middle of the white screen. Moving the joystick or trackball moves the cross hair; the fire button turns the pen on and off, determining whether a thin black line is left behind by the moving spot. The number keys control the thickness of the spot, and you can fill in any white area by pressing P (for paint). At the touch of a key, you can reverse your picture left-to-right or black-to-white. If you forget how to do any of this, RETURN returns you to a simple picture menu, and another RETURN takes you back to your work of art. And you can memorize a spare copy of your picture-in-progress as insurance against a slip of the joystick. Most drawing programs stop here, but this is just a partial description of one of DOODLES nine modes!

Erase mode is the same as Draw mode, except that the spot erases instead of drawing. In Line mode, you move two spots, and the fire button makes a straight line between them. Box mode allows you to create boxes of any size and shape, and make transformations on whatever's inside them. Copy mode lets you make copies of boxes in other parts of your picture. Zoom gives you a microscopic view of part of your picture; you can move the scope around the whole picture, making changes to individual pixels along the way. Letter lets you put text on your creation, sideways or upside-down if you like. Disk and Print lets you save your masterpiece on disk or print it out on a graphic printer. There's even a color mode, which allows you to add color to a black-and-white picture.

The documentation that comes with this program is readable and complete; it even tells you how to add doodles to your own BASIC programs. But most of the features of the program are simple enough to use that you probably won't look at the instructions much; you'll be too busy playing. For kids and artists of all ages.

● **PAINT MAGIC**, from DATAMOST. DOODLE is great for drawing in black-and-white, but it's not the best for color work. If you're interested in drawing hi-res color pictures on your screen without investing in extra hardware (besides a joystick), try PAINT MAGIC, a disk that lives up to its name. PAINT MAGIC is roughly for color what DOODLE is for black-and-white. With PAINT MAGIC you can draw color sketches freehand; add

computer-generated circles, lines, boxes easily; fill parts of the picture with patches of solid color, stripes, or textured patterns; move things around on the screen; zoom in for close-up work; and save pictures on disk or tape. (You can't print pictures on your printer, but it's not hard to photograph the screen.) You'll be amazed at what you can do with this program!

- **64 BANNER**, from Midwest Micro Associates. This simple program lets you use your computer and printer to create signs, posters, or banners, by turning the itty-bitty letters you type into headline-size print across the printer or giant text that's printed out sideways. It's a fun toy for kids, but it's also a useful tool for those of us who don't have a natural talent for making HAPPY BIRTHDAY or GARAGE SALE signs.

(There's more on music and art in the last part of Chapter 8.)

CHAPTER 7

THE MARVELOUS TOOL: WORKING SOFTWARE

If every instrument could do its own work, if the shuttle should weave of itself and the plectrum play the harp unaided, then managers would not need workers and masters would not need slaves.

—Aristotle, *The Politics*

Most home computer owners never get much beyond fun and games, because most home computers aren't designed to get serious. The Commodore 64 is the first low-cost computer with the potential to be a powerful workhorse, too. And you don't need a closet full of software to make it work for you, either.

There are hundreds of programs being sold as household helpers: programs to tell you how much to pay for a mortgage, how much insulation to put in the attic, how much paint you'll need to turn the kitchen yellow, and how many liters of milk to put in the soup. It's an appealing idea to have the computer answer these kinds of household questions at the touch of a button. But what the advertisements for these programs don't tell you is that in the time it takes you turn the computer on, locate and load the program, and type in the necessary inputs, you might have been able to do the job easier with a calculator. And even if the computer program is faster than your calculator, will you use that program often enough to justify owning it in the first place?

The answer of course, depends on the program, the situation, and you. What looks like a waste of time to me might feel like a time saver to you. Since everybody's needs are unique, it's difficult to make general statements about the utility of most specialized programs. General purpose programs are another matter; these programs are versatile enough that they can be tailored to fit the needs of many a household.

Most of the “serious” jobs that microcomputers do fall into one of three categories: word processing, numeric calculation, and data storage and retrieval. So a home software library that contains a word processing program, a calc-type spreadsheet program, and a database management program is well-equipped to handle most situations. This chapter will discuss each of these kinds of programs, along with a representative handful of special-purpose programs.

TYPING WITHOUT TEARS: WORD PROCESSING FOR BEGINNERS

Between television and telephones, the average person doesn't write much anymore. Writing is a slow, inefficient method of communicating: correcting spelling and errors retyping whitening out—who needs it? But if writing is a dying art, word processing may revive it. A good word processor allows even a hunt-and-pecker to easily produce copy that looks like it was generated, not by a professional typist, but by a professional typesetter.🐾 But more importantly, a word processor strips away most of the drudgery from the writing process, in the same way that a food processor eliminates much of the tedium from making a vegetable stew. Bothersome details like margins, spacing, layout, page numbering, centering, indentation, and underlining are all taken care of automatically by a good word processor.

What about the bothersome details of spelling? Most word processing programs don't check your spelling for you. But many of the good ones are compatible with proofreading programs that will. A spelling checker can, in a matter of minutes (as programs go, that's slow!), take the text generated on a word processor and look up every word in a dictionary-on-a-disk, pointing out each one that it can't find so that you can decide whether it's an error, or just an esoteric word or name that's not in the dictionary. You then have the choice of changing the spelling in the text (if it was

🐾 In the business world, the term “word processor” is often used to refer to a WORD PROCESSING STATION, which is basically a DEDICATED microcomputer system that can ONLY be used for word processing. When applied to personal computers, the term refers to a program that makes the computer perform some or all of the functions of those expensive dedicated systems.

an error), adding the name to the dictionary (if it's a correctly-spelled word that you're likely to use again), or (if it's a word you don't often use) doing nothing. Of course, a spelling checker is no substitute for human proofreading; it won't tell you that you typed "food" when you meant "good," or that your correctly spelled words don't make any sense together. But if spelling is your bugaboo, it's a big help.

The best thing about electronic writing aids like word processors and spelling checkers is that they free the writer to concentrate on the words and the ideas—the fun stuff—rather than on the technicalities of the medium. Writing with a word processor is not only easier, faster, and more accurate—it's a whole new form of creative expression!

There are dozens of things you can do with a word processor, including some you might not even consider doing now because they're too much work the old way: writing a letter to the editor, your senator, or grandma; a term paper or take-home exam, an article for the church newsletter or *Creative Computing*; your memoirs or the Great American How-To Book. And because your ideas can flow fast and freely without being dragged down by erasures or carriage returns, a word processor makes an ideal tool for keeping a diary or a therapeutic journal. (Nobody's likely to read your diskette because it accidentally fell open while they were cleaning, either.)

The best way to learn about word processors is to use them. If that's not practical for you, read this next section and pretend. I'll walk you step-by-step through the process of creating a letter using a simple, entry-level word processor called the WORD MACHINE. You'll get a feel for how that program works, but you'll also learn principles and terms that apply to all word processors. Then for contrast, I'll show you a short example using EASY SCRIPT, a fancier word processor from Commodore. With two vantage points for perspective, you'll be able to see the alternative word processors at the end of the section clearly.

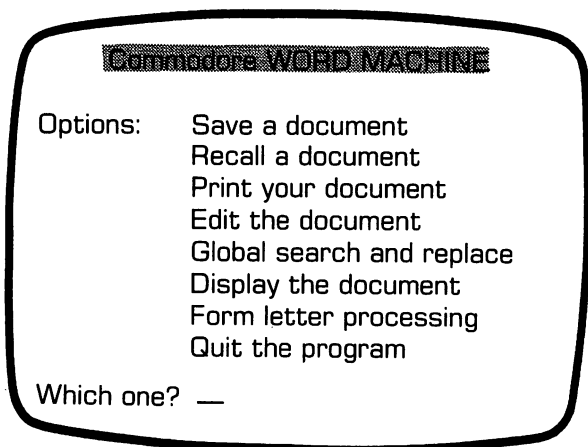
THE WORD MACHINE. If you're not ready to jump into the deep end of the typing pool, there's a low-cost way to top test the water. **THE WORD MACHINE** is Commodore's "no frills" word processor, designed to be easy to afford and, above all else, easy to use. Since it has all of the essentials and very few distracting extras, the **WORD MACHINE** is a good word processor to use to get a feel for the basics.

When we load the **WORD MACHINE** into our computer and type **RUN**, we're given a chance to choose the screen's background, border, and character colors. This may seem inconsequential, but it's an important feature for a word processor to have. You've got to be able to read the characters on the screen easily, and some color combinations just don't work on some screens.

Next, we're given a choice of using disk, tape, or both for storing your documents. Options are nice, but word processing with a datasette is a questionable choice (Chapter 1).

The **WORD MACHINE** then asks for the date. It prints the date at the top of formal letters, and stores it with any documents that we save during this session, so that we have a way of knowing later when the document was last saved. This handy feature is unusual for a low-cost word processor.

Finally, the screen shows this **MENU**, and the program waits for us to type the letter indicating our choice:



Before we can do anything with a document, we have to create it, so we'll start by typing C. The WORD MACHINE responds by asking what we want to call the document, and we respond by typing in pancake. The screen clears, leaving only pancake and the cursor, and we're ready to start typing. With the WORD MACHINE, and with word processors in general, we don't need to type carriage returns at the end of each line. The text just wraps around for now, to be sliced up into lines later, when we print the document. The only time we need to insert a carriage return is when we want to FORCE a new line, such as at the end of a paragraph. (It's a minor inconvenience that the WORD MACHINE recognizes the backarrow (←) key, rather than the RETURN key, as a carriage return.) Unlike more expensive word processors, the WORD MACHINE doesn't let us set tab stops or change margins in the middle of a document, so if we want to have part of the text indented, we have to include extra blank spaces in the text. Here's our screen after we type a short note with the closing moved to the right by padding it with blanks:

Pancake

Dear Editer, I'm writing this letter to tell your readers about the Sheep Lodge Recycled Pancake Breakfast at the Armor y this saturday. we'll have something f or everbody: the little-leaguer dunk-an -alderman booth, a performance by the re gional champion teakettle band, and an i nspiring speeches by our own Senator Pay off and Editer Bullroar. See y'all there e!—

Sincerelyp,---

Bodak 3E. Rumad

You may find this letter a little difficult to read because some words (like "Armory") are split between two lines. When we print the finished document, those broken words will be glued back together again. But for now, you'll have to read them in pieces because the WORD MACHINE doesn't have the word wrap feature found in more expensive word processors. If it did, words that didn't fit on a screen line would be automatically moved to the next line.

Editing the errors out of this letter isn't quite as easy as it would be if we had used a fancier word processor. We can use the DEL key to backspace over the unwanted "3" in the middle name, and then retype the rest of the line. But it's not possible to just cursor-up to the other errors on the screen and fix them without going into EDIT mode via the main menu. Punching the (£) key returns us to the menu, and E takes us to the Editor.

Because of the low-budget way the WORD MACHINE was programmed, we can only edit one "block" (about four lines) of the text at once. But within this block, we're free to move the cursor around delete characters, and insert text using the same keys that we use for editing in BASIC. (The keys are the same, but unfortunately, they don't work quite the same.) When we're through editing a block, the next block is just a carriage return away. (And the previous block is just a shifted return away.) As before, the pound sign can return us to the main menu.

The repeated misspelling of "editor" could be repaired in edit mode, but the WORD MACHINE has an option called global search and replace that makes the job easier. All we need to do is type G from the menu and answer a couple of quick questions. Search for? Editer. Replace With? Editor. The program responds by making the substitution every time it finds the target word in the text. (This is handy when you write a long, personal letter to Kathy and then discover that her name is Cathy.)

To look at the complete, edited letter, we can type D to Display it on the screen. (Of course, it won't look exactly the same on the screen as it will when we print it, because the screen is only 40 columns wide.) When we're ready to print, we type P and specify that we want to print an informal draft. The WORD

MACHINE then asks us about margin width, spacing, and other details. If we refuse to answer, it chooses standard **DEFAULT** settings for us, and prints this:

Pancake

Dear Editor,

I'm writing this letter to tell your readers about the Sheep Lodge

Recycled Pancake Breakfast at the Armory this Saturday. We'll have

something for everybody: the little-leaguer dunk-an-alderman booth, a

performance by the regional champion teakettle band, and inspiring

speeches by our own Senator Payoff and Editor Bullroar.

See y'all there!

Sincerely,

Bodak E. Rumad

Once the letter is printed, the only remaining job is to save it as a disk or tape file (S from the menu) so that we can recall, refer to, rework, and reuse it later.

The **WORD MACHINE** can also simplify writing a formal letter, complete with personalized letterhead and inside address. The information for the letterhead must be stored in three data statements in the program; the documentation explains how to modify those statements. The information for the inside address may be typed in at the time the letter is printed, or it may be stored in a data file created by the companion program, the **NAME MACHINE**.

The **NAME MACHINE** allows the user to easily create a data file containing names and addresses of friends, colleagues, clients, or whomever. this program is discussed in the section

on data bases; the important point here is that a Name Machine file of addresses can be used by the WORD MACHINE to produce individually-addressed form letters. In the jargon, the WORD MACHINE has MAIL-MERGE capabilities, so you can build a junk-mail empire with individually-addressed letters like this one:

Exploitations Unlimited
P.O. Box 765
Ripoff, MA 43210
October 31, 1984

Mr. Mike Roprocessor, CPU
Commodore Computer
6510 Chip Av.
Commodore, CO 00064

Dear Mike,

Hooray!! They're finally here! And you'll be the envy of all your neighbors when you have your very own! Just think of what they'll be saying when they learn that your bathroom is accented with Fashion Designer Endangered Species Toilet Seat Covers!

Oh so sincerely yours,

Anna Maul, President
Exploitations Unlimited

PS! They make great gifts!

Mailmerge and global search-and-replace are unusual features to find in an inexpensive word processor. All things considered, the WORD MACHINE is a bargain. With its low price, easy-to-read documentation, and easy-to-understand commands, it's a great word processor for the curious beginner. But it's a far cry from a professional word processing system. If you do much writing at all, you'll be frustrated by two limitations of the WORD MACHINE: speed and flexibility.

- **Speed.** Because it's written in BASIC, the WORD MACHINE is easy to modify and back up. But it's also very slow, especially when editing and printing. It takes forever to print a long document. And the lack of true screen editing also contributes to the slowness of the program.

- **Flexibility.** The WORD MACHINE is designed to make letter writing easy, and it does a good job of that. But it doesn't do quite as well with term papers, magazine articles, books, newsletters, and such. It lacks page-numbering, titles, headers, centering, columnar tab stops, document-chaining, and other advanced features that make a word processor well-suited for those other jobs. Which is not to say that you can't use it for them; I wrote the introduction to this book with the WORD MACHINE. But an introduction was enough!

EASY SCRIPT. I'm now using EASY SCRIPT, Commodore's "comprehensive" word processor for the C-64. The WORD MACHINE and EASY SCRIPT have very little in common except their brand, so a look at EASY SCRIPT will give you a good idea of the choices involved in selecting a word processor.

In the first place, EASY SCRIPT is written completely in machine language, so it's very fast. Loading a text file from disk takes seconds instead of minutes. During editing, the cursor doesn't creep around the screen—it flies! and when printing, the word processor has to wait for the printer, instead of the other way around.

Secondly, EASY SCRIPT is packed with features and options. It's an extremely flexible program, designed to process business words, scientific words, and literary words with ease. Of course, this kind of flexibility doesn't come free; you give up simplicity to get it. Like a car or a stereo with lots of options, EASY SCRIPT takes a while to get to know. The documentation explaining how to use all of those features is nearly 200 pages long! (The first half is an excellent tutorial that leads you by the hand through the things you have to know; the other half is a reference guide, presenting all of the features in an easy-to-look-up form.) It takes a few hours to learn the ins and outs of EASY SCRIPT, but that time investment pays for itself quickly if you do much writing.

Another difference: in EASY SCRIPT, there's no distinction between input mode (when you type a document in), and edit mode (when you move the cursor around to make changes). When you type text into EASY SCRIPT, the screen fills from

top to bottom. When it's full, the top line scrolls off the screen as you add to the bottom. That's the way the WORD MACHINE works, too. But with EASY SCRIPT, you can interrupt your typing and cursor-up to another part of the text, even if that part has scrolled off the screen. (Cursoring up scrolls it back on.) So you're free to make changes, additions, or deletions anywhere in the document when they occur to you, even if you're in the middle of entering text. (And thankfully, the INST/DEL key works the way it does in BASIC.)

Finally, the way you communicate information to EASY SCRIPT is different from the menu-and-question style of the WORD MACHINE. Except for three quick questions when you first turn it on, EASY SCRIPT doesn't do much prompting. Instead, it gets the information it needs for you by way of control commands and embedded format instructions.

Control commands are used to tell EASY SCRIPT to do things like text saving, printing, and global search and replace. You indicate a control command by first pressing the f1 button (top right of the keyboard) and then pressing another key: F to save a file, L to load, D to delete a block of code, O for output to put you in the insert mode (so that you can type new text in the middle of the document and everything will automatically move over and down to make room for it), and so on. When the f1 key is used with the cursor keys, it gives them special meanings too: f1 followed by DEL deletes a line rather than a character, and f1-INST inserts a line. A few commands have their own special keys and don't need to be preceded by f1, example, f5. For shifts to CAPS mode, where all alphabetic text is typed in capital letters. Any of the three screen colors can be changed anytime by simultaneously pressing CTRL and either 1, 2, or 3.

Embedded format commands are used to tell EASY SCRIPT how to lay the document out on the page: where to put the margins, where to put the page headings and page numbers, how many spaces to put between lines, whether to put extra spaces between words to make the right margin smooth (right justification), and the like. Commands can go anywhere in the text. For example, you could set the left margin to column 10 at the beginning of the text, later switch to column 15 to indent a quotation, and then switch back to column 10 for the rest of the document. These

embedded commands are distinguished from the rest of the text by an inverse asterisk (*), which is made in EASY SCRIPT by pressing the f3 key. Here's a familiar example:

```
EDIT MODE                                L024 6:00
*nb:"Pancake"
*lm12:rm72
Dear Editer,
I'm writing this letter to tell your readers about the Sheep Lodge Recycled Pancake Breakfast at the Armory this Saturday. We'll have something for everybody: the little-leaguer dunk-an-alderman boot h, a performance by the regional champion teakettle band, and an inspiring speeches by our own Senator Payoff and Editer Bullroar. See y'all there!
*lm52
Sincerelyp,
Bodak E. Rumad
```

The top line of the screen is the "status line"; it's the only place on the screen that you can't enter text. It tells you what EASY SCRIPT is doing at any given time, what it expects from you, and where in the document the cursor is (line and column). The next line contains our first embedded format command. This particular one, "nb," is a comment, like a BASIC REM. The inverse arrow at the end of the line is the symbol that represents a carriage return; it's made, conveniently enough, by hitting the RETURN key. The next line sets the left and right margins at columns 12 and 72, respectively. Then, the text. Notice that the left margin is reset to 52 before the closing so we don't need to include all those extra spaces. Now that

the text is typed, we can zip around the screen with the cursor keys making changes and additions as we go. (The manual doesn't tell you, but you can also use a joystick to move the cursor around.)

When we're done editing, how do we know that our formatting commands will do what we think they'll do? Most professional word processors allow you to view the document on the screen exactly as it will look on paper. It's hard to do that with the Commodore 64's 40-column screen, but EASY SCRIPT gets around that limitation by allowing you to look at the document through a 40 by 25 character "window" that you can move around the page with the cursor keys. It's not the best way to read your document, but it's great for checking the layout of the text on the page before you commit it to paper.

This simple example doesn't begin to convey EASY SCRIPT's many features. Take, the disk mode. If you punch f4 while you're in EASY SCRIPT, you are magically transported to disk mode, where you can do all of the important disk operations (look at the directory, scratch files, etc.) without using all of those ugly OPEN and PRINT statements. And without losing the document you're working on! Very handy.

Easy Script is especially nice for long documents, like term papers, articles, and books. It allows you to easily move, delete, save, and duplicate large blocks of text. Each page of a document can have a header and/or footer containing title, author, page number, etc. EASY SCRIPT can hold extremely large documents in the computer's memory (this chapter up to this point, for example). But if your document won't all fit, you can CHAIN two or more text files together with embedded commands so that they will print one after another, with no break in the output or page numbering. If you need to make a minor change in one page of a document after it's been printed, you can print only that page.

There are lots of features that make EASY SCRIPT attractive for business-type uses, too: BOILERPLATING (reusing standard blocks of text and/or commands in different documents—handy when you want to send similar letters to several people),

tab stops, NUMERIC MODE for lining up columns of figures by decimal point, printing of multiple copies, mail merge (using files created with EASY SCRIPT, but not with the NAME MACHINE or EASY MAIL), and bold-face printing and underlining (depending, or course, on the printer), to name a few. Commodore will soon release a compatible spelling checker called EASY SPELL to make writing easier for problem spellers.

While EASY SCRIPT is a good choice for prolific writers and busy typists, it may not be the best word processor for everybody. If you're a pro or a perfectionist, you'll probably want a word processor with word wrap and a true 80-column screen display. On the other hand, if you're planning on using your word processor just for an occasional letter, you might be a little bit intimidated by the size of EASY SCRIPT. Since it doesn't lead you by the hand with menus and questions, EASY SCRIPT requires some effort in the beginning, and regular use for a while, before it becomes REALLY easy.

Another factor that might make you think twice about EASY SCRIPT is printer compatibility. While EASY SCRIPT is designed to work with most popular printer models, it works easier with some than others. If you're working with a Commodore printer, wonderful—that's EASY SCRIPT's specialty. Interfacing with other major brands is discussed in the appendix of the manual. In general it's not a problem. But it's not as easy as it is with other word processors that support printer command files.

Other Word Processors. If neither of the two Commodore word processors seems right for you, that's OK. There are plenty of alternatives available, many of them truly first-rate. Here's a survey of some of the best values in word processors and spelling checkers for the 64.

● **WORDPRO 3+.** Commodore really should have given a footnote in their EASY SCRIPT documentation to the folks at Professional Software; it's obvious that they used WORDPRO as a model for that Word Processor. And with good reason—WORDPRO has long been considered by many to be THE standard word processor for Commodore computers. It's powerful,

fast, flexible, and extremely popular. WORDPRO 3+ has most of the same features as EASY SCRIPT. WORDPRO 3+ doesn't let you preview your output on the screen, but it allows you to switch back and forth between two texts in memory.

A choice between these two word processors is largely a matter of personal needs and personal taste. I lean toward EASY SCRIPT. I'm especially fond of its large workspace, output preview, and reasonable price. The documentation is a little easier to use, too; it's nicely organized and indexed. I also find commands in EASY SCRIPT easier to remember and use. That means I spend less time looking things up, and I make fewer mistakes. More than once when using WORDPRO to work on linked files I've lost an hour's work because I accidentally hit a wrong key and dragged a new file into memory to wipe out the one I'd been working on.

Of course, there's another point of view. Every word processor I've ever used—including EASY SCRIPT—has cost me work because of a bug or a design flaw. Give WORDPRO a point for customer support, too—it's much easier to get help from Professional Software than from Commodore. That's not important until something goes wrong or you have a question (such as "How do I make this feature work on my printer?"), and then it's REALLY important.

- **SPELLRIGHT PLUS.** Professional Software also offers this optional spelling-check program that's compatible with WORDPRO 3+. SPELLRIGHT PLUS is sold as a separate program or in a package with WORDPRO. The program has a dictionary of more than 16,400 words most used in business letters, and allows you to add about 1,500 of your own favorites. (I ran the section on diskette care in Chapter 4 through SPELLMASTER and I found it didn't recognize words like "exasperating," "deteriorate," "dust-free," and "clips.") Like any spelling checker, SPELLMASTER takes a while to run. But it's an excellent tool for detecting typos that might go unnoticed by a human proofreader.

- **OMNIWRITER and OMNISPELL,** from Hesware. This package contains a powerful but easy-to-use word processor

along with a spelling checker that has a 30,000-word expandable dictionary. The word processor lacks some of the formatting flexibility of EASY SCRIPT (I wasn't able to make it print double-spaced text, for example), and the accompanying manual looks like a pamphlet next to EASY SCRIPT's. (An encyclopedia isn't necessary, but an index would be nice.)

But don't write OMNIWRITER off without considering its strengths. Unlike EASY SCRIPT and WORDPRO, OMNIWRITER formats the text as it's being entered. That means that you can see anytime (via side-scrolling) exactly what your document is going to look like. Or, if constant side-scrolling makes you dizzy, you can switch to a screen-width format while you're typing. Even with this narrower format, OMNIWRITER has word wrap capability, so that words are never split between two lines, as they are in EASY SCRIPT.

OMNISPELL is an extremely fast and thorough spelling program for use with OMNIWRITER. The two programs aren't just compatible, they're integrated, so that you can easily switch back and forth between the two without issuing load commands. In addition to checking spelling, OMNISPELL displays on command a frequency count of words used in a document (ordered alphabetically or by frequency), so you can better tailor your vocabulary to your audience.

● **BANK STREET WRITER.** This product of Broderbund Software was originally developed by Intentional Educations, of Watertown, Massachusetts, and the Bank Street College of Education in New York as an educational writing tool for children. It's being marketed now as the "family" word processor, because it's so simple to use that even adults can understand it. While it doesn't have all of the professional features of EASY SCRIPT or WORDPRO, it has most of the important ones that the average person needs, plus some word wrap: global search and replace, block add, move, and delete, automatic centering and indenting, documentation chaining, and page headers, to name a few.

But the real strength of BSW is its friendliness. It's easy to learn: if you use the excellent tutorial that comes on the back

of the program disk, you'll be word processing in an hour. (Why doesn't every program come with a tutorial like this one.) It's simple to use: Menus guide you easily through the processes of editing and transferring text. And it's forgiving: it has an UNERASE command for that inevitable accidental wipeout.

All of this friendliness might begin to get in your way if you do much writing. While menus and prompts are helpful if you're an occasional typist who doesn't want to memorize command keys, they can become annoying once you've learned your way around the options. Prolific writers usually prefer the speed and convenience of keystroke commands to the helpfulness of menus. And you may not like the way BSW separates text entry and text editing into separate functions. But for many family needs, BANK STREET WRITER is a great compromise between simplicity and power; try it if you can.

● **TOTL2.6.** TOTL Text, from TOTL (for Tuna Of The Land!) Software, is an easy-to-use word processor with a very reasonable price. In some ways, it's similar to the WORD MACHINE. It's written mostly in BASIC, so it doesn't set any speed records. It divides text into blocks for editing, rather than treating it as a whole. It prompts you with a menu to simplify your decision-making. And it has a companion program (TOTL Label) that can be used for mail-merge. (A compatible spelling checker is in the works.)

But TOTL-Text is considerably more powerful and flexible than the WORD MACHINE. Its got features that you don't normally find in a low-cost word processor: block move, insert, and delete; tab stops multiple page header lines; page numbering; centering; chaining; and others. In fact, it's the only low-priced processor I've seen for the 64 that has automatic footnoting capability, making it a good choice for students. 🐾 One aggravating aspect of TOTL Text is the use of four-character embedded format commands for everything from tabs to carriage returns. (It helps a little that many of those commands can now be abbreviated with a single keystroke.)

🐾 Because, as everybody knows, term paper grades are directly proportional to the number of footnotes.

One thing I can't fault about TOTL Text, though, is the customer support. The TOTL people encourage you to contact them with questions, suggestions, etc. They make improvements in their programs based on feedback they get from users. They even publish an occasional newsletter clarifying features and pointing out bugs and how to fix them!

• **SCRIPT 64**, from Richvale Communications, distributed by Computer Marketing Services, Inc. SCRIPT 64 has a list of features that makes most other 64 word processors look incomplete. Two features stand out particularly. First, it has a built-in spelling checker. The initial dictionary has only a few words; you add to it each time you use the dictionary option by telling the program which words are spelled correctly. You can expand the dictionary in this way up to 20,000 words. Second, SCRIPT 64 operates in the normal 40-characters-per-line mode or in 80-column mode like the pros use! That means you can preview the output on the screen as it will look on paper without having to scroll back and forth across the page. (You'll probably need a monochrome monitor to read the tiny characters on the screen, though.)

In addition to these two important extras, SCRIPT 64 has all of the important features you'd expect from a professional-quality word processor. The 140-page ring-binder manual is both a clearly-written tutorial and a well-organized reference manual. There are a series of HELP SCREENS you can conjure up at the touch of two buttons to remind you which key does what.

Still, there's something about the STYLE of this word processor that makes it less appealing to me than some of the others. When you're using WORDPRO or EASY SCRIPT, the screen is a window through which you create, view, or change a stream of text. Within the confines of the memory available, you can scroll the window up or down through your document simply by moving the cursor. With SCRIPT 64, the screen is the fundamental unit of organization for text. Moving around in a large document involves jumping from screen to screen, and virtually every operation you perform, from moving blocks of text to printing your document, involves specifying screen numbers. Even when you're entering text, you have to con-

stantly be aware of how much space you have left on the screen, in the same way you have to listen for the bell when you're using an old-fashioned typewriter. There's nothing wrong with this scheme; I just find it inconvenient. Of course, this is a personal judgement. Long-time users of SCRIPT 64 probably perform these operations as unconsciously as I type `f1 O C L P c6.10` to tell EASY SCRIPT to print the text contained in a series of linked disk files starting with the one called "c6.10."

- **QUICK BROWN FOX**, from Quick Brown Fox, Inc. This is another popular word processor for the C-64 that has lots of features but an awkward style. QBF is available in cartridge as well as disk format, and it works with tape or disk, so you don't need a disk drive to use it. It's written in machine language, and it's compatible with most 80-column converters for the 64 (see Chapter 8). But it was apparently designed on a computer that didn't support full-screen editing capabilities, because it operates as a LINE EDITOR that can only work with one line at a time. In addition, QBF separates INPUT and EDIT operations into two different menu choices. The combination of these two factors makes QBF very awkward to work with. For example, if, when you're entering text, you notice an error on the previous line—or you decide you want to reword something—you have to switch to edit mode before you can do anything about it. The process of replacing a phrase is much more complicated than it is with EASY SCRIPT.

QUICK BROWN FOX has its strong points. It's faster, it has an easy-to-use menu format, and it's well documented and attractively packaged. But with all of the word processors available for the C-64, a line editor like this one certainly wouldn't be my first choice.

- **MIRAGE WORD PROCESSOR**, from Mirage Concepts, Inc. Like SCRIPT 64, the MIRAGE WORD PROCESSOR boasts a gigantic list of features, including the important choice between 40- and 80-column screen display. But with this product, you don't pay for those features by sacrificing convenience and ease of use. A straightforward menu offers choices between the major functions of the program: edit file, save file, load file, merge files, delete file, format disk, directory, print file (for

most documents), advanced print (for documents with unusual formatting requirements), and quit. Once you're in EDIT mode, you can access most of the powerful editing features with simple keystroke commands. The editor has enough features that it will take you a while to learn, but the excellent manual contains a very clear tutorial.

Probably the most important of these features is the 40-80 column display toggle, which allows you to create your document with a 40-column screen, then switch over to 80 so you can peek at the product, and then switch back to 40 to make changes. It's possible to do all of your editing in 80-column mode, but it's a lot slower than in 40. In both modes, the display features complete word wrap, so words don't get split between lines as they do in EASY SCRIPT. There's a price for these advanced display features in speed and space: most editing goes a little bit slower here than in EASY SCRIPT, and documents can't be quite as big without chaining.

But if you've got a non-Commodore printer, you'll appreciate being able to create printer files to customize your printout. printouts. And if you do many cut-and-paste kinds of operations, you'll like the fancy editing and display capabilities. The ring-bound manual that accompanies the program is very thorough, and offers excellent support if you run into a problem, an important factor with a new piece of software like this one. This is a professional product for people with professional needs.

There are lots of other good (and not-so-good) word processors on the market for the Commodore 64. If you have a clear idea of the features you're looking for, it pays to shop around until you find one with those features. But for my money, the features aren't as important as the FEEL of the program when you're using it. I like to move things around as I write, experimenting with different wordings and organizational schemes. I want my word processor to make that as easy and as natural as possible. I like to adjust the colors of the background and text, depending on my mood and the mood of my finicky TV. My word processor has to make that easy, and so on. I still haven't found my perfect word processor, but I've found several that feel pretty good. Hopefully you'll be able to try several before you buy so you can pick the one that feels right for you.

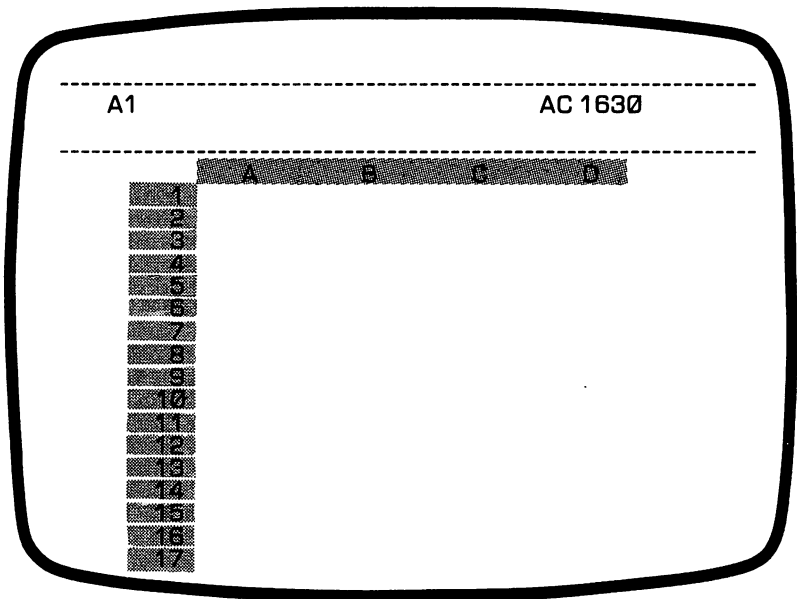
CALC POWER: AT HOME WITH SPREADSHEETS

Word processing software is a wonderful way to make your micro into a powerful tool for working with words. There's another kind of program that does for numbers what word processors do for words: the calc-type spreadsheet program. A good calc program can make short work of almost any job that involves repetitive calculations: budgeting, checkbook balancing, menu conversion, teacher gradebooks, student math/science projects and drills, and so on. Plus, it can give you information about number relationships that can take a lot of the guesswork out of financial planning and speculation.

The original program in this genre was conceived in 1978 by Dan Bricklin during a class at Harvard Business School. As he watched his professor continually erase and recalculate rows and columns of numbers during chalkboard exercises in corporate financial planning, he envisioned a computer program that would do it all on a screen automatically. With MIT friend Bob Frankston, he turned that idea into VISICALC. That program has become what may be the biggest selling piece of software ever written! In fact, many analysts believe that the invention of VISICALC is largely responsible for Apple's phenomenal business success.

Success breeds imitators, and the software shelves today are crowded with Visi-clones. Like word processors, these programs vary in their size, features, packaging, and promises. But they are similar enough that once you become familiar with one of them, you should have little trouble working with other brands. The examples in this section are designed to give you an idea of the possibilities for using one of these programs in your home. For the sake of concreteness, they were written to work as-is on one specific program: CALCRESULT, by Handic Software. But they'll work with minor modifications on any of the calc programs discussed at the end of this section.

All spreadsheet programs are based on one simple concept: the malleable matrix. When you load CALCRESULT into your computer, your screen looks like this:

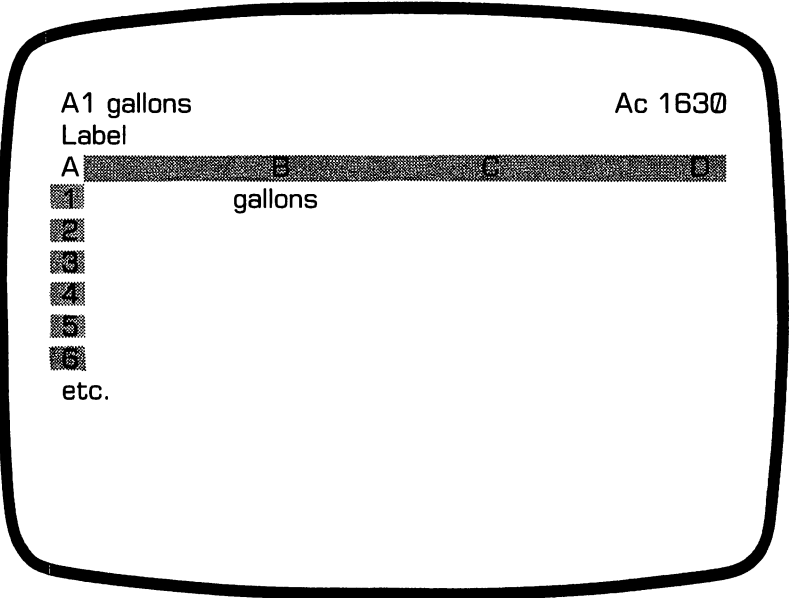


Except for the top three lines, the screen is divided into rows, numbered 1 through 21, and columns, lettered A through D. These rows and columns make a grid of 84 CELLS, each of which can be referred to by one letter and one number. For example, the cell in the upper left-hand corner of the grid is called cell A1. That's where the cursor sits, which explains the A1 on the top line.

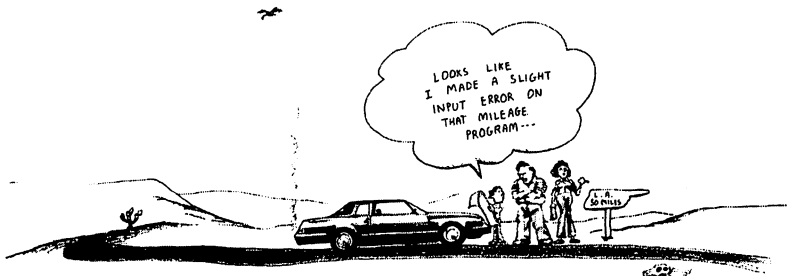
Of course, you can use the cursor-move keys to move it around the screen: pressing the cursor-right key moves the cursor to cell B1; pressing it again moves it to C1; cursor-down would then move it to C2; pressing home moves it back to A1. All of these cells are empty when you start; it's up to you to fill them. Each cell can contain either a number, an alphabetic label, or a formula. This is easiest to show with a simple example.

Let's say you want to calculate your car's gas mileage. To keep things simple, we'll assume you know the number of miles traveled between fillups and the number of gallons it took to fill the tank the second time. You want to know the number of miles per gallon: miles/gallons. Let's start by putting some

labels across the top. Different calc programs identify labels in different ways; in CALCRESULT, you indicate that a cell is going to contain a label by typing a space first. So to put the label "gallons" in cell A1 (the cursor's current location), you type "gallons," followed by a RETURN. As you type, the word "gallons" appears on the top line of the screen, next to the A1, and also in cell A1:



The screen's top line is effectively saying that the current cell, A1, contains the label "gallons." To put the label "miles" in cell B1, you cursor-right once and type "miles." Cursor-right once more to type "mpg," and your labels are done.



In cell A2, you'll put the number of gallons, say 13. If you don't precede the "13" with a space **CALCRESULT** assumes that the 13 is a numeric **VALUE**, rather than a label. The same goes for the number of miles, 282, when you type it into B2. C2 is where the **FORMULA** for miles per gallons goes. Spreadsheet formulas generally look like formulas in **BASIC** (see Chapters 4 and 5), except that cell names are used instead of variable names. Since the number of miles is stored in B2, and the number of gallons is in A2, the formula for mpg will be represented by **B2/A2**. Once again, no space in front, because this is not a label; it's a formula that represents a value. As before, your typing appears at the top of the screen and in the current cell. But as soon as you hit **RETURN**, the formula in the cell disappears, and the numeric value 21.69230 appears:

A1		B2/A2		AC 1630	
Value					
1		A	B	C	D
2		gallons	miles	mpg	
3		13	282	21.69230	
4					
5					
6					
etc.					

Now suppose you notice that you miscalculated the mileage. No problem; just move the cursor back to B2 and type in the new value, say 292. As soon as you hit **RETURN**, the mpg figure in C2 changes to reflect the new mileage!

The importance of this feature may not be obvious to you from this simple example. But think back to the last time you added a long column of figures with a calculator (balancing your checkbook? doing your taxes?) and had to repeat the whole job because of a minor error. Remember? If you had used a program like this one as your calculator, correcting your error would be as simple as retyping a single number. Of course, you'd need to set up the spreadsheet first. But that's easy, too.

Starting with a clean slate, simply move the cursor to C1 (or anywhere you like) and type the formula "SUM(A1:A21)". This is the way you tell CALCRESULT to add all of the numbers in the cells between A1 and A21, and put the result in the current cell. (Other calc programs have similar functions, although they might not look quite the same.) When you hit RETURN, the formula is replaced by a zero, because there are no numbers in column A. Each time you stick a number into one of the cells of that column, this sum value will change accordingly.

Do you have 35 numbers to add? Just change the 21 in the formula to a 35, and enter your numbers in column A. When you reach the bottom of the screen, a cursor down will simply bump row 1 off the screen, revealing row 22. You see, you're actually looking through a window at a small portion of your spreadsheet; you have a total of 63 columns and 254 rows to work with. (These numbers also vary with different products.) When you've entered all 35 numbers, you can get back to the original screen by pressing HOME.

SUM is just one of many functions available with CALCRESULT; you could, if you wanted, put the MEAN of these numbers in C3, the COUNT in C5, the STDDEV (standard deviation) in C7, the MINimum in C9, and the MAXimum in C11. You can even build IF statements like "IF A1>100 THEN A1*1.10 ELSE A1*1.05", and put them into cells. Formulas can refer to cells that are made up of other formulas, making it possible to create ridiculously complex mathematical relationships with a few keystrokes. Anytime a value is changed anywhere on the grid, all of these formulas correct themselves for you automatically!

(For large, complicated spreadsheets, recalculation can be kind of slow, so CALCRESULT allows you to turn off the automatic recalculation feature. That means that formulas don't recalculate unless you say so. Some spreadsheet programs operate this way normally.)

A program like this has thousands of applications—it's sort of a programming language in its own right. To start your idea factory, here are a couple of simple household applications.

Let's start with a checkbook calculator for lazy folks. Across the top, in big letters, we'll put the title: CHECK BALANCER (cells B2 and C2). Next row, our headings: "checks" and "deposits" (B3 and C3). Each of these will head a column of numbers, which we'll want to sum toward the bottom. We could make the columns as long as we needed, but let's keep them short so they'll fit on one screen. In B19, we'll enter the formula SUM(B3:B17), and in C18, SUM(C3:C16). C19 will just contain a copy of B19, the sum of the checks, and C21, the difference between that number and the sum of the deposits: C18-C19. A few labels and beauty lines, and our simple checkbooker looks like this:

	CHECK	BALANCER
	checks	deposit
516	5.90	2582.07 old bal
517	42.55	993.54 pay
518 fuel	139.00	200.55 bonus
520	15.68	
522	8.95	
523	58.13	
524	29.00	
525	10.00	
526	200.00	
527 hous	520.48	
528	15.00	
	-----	-----
	1044.69	3776.16 total+
		1044.69 total-

		2731.47 balance

There are two ways you can use this spreadsheet. At balancing time, you can enter all of the checks and deposits that have been processed by the bank to make sure their end-of-month balance matches yours. If it doesn't, all of the numbers are right there in front of you for examination and modification. Columns A and D can be used for check numbers and explanations, so that you can print a descriptive copy of the spreadsheet or save it on disk. Of course, this method doesn't deal with checks that haven't yet cleared the bank, but it could easily be modified to do so.

Some of us are only concerned about the ones that haven't cleared. We rarely keep our balances up-to-date, and we don't want to be bothered with checking the bank's arithmetic. All we want to know when we check the bank statement is, "How much money is left in the account?" If you recognize yourself in this description, this spreadsheet template can help you, too. When you get your statement each month, just enter the UN-CLEARED checks in the column labeled "checks," the bank's closing balance at the top of the "deposits" column, and the unprocessed deposits under that. The balance shown at the bottom, then, is your actual current balance. Hopefully positive.

Finally, let's go back to our simple gas mileage example, and try to expand it into something more useful. How about an ongoing log of gas consumption that keeps up-to-date figures on gasoline consumption, expenses, and mileage? We'll label it GAS CALC, and have columns for entering the gasoline purchase date, odometer reading, amount spent, number of gallons, distance travelled since last fillup, and miles-per-gallon on that tankful. That's six columns, and only four show on the screen. We could, of course, just cursor back and forth to look at the whole thing. But it's also possible to change the widths of the columns to six characters, so that all six columns fit in the display:

A1	AC 1630					
	A	B	C	D	E	F
1		GAS		CALC		
2	date	odom	cost	gals	dist	mpg
3						
4						
5						
6						
	etc.					

You'll enter the relevant information into the first four columns each time you buy gas, or more realistically, once a month from receipts. Information in the last two columns will be calculated using the information from the first four columns of this row and the previous row. (Since there is no previous row for the first entry, you'll just put zeroes in columns E and F of that row.) So E5 will contain the formula $B5 - B4$ (current odometer reading minus reading at last fillup), and F5 will contain the formula $E5/D5$ (distance in miles divided by gallons). For each additional row in your log, the formulas in columns E and F will be the same, with the row numbers adjusted accordingly. Because spreadsheet programs allow you to REPLICATE a formula with RELATIVE values, it just takes a few keystrokes to fill these columns with the correct formulas.

Toward the bottom of the screen, you'll draw a bottom line, and enter some SUM formulas for columns C, D, and E. As a bonus, you can throw in the MIN, MAX, and MEAN for column F, the average cost per gallon ($C17/D17$), and the average cost per mile ($C17/E17$). Here's what the finished spreadsheet looks like after a few tankfuls:

P 1	A	B	C	D	E	F
2	date	odom	cost	gals	dist	mpg
3	-----					
4	1/1/4	55328	0	0	0	
5	1/4/4	55642	14.85	12	314	26.17
6	1/7/4	55998	15.92	13	356	27.38
7	1/10/4	56333	14.85	12	335	27.92
8	1/12/4	56712	12.22	11	379	34.45
9					NA	NA
10					NA	NA
11					NA	NA
12					NA	NA
13					NA	NA
14					NA	NA
15					NA	NA
16	-----					
17	totals		57.84	48	1384	
18						
19	ave	\$/gal	1.205	max	mpg	34.45
20				min	mpg	26.17
21	ave	\$/mi	0.042	ave	mpg	28.98

Hopefully, these simple examples give you a feel for the flexibility of spreadsheet programs. The truth is that we've barely cracked the egg. The real power of spreadsheet programs is in their ability to do financial projections, and answer questions like, "Would I make more money putting my savings in a high-yield IRA tax-sheltered account with a withdrawal penalty, or a low-yield passbook account with no penalty?" There's no room here to develop such a model, but there are lots of books and magazine articles on the subject. 🐾

There are several spreadsheet programs available for the Commodore 64, and more on the way. Here are some of the best:

CALCRESULT, made by Handik Software, distributed in this country by Computer Marketing Services, Inc. This is the one we've been using in our examples. It comes in two forms: an "easy" cartridge that plugs into the back of your 64, and an "advanced" version that uses both cartridge and disk. The advanced form is one of the most versatile spreadsheets available

🐾 For example: *Creative Computing*, February, 1983, page 222.

on any computer at any price. Its most unique feature is that it allows you to create up to 32 different spreadsheets on separate "pages," effectively making a three-dimensional spreadsheet. (In order to accommodate large multiple-page spreadsheets, CALCRESULT uses a work area on disk to store data when memory fills up.) CALCRESULT allows you to split the screen so that you can look at two different pages at once, or create "windows" to peek at data from other pages or other parts of the current page.

Like most advanced spreadsheets, CALCRESULT contains editing commands for inserting, deleting, and rearranging information on the sheet. It also affords tremendous flexibility for formatting spreadsheet printouts, including individual column-width settings and graphics for bar charts, etc. CALCRESULT allows you to save data on disk for later use whenever you choose. But it also remembers where you left off, when you quit a session, so that it can bring back your current worksheet automatically at the beginning of your next session. The instruction manual is complete (except for an index) and easy to read, in spite of the occasionally awkward translations from Swedish. But you don't need to refer to the book very often once you've gone through the introductory tutorial, because most of the essential reminders can be displayed on the screen at the touch of a "help" button.

The "easy" version is basically the same program without the help screens, multiple pages, and automatic recall of last session's data. It's easier and faster to load and use than the advanced version, and it costs about half as much. For most home uses, it's more than adequate.

PRACTICALC and PS, from Computer Software Associates. PRACTICALC is a bargain-priced spreadsheet that's ideal for home or small business use. You won't find all of CALCRESULT's advanced features here, such as multiple pages, split screens, windows, and help screens. Early versions of PRACTICALC also lacked automatic recalculation, variable column width printouts and IF THEN ELSE functions, but newer packages include all of these features, along with three

useful options that even CALCRESULT doesn't offer. Once command allows you to print out all of the underlying formulas used in a spreadsheet, so you don't need to cruise around with the cursor to see them. A SEARCH command tells PRACTICALC to search for a particular name or number in the matrix. A SORT feature allows you to rearrange rows of information in numeric or alphabetical order based on a particular column. This SORT feature greatly increases the usefulness of the program for applications like storing student grades, household budgets, phone lists, and so on.

Because it's simpler, PRACTICALC is a little easier to get to know than CALCRESULT. The manual is well-written even though it is a bit brief. (A minor point: PRACTICALC uses letters for rows and numbers for columns, going against the VISICALC tradition, which could be bothersome when borrowing spreadsheet ideas from books and magazines.)

PRACTICALC's big brother is called PS, for Programmable Spreadsheet. Designed for scientific applications, it has all the features of PRACTICALC plus programmability. That means that you can write short BASIC programs to modify your spreadsheet (e.g., reverse the rows and columns). PS costs only a few dollars more than PRACTICALC.

MICROSOFT MULTIPLAN, from HesWare. Considered by many to be the state of the art in spreadsheet programs, MULTIPLAN has become one of the most popular worksheet programs on other personal computers. With its many powerful features, easy-to-use design, and professional documentation, MULTIPLAN makes most CALC programs look crude by comparison. The Commodore 64 version of MULTIPLAN offers all of these advantages at a fraction of the cost of previous MULTIPLAN packages.

MULTIPLAN has just about all of the features of CALCRESULT, the most notable exceptions being the bar-chart graphics feature and the 32-page spreadsheet option. (This multi-page option can be simulated by linking different spreadsheets from disk.) MULTIPLAN also has several additional commands that add to its power, including a handy SORT command for rearranging rows in alphabetical or numerical order. But the features of MULTIPLAN that really sets it apart from the others are the ones that make it easy to learn and use: the 400-page indexed manual, complete with a thorough tutorial and a reference section, the quick-reference card and keyboard function-key overlay, the excellent help screens that can give hints and reminders anytime at the touch of a couple of buttons, and the unique cell-naming option that lets you give meaningful names to cells, so that a formula can be MILES/GALS instead of B2/A2. (In fact, B2/A2 won't work at all on MULTIPLAN; the correct notation would be R2C2/R1C2.)

All of this friendliness and power won't fit into 64K at one time, so MULTIPLAN has to access the disk frequently to swap information in and out. That takes time, but for most applications, the trade-off is well worth it. All things considered, MULTIPLAN is one of the most professional pieces of software yet released for the 64.

USING YOUR COMPUTER AS A FILING CABINET

Word processors, calc programs, and database managers were developed for use in office environments. As you've seen, word processors and calc programs can be adapted for use in the home. The same is true for database managers—programs that allow you to store and retrieve information in various forms. Most of us do a certain amount of data storage and retrieval in the course of our day-to-day existence: names and addresses of friends, co-workers, and businesses, important dates and appointments, recipes, tax records, and so on. It's certainly possible to computerize any of these operations with a database program. The question is whether the benefits (accuracy, flexibility, and speedy retrieval) outweigh the costs (the time and effort of setting up the system and keeping it up to date).

There are three major problems with using a database system at home: (1) You have to type in the data before you can use it; (2) You have to start up the program every time you want to put anything in or get anything out; and (3) equipment failure or operator error can wipe out your data in a hurry if you don't religiously back up your data disks. (If you haven't yet introduced yourself to the 1541 BACKUP program on the Commodore Bonus Disk Pack, this is a good time.) For some home applications, these aren't major problems. The average person wouldn't get finger blisters from typing her name and address book into the computer, and wouldn't have to update the list very often. But for other home applications — recipe files, appointment calendars, record book collections — the tedium of data entry and the inconvenience of using the program may outweigh the potential benefits.

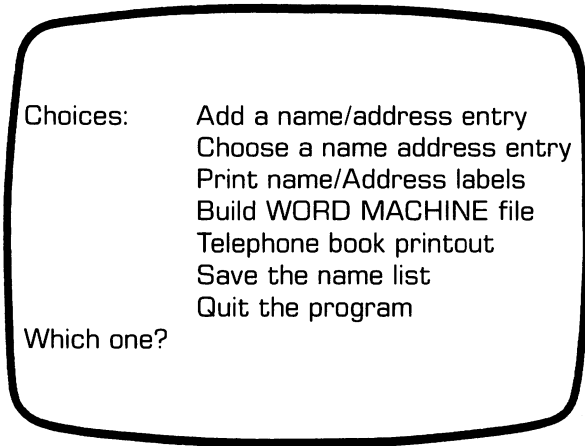
Accordingly, this section won't explore the intricacies of database programs. Instead, we'll just take a look at one simple data storage program—a mailing list program—and an overview of the rest of field.

But first, a jargon break. As you may remember from Chapter 4, data is stored on disk or tape in *files*. A file is made up of individual chunks of data called *records*. In the case of a mailing list program, each record would contain the name and address; in the case of a library catalog, each record would contain information on one book. Each record is made up of individual *fields* of information, such as last name, zip code, title, etc. Most database programs allow you to search for information in particular fields using *keywords*. You might, for example look for names of people living in Oregon, or books with the word "Peace" in the title. More sophisticated programs allow you to make more detailed requests — such as the names of all left-handed computer-owners within area code 503 — if you have taken the time to learn the system and correctly feed the information in. In general, the fancier programs demand more on your part.

THE NAME MACHINE. This Commodore program for storing names and addresses accompanies the WORD MACHINE, discussed earlier in the chapter. Designed for simplicity and economy, THE NAME MACHINE is *not* a general purpose

database manager, but it has features that are common to all data storage programs, so it's a good place to start.

After asking you whether you're using tape or disk for storage, the NAME MACHINE asks whether you're starting a new data file. If you answer no, it asks for the name of the file, and loads it into memory before displaying the main menu:



If you're using the program for the first time, you'll need to start by adding some entries to your currently empty list. The NAME MACHINE prompts you to enter each field—last name, first name, address information, and telephone number. Next, it asks you which of seven categories the entry belongs in: family, friends, workmates, etc. (you can easily change these categories). The categories are used when you retrieve data later. They allow you, for example, to ask for a phone list of family names only, or a set of mailing labels addressed to all co-workers. When you're done entering the information, you're given the opportunity to check it and change it before it becomes part of the file. After you've added all the names you want to add, you're returned to the main menu.

The next option on the menu, Choose a Name/Address entry, allows you to search through the list for a particular keyword

in a field: a last name of Smith, or a phone number of 555-9100, for example (handy for deciphering mystery long-distance charges on your phone bill). You can even ask to see all of the entries with last names that begin with "Co", if you can't remember the exact name. Names that fill the request are displayed in order until the list is exhausted or you tell it to stop looking. Each time a match is found, you're given the opportunity to make changes in the record or delete it from the list.

The mailing label option allows you to print four-line labels for any or all of the categories of names. You can, if you want, preview the entries on the screen and choose to print only certain labels from a category. Phone numbers, of course, aren't printed.

The next option creates a file for generating form letters on the WORD MACHINE, as discussed earlier. Once again, you can choose the names that go into the file.

Telephone book printout produces a complete or partial list of names, addresses, and phone numbers, depending on you request. Unfortunately, you can't change the ordering of the list—they come out in whatever order they were originally entered.

The save option is used to save the name list as a disk or tape file with a name of your choosing. (If you choose a file name that already exists on the disk, the program checks to make sure you want to wipe out the old one.) You have to save the list if you've made any additions or changes during the current session, or those changes will be lost when you quit the program. Fortunately, the program reminds you when you quit if you've forgotten this important detail.

All in all, the NAME MACHINE does what it does very well. It's easy to use. It's written in BASIC, so it's easy to modify. And it's cheap. But don't expect to run your business using this program; it lacks several important features. For one thing, it brings the complete data file into the computer's memory while you're using it, so it can only handle files of up to 150 records at a time. It has no option for sorting the items in the list alphabetically or numerically. And of course, it's slow.

Other mailing list programs. If you need a fancier mailing list program than the NAME MACHINE, you have plenty of choices. Here are a few.

- **TOTL LABEL 2.6.** From TOTL Software, is the obvious choice if your using TOTL TEXT as a word processor. It's designed to be compatible with that program in terms of command format, and the two can be used together for generating form letters. TOTL LABEL sorts records alphabetically or numerically, and it allows you to define the record and field sizes. If you're resourceful, you can use this program for storing recipes or collections as well as information about people. It has good documentation.

- **EASY MAIL,** from Commodore. This one is designed for professional use. It works with the data directly on disk, so it can handle much larger files than can a memory-based program. It has easy-to-use sort and search options, and easy-to-read documentation. But surprisingly, there's no apparent way to use EASY MAIL with EASY SCRIPT to generate form letters!

Other specialized database programs. If your data storage needs go beyond names and addresses, you might want to look at one of these programs:

- **RESEARCH ASSISTANT 2.0** (Keyword Cross Reference), from TOTL Software. This is a system of two programs designed to replace 3x5 cards as a tool for keeping track of information from books and periodicals. It allows you to store notes from articles and books along with keywords, so that you can create cross-referenced lists of sources for term papers, articles, or arguments. With this system, you can retrieve information using up to three keyword and date ranges. Good instruction manual, as usual.

- **DIARY 64,** from Handik Software, distributed by Computer Marketing Service, Inc. This program from the CALCRESULT people comes in a cartridge, so it's ready to go when you turn the machine on. It's an easy-to-use general database program, but you'd never know it from the poorly-translated instructions. The menu is straightforward, but it uses color combinations that are hard to read on some TVs. The program is

designed to store names and addresses as well as appointment-calendar entries and miscellaneous. Each record can contain up to 10 lines of 27 characters each, but only the first four lines of a record (presumably name and address) can be printed. While this program isn't flexible enough to meet everybody's needs (you can't sort the data for example), you might want to consider it if you're looking for an electronic appointment book.

General-purpose database programs. Purchasing a generic database program is a good idea if you have several different kinds of data to store. It can not only save you money, but it can save you the bother of learning and remembering how to use several different programs. Here are some choices for C-64 owners.

- **DATA MANAGER**, from Timeworks, Inc. This is a low-cost general-purpose database program, allowing records to have up to eight lines of 30 characters. It allows you to establish keyword classifications, and has features for easy statistical analysis of data. You could use this one to store anything from data on your insect collection to grandma's favorite recipes. In part because of its flexibility, it's not as easy to get started with as some of the others listed here. The documentation is readable, but short on examples.

- **FLEX-FILE 2.1**, from A. B. Computers. **FLEX-FILE** costs more than most of the other programs listed here, but considerably less than most "professional" database programs with similar features. Like **DATA MANAGER** and **DIARY 64**, it's a store-everything-you-want-in-it database manager. But it's much more flexible than either of these two programs, and more powerful, too. Even if you're just using it for mailing lists, you'll appreciate its flexibility. You can set up records to contain as many fields as you like, and you don't need to specify how big those fields are going to be. Records are automatically stored in whatever order you specify, and you can specify that they be ordered in two different ways (e.g., zip code or last and first name). You decide which fields appear where on the labels, and which records get printed as labels. You're guided through all of these choices by straightforward menus and questions, and a jargon-free manual.

FLEX-FILE also contains a report writer program that allows you to print out custom business-style data reports with enough subtotals to impress the most calculating middle manager. It lets you change the structure of your file anytime you decide you need more fields. And if that's not enough flexibility for you, you can change the program—it's written mostly in BASIC, and the manual gives you modification tips.

● **MIRAGE DATABASE MANAGER**, from Mirage Concepts, Inc. Another professional-quality program with a surprisingly low price tag, this program looks more like the big-league databases than any of the others reviewed here. It comes packaged in a handsome three-ring binder with a thorough tutorial and reference guide. It's written in machine language, so it's fast. And considering its power, this program is amazingly easy to use. Unlike **FLEX-FILE**, the Mirage program requires you to specify field-lengths for all of the fields in your record. But that's as simple as drawing lines on the screen with an underline key. You set up your record format by simply designing a form on the screen. There's no need to specify field names or numbers, because all field transactions — data modifications, sorting, printout design, etc. — involve simply moving the cursor to the desired field on the screen.

There's no room here to describe all of the capabilities packed into this menu-driven program — multi-level sorts; saving sort indices; printing lists, reports, forms, or mailing labels; conditionals (BASIC IF statements); calculating fields; merging files; and creating sequential files for use with word processors or BASIC programs, to name a few. The manual is well-written and thorough in its explanation of all of these features, but the menu-driven program is easy enough to use that you're not likely to need to refer to it very often.

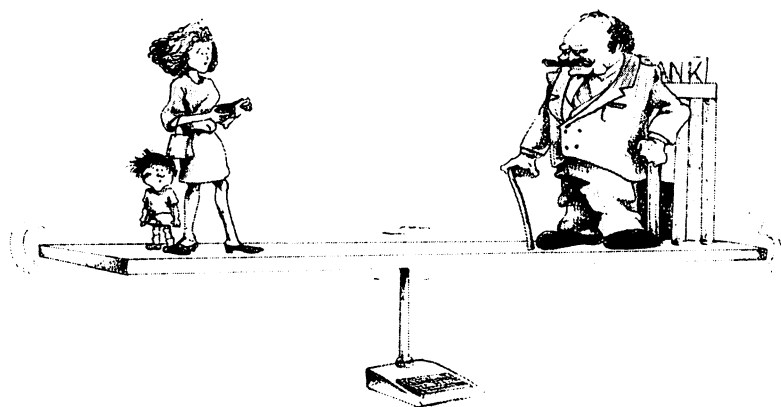
TAKING CARE OF BUSINESS

We've covered the three main categories of business programs for micros. That leaves a handful of programs that don't easily fit into any of these categories: special-purpose household helpers. 🐾

🐾 The definitions here get fuzzy: you'd be right if you argued that a checkbook record-keeper is really a kind of database program.

● **64 CHECKBOOK MANAGER**, from Data Equipment Supply Corporation, is a menu-driven checkbook program that leads you by the hand through the processes of creating a check file, entering and saving check-stub data, clearing canceled checks, balancing your account, and printing detailed reports for tax purposes or general information. It allows for up to 250 outstanding checks at a time, and even has a password feature to keep prying eyes from snooping in your account.

CHECKBOOK BALANCER



While this program is certainly easy enough to use, it has the same problem that curses all checkbook programs: somebody's got to type all the data in. In theory, you're supposed to load the program to enter data **EACH TIME YOU WRITE A CHECK**, and load it again to balance the account each month. Not likely? Then you can let the checks accumulate and enter data whenever you want or need to know the current balance. Or you can enter everything at balancing time, if up-to-the-minute balances aren't important to you. If you do it this way, allow an hour for typing. If it normally takes you that long to balance your checkbook anyway, use this program (or one like it) and you'll save yourself headaches at tax time.

(If you REALLY don't mind typing, you can type in the simple checkbook checker included in Appendix A of this book.)

● **THE HOME ACCOUNTANT**, from Continental Software, and **THE COMPLETE PERSONAL ACCOUNTANT**, from Programmer's Institute. If you're going to take the time to type in your checks, you might as well make the most of that investment by typing it inot a full-service accounting package. Besides just balancing your checkbook, these two popular home accounting packages will maintain accurate records for taxes, track detailed household budgets, print checks and net worth statements, display financial graphs, and handle just about all of the nitty-gritty financial details of running a household or small business.

Of course, they can't do it all by themselves; you'll need to decide exactly what you want from them, set up and initialize accounts and budget categories, and accurately record your income and out-go. But you don't need a degree in accounting to do these things; both of these menu-driven programs are easy-to-use and clearly-documented for beginners. If your financial affairs are relatively complex, and if you're willing to make the initial down payment and regular monthly payments in time, you'll get a handsome payoff on your investment in terms of simplified money management, accurate records, and valuable feedback on your family's financial habits.

CHAPTER 8

Creating Your Own Software: When The Bug Bytes

I speak Spanish to God, Italian to women, French to men, and German to my horse.

—Charles V

A book on furniture building might start with a chapter on how to use a hammer and a handsaw. Those tools are easy to learn, and they aren't likely to amputate anything while you're turning the page. But the book wouldn't be complete if it stopped there. If you're serious about building furniture, you'll want to learn about other tools that can save labor and improve the quality of the finished product. More importantly, a book of this type should discuss the processes of furniture design and construction, so that you can learn how to turn an idea into a blueprint into a chair.

Unfortunately, most how-to-program-your-microcomputer books stop after Chapter 1. They run through laundry lists of BASIC keywords and memory locations, showing how each of these mysterious symbols can be used to make your magic box do tricks. There's nothing wrong with this approach; it's fun, and it's a good way to learn the computer's BASIC vocabulary. It can give you the background you need to write small programs by putting together different combinations of statements until you get the results you want, and for most casual computer users, that's good enough.

But at some point, for whatever reason, you may find yourself trying to write a large, complex program. Maybe you'll need a program to coordinate the training regimen for your racing turtle, and you won't be able to locate a commercial software package to do the job. Or maybe you'll have a cosmic vision while you're putting on your socks, and become obsessed with turning it into a video game. In any case, you'll need to know more than BASIC keywords to write large, professional-quality programs. You'll need to know how to systematically design

programs so that you aren't so dependent on design-by-accident. And you should know about the power tools that you can use to speed up the programming process, improve the quality of your finished product, and generally make programming easier and more fun.

This chapter won't make you into a master programmer, but it will get you started in that direction. It begins with a short discussion of the process of designing programs, and then moves on to a survey of programming tools. These tools range from simple extensions of your computer's BASIC language to whole new programming languages that are much more powerful (and, in some cases, easier and more fun to use) than BASIC. By the end of the chapter, you'll be better equipped to choose the right tool for the job, whatever that job might be.

THE ART OF PROGRAM DESIGN

As a human activity, computer programming is a relative newcomer. But it's really nothing more than a specialized form of the age-old process of problem solving, using a special tool: the computer. Solving any kind of problem involves going through a series of steps, with or without our conscious awareness:

1. Defining the goal: "Where do I want to go?" (Since the goal is to solve the problem, this means clearly identifying the problem.)
2. Figuring out what resources are available for solving the problem: "Where am I?"
3. Applying those resources to the problem in such a way that the goal can be reached: "How do I get there from here?"

Steps 1 and 2 are fairly straightforward, so they're often overlooked. But they're important nonetheless. If you want to solve a problem, you've got to be able to state the problem clearly. No less important is an understanding of the available tools: the computer, the computer language, and the input data you have to work with. Step 3 is harder to pin down. Often it takes care of itself as soon as the first 2 steps are done, consciously or unconsciously. In other cases, a kind of mental trial-and-error process brings results. But for big problems with complex

goals it's sometimes necessary to resort to pencil and paper, research, brainstorming, or a computer to reach a solution.

When you use a computer to solve a problem, it's necessary to develop a specific set of instructions for solving the problems. The computer's unrelenting demands for details can obscure the goal and make the process much more difficult and error-prone. So most professional programmers start by developing an overall plan, or ALGORITHM, [🐾]without concerning themselves with the details of a programming language. The algorithm can be vague at first; it may be nothing more than a list of subproblems that need to be solved in order to solve the original problem. Each of these subproblems can then be analyzed individually in the same way, until enough detail exists to easily translate the algorithm into a computer program. This process of STEPWISE REFINEMENT of the algorithm is similar to the way a writer develops an outline before writing a paper or a book.

The easiest way to see how this works is to look at a simple example. So let's develop a program to play a guessing game. In this game, the computer will pick a number between 1 and 100, and give the player seven tries to guess the number. [🐾]Here, then, is the algorithm, ignoring the details for the game:

BEGIN GAME.
REPEAT TURN UNTIL NUMBER IS GUESSED OR
SEVEN TURNS COMPLETED. END GAME.

That was easy! Now let's fill in some of the details of the game:

[🐾]This strange word comes from the name of a ninth-century Persian textbook author, in case you were wondering.

[🐾] Why seven tries? Because that's all you should need, if you use the right strategy. (The "right" strategy in this case is affectionately referred to by computer scientists as the BINARY SEARCH.)

```
-----  
BEGIN GAME:  
  GIVE INSTRUCTIONS.  
  PICK A NUMBER BETWEEN 1 AND 100.  
  REPEAT TURN UNTIL NUMBER IS GUESSED OR  
  SEVEN TURNS COMPLETED.  
  INPUT GUESS.  
  RESPOND TO GUESS.  
END GAME.  
-----
```

We still have some details to attend to. How, for example, will the computer respond to a guess? Let's replace RESPOND TO GUESS with this:

```
-----  
IF GUESS = NUMBER, THEN SAY SO AND QUIT;  
ELSE, IF GUESS < NUMBER, THEN SAY SO;  
ELSE, SAY GUESS IS TOO BIG.  
-----
```

Finally, we've got to give the computer a way of knowing when seven turns are used up. We can do that by setting a counter to 0 initially and adding 1 to it after each turn. If all seven turns are used, the looping stops and the computer should print a message. That gives us the following algorithm:

```
-----  
BEGIN GAME.  
  GIVE INSTRUCTIONS.  
  PICK A NUMBER BETWEEN 1 AND 100.  
  SET COUNTER TO 0  
  REPEAT TURN UNTIL NUMBER IS GUESSED OR  
  COUNTER = 7:  
    INPUT GUESS.  
    IF GUESS = NUMBER, THEN SAY SO AND  
    QUIT;  
    ELSE, IF GUESS < NUMBER, THEN SAY SO;  
    ELSE, SAY GUESS IS TOO BIG.  
    ADD 1 TO COUNTER.  
  END GAME:  
  PRINT MESSAGE.  
-----
```

It's now a fairly simple task to take each section of this algorithm and turn it into BASIC code, without getting distracted by other parts of the algorithm. Here's what the finished program looks like:

```
-----  
10 REM NUMBER GUESSING GAME  
100 PRINT "LET'S PLAY A GUESSING GAME."  
110 PRINT "I'LL PICK A NUMBER BETWEEN 1 AND  
100,"  
120 PRINT "AND YOU TRY TO GUESS IT."  
130 PRINT "I'LL GIVE YOU SEVEN TRIES."  
200 N = INT(RND(0)*100) + 1  
210 T = 0  
290 REM REPEAT LOOP STARTING HERE  
300 INPUT "WHAT'S YOUR GUESS";G  
310 IF G=N THEN PRINT "YOU GOT IT!": END  
320 IF G<N THEN PRINT "TOO SMALL, GUESS  
AGAIN.":GOTO 400  
330 PRINT "TOO BIG, GUESS AGAIN."  
400 T = T + 1  
410 IF T<7 THEN GOTO 300  
500 REM END OF GAME  
510 PRINT "I FOOLED YOU SEVEN TIMES!"  
520 PRINT "THE NUMBER WAS";N;"!"  
530 END  
-----
```

Maybe this seems like more trouble than it's worth. For a program of this size, that may be true. But for bigger programs with more complex logic, this design process can save a lot of time and headaches. The nice thing about developing algorithms in this way is that you don't need to worry about the

details of the computer language—or even choose a language—until the design work is done. The algorithm is represented in PSEUDOCODE, which is a kind of cross between a computer language and plain English. 🐾

MAKING BASIC BETTER

One of the reasons that pseudocode is so appealing as a program design tool is that it allows programmers to express their ideas in a language that's more human-oriented than BASIC. It's much easier to think in phrases like PICK A NUMBER BETWEEN 1 AND 100 than in statements like $N = \text{INT}(\text{RND}(\emptyset) * 100) + 1$. It would unquestionable make programming a lot more pleasant if the computer could understand those English-like phrases without the aid of a human translator. Why shouldn't we be able to let the computer take care of those boring details?

That isn't as far-fetched as you might think. If you read about LOGO in Chapter 6, you've already seen an example of a modern computer language that looks a lot more like pseudocode than it does BASIC. You'll see another shortly. These languages contain single statements that do the work of many BASIC statements, making some programming tasks much easier. But more importantly, they allow you to temporarily add new statements to the language, so that you can say things like PICK A NUMBER and the computer will understand it! BASIC is a relative old-timer among microcomputer languages, so it doesn't have all of the features that are being designed into programming languages today. In fact, many professionals go so far as to call it antiquated or obsolete.

🐾 If pseudocode doesn't suit your fancy, there are other ways of doing the same thing. The FLOWCHART is the traditional method; it graphically represents the flow of control in a program. But most experts today consider flowcharts to be old-fashioned and error-prone. An alternative that combines some of the advantages of flowcharts and pseudocode is the Warnier diagram. Warnier diagrams (pronounced worn-yay) takes some getting used to, but they're extremely powerful tools for organizing ideas. For more information on program design, see the book section of Appendix A.

BASIC is not likely to be confined to museums for a while—it has too much going for it. It's a fairly simple language to learn, at least at the introductory level. It's convenient to use, since it's built right into a tiny corner of your microcomputer (and almost everybody else's), and it translates your programs into machine language automatically when you run them. It's probably more widely used than any other microcomputer language in the world. So it's easy to find BASIC programs and programmers.

But BASIC does have some serious shortcomings as a programming language, which we'll consider in the next two sections of this chapter. Many of these problems are inherent in the language, and BASIC wouldn't be BASIC without them. But no computer language is cast in stone forever, and many of the shortcomings of BASIC can be overcome with the appropriate UTILITY PROGRAMS. You've seen one such utility for Commodore 64 BASIC in Chapter 4: the DOS Wedge, for making disk operations easier. Let's look at a few more.

Power to the programmer. If you're serious about BASIC programming on your 64, you'll like POWER 64, a utility package from Pro-Line Software. POWER 64 is a programmer's program. It can chop hours—or days—off the time it takes to write large programs, and make the remaining programming time more pleasant. When you run POWER 64 at the beginning of your work session, you add features and commands to the language that extend it in ways that any programmer can appreciate. Some examples:

- When you're looking at a big program listing on the screen, POWER 64 lets you scroll up as well as down, so you can easily study the whole program.
- When you're entering code, POWER's AUTO command creates and types the line numbers for you. If you make some additions to the program that require renumbering of existing lines, the RENUM command will take care of that detail for you. If you decide to delete a chunk of your program, DEL does it instantly.

- When you're typing in a program, POWER recognizes one-letter abbreviations for BASIC keywords. It also lets you define entire phrases or subroutines that can be evoked at the touch of a key.
- For editing your program, POWER has a search-and-replace feature similar to those found in the best word processors. It also allows you to easily merge two or more BASIC routines into a single program.
- When your program gives you an error message you don't understand, you can ask "WHY?" and POWER will answer you with a more detailed explanation. For additional debugging help, you can SINGLE-STEP through the program one step at a time), TRACE the changing values of variables, and DUMP the contents of key memory locations.
- **MOREPOWER**, a companion program to POWER, simplifies disk operations with commands like DISK (for sending commands to the drive) and UNDO (in case you accidentally NEW the wrong disk).

This is a professional package, complete with friendly but thorough documentation. Use it for a while, and you'll wonder how you got along without it!

(It can be purchased at a lower cost as part of a set with PAL, an assembly language package that takes advantage of some of POWER'S features. PAL is discussed later in this chapter.)

Sound and graphics. There's another problem that's pretty much unique to your Commodore BASIC: the tedium of programming with sound and graphics. Your computer hardware allows for some of the fanciest visual and auditory effects you can find in any home computer. But in order to achieve those effects in unadorned BASIC 2.0, you have to write all kinds of cryptic PEEK, POKE, and DATA statements. Wouldn't it be nice if you could use statements like CIRCLE and LINE to draw pictures on the screen? Or use equally simple commands to make musical notes?

You can, but not without a little help. These statements aren't built into the BASIC translator that comes with your computer, but it's possible to extend BASIC 2.0 with a Video Support Package (VSP) called the SUPER EXPANDER 64. This cartridge adds many sound and graphics statements to the translator, so you can use them in your BASIC programs to create dazzling effects without having to study the memory map for hours. If you're interested in sound and graphics programming in BASIC, take a look at this package.

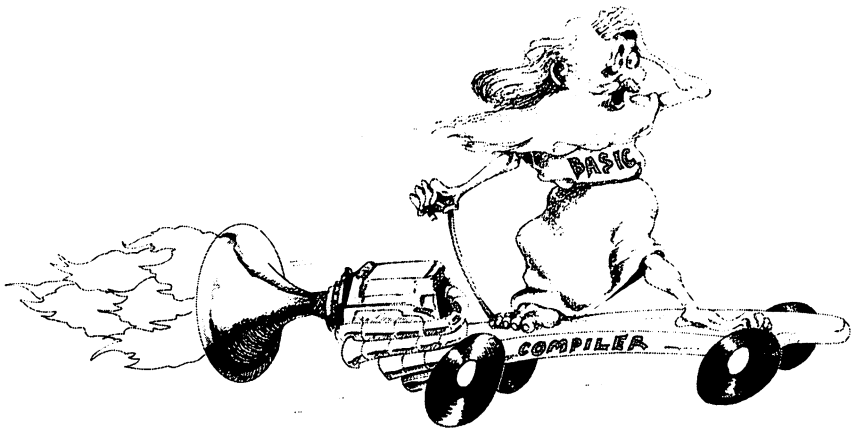


By the way, there are lots of graphics support packages on the market besides the one from Commodore. Many companies manufacture **SPRITE EDITORS**, which simplify the process of creating sprites in BASIC. Normally, sprite creation involves drawing a picture on specially-labeled graph paper and converting lots of binary numbers to octal. A sprite editor allows you to do the drawing on the screen, and it takes care of all of the arithmetic for you. A simple sprite editor is included in Commodore's Disk Bonus Pack, along with a **CHARACTER EDITOR**, for defining new character sets, graphic or otherwise. In addition, several magazines have published listings for sprite editors.

Screen editing. Another program utility that can make life easier if you're writing your own software for the 64 is the COMMODORE 64 SCREEN EDITOR. This program, like the SOUND and GRAPHICS PACKAGE, makes the built-in BASIC a little bit smarter by adding new statements to the translation dictionary. These commands are designed to make it easy to control input and output on the screen without having to rely so heavily on PEEK, POKE, and PRINT full of funny-looking graphics characters. The utility of this editor will be apparent to anybody who's tried to write programs that can be easily used by novices. These commands facilitate setting up the screen so that input can be effortlessly typed into specific areas, disabling the STOP key so the program can't accidentally be killed, and the like. Beginners beware: the COMMODORE SCREEN EDITOR, like many utility packages, is designed for use by experienced BASIC programmers.

SIMONS' BASIC. If you're intrigued by the BASIC possibilities of POWER, SUPER EXPANDER, and the SCREEN EDITOR, wait 'til you see SIMON'S BASIC! This amazing language extension cartridge, designed by British programmer David Simons before he was 16 years old, adds 114 new commands to BASIC 2.0—commands that extend the language in just about every imaginable way. There are powerful POWER-like commands for program editing (like AUTO, RENUMBER, TRACE, and OLD); DISK and DIR commands to replace the DOS Wedge and MOREPOWER; super graphics, sprite, music, and joystick commands to rival those of the SUPER EXPANDER; screen manipulation and keyboard control commands (including commands for dumping a screen display to the printer) that are much simpler to use than those of the SCREEN EDITOR; and Pascal-like structured programming extensions (like IF-THEN-ELSE and REPEAT-UNTIL) that make program design and debugging easier than you imagined they could be. All of these commands take up only 8K of memory, leaving plenty of room to write programs that use them. There's a well-written 120-page manual packed with SIMONS' BASIC to get you started. This is the BASIC that should come built into the Commodore 64. (Are you listening, Commodore?) If you're serious about programming, you'll want SIMONS' BASIC.

Speed. The BASIC programs you write are translated into machine language by a BASIC INTERPRETER. An interpreter translates each statement, one at a time, into a form the microprocessor can understand. This is handy because it allows you to make changes in a program and run it immediately, knowing that the new statements will be translated when the computer gets to them. But it also can be a problem, because translation while the program is running takes time and slows the program down. This is especially noticeable when it contains lots of loops, because each statement inside a loop must be translated every time the loop is executed. For most programs, a little bit of wasted time is no problem since the computer still works faster than a speeding pencil. But for video games, word processors and other programs that depend on instant feedback, a few milliseconds can seem like an eternity.



A significant amount of time can be saved by compiling the program before it is executed. A COMPILER is a program that translates the whole program at once into machine language (or a language that's much closer to machine language). The output from the compiler is a program that runs 10 to 40 times faster than the original BASIC program. The price you pay for this extra speed is the inconvenience of having to go through the extra translation step. Every time even a minor change is made in the program, it's necessary to recompile the whole program before you can run it.

Some languages, like Pascal, require that all programs be compiled, rather than interpreted. With BASIC, you have a choice, but only if you invest in a BASIC compiler. If you're addicted to BASIC, but you like speed too, then a BASIC compiler is a must. The most popular BASIC compiler for Commodore computers is Commodore's PETSPEED.

BEYOND BASIC

There are lots of things you can do to improve on the BASIC that comes with your machine. But no matter what you do to BASIC, it's still BASIC, and its structure is not without its problems. In his book *Mindstorms*, Seymour Papert, the inventor of Logo, compares BASIC to the inefficient QWERTY typewriter keyboard. Alfred Bork, expert on computers in education, goes so far as to call BASIC the junk food of programming languages! What, exactly, is the problem?

Computer languages, like other languages, are tools for communication; they allow us to communicate with machines. And like other languages, computer languages have different strengths and weaknesses as communication tools. 🐾 No one has yet invented the perfect programming language. No one is even sure what it means! But there is a general agreement among those who think about such things that a good programming language should make the process of communicating with the machine as easy as possible for humans. Each new generation of languages gets a little closer to this goal, usually by looking more like English and less like machine language. The idea, then, is to create a computer language that makes writing programs as similar as possible to the other forms of thought and communication that we're used to.

Using this as a yardstick, BASIC comes up short, especially in comparison with some of its younger siblings. While BASIC is easy to learn, and handy for writing quick-and-dirty little programs, it's not designed to conform to the shape of the natural human reasoning process. The simplicity of vocabulary

🐾 For example, Eskimo languages have massive vocabularies for communicating different characteristics of snow, but are not particularly noteworthy for discussing higher mathematics.

that makes BASIC so easy to learn (at least at first) is also the feature that makes it difficult to use for any but the simplest programs.

You may have noticed that it's difficult to write even a short program in BASIC without using at least one GOTO statement. At the time BASIC was born, that was no surprise; the GOTO statement was simply a way of representing the machine language jump and branch operations. But research in computer science since that time has shown that programming with the GOTO statement goes against the grain of human thought processes, resulting in programs that may be full of logic errors and difficult to understand. Every branch in a program represents another loose end the programmer has to remember to tie up and another invitation to forget something important. The bigger the program, the less negotiable the maze of branches becomes, until even the programmer doesn't have any idea what's going on.

One of the biggest problems with BASIC, then, is that the logical structure of a typical large program looks like a tangled spider web. BASIC lacks the CONTROL STRUCTURES that help in writing neat, easy-to-understand programs. Aside from GOSUB, IF . . . THEN, and FOR . . . NEXT, there are no real alternatives to using a GOTO to transfer control in the program. It's possible to write well-structured error-free programs with just these structures, but it requires considerable discipline on the part of the programmer.

A related problem is the difficulty of reading BASIC programs. Because the logic of most BASIC programs is so full of jumps, and because BASIC variable names are generally short and not very descriptive, and because BASIC programmers tend to cram statements together so tightly, it's very difficult for most humans to read and understand big BASIC programs.

Also, there's the lack of DATA STRUCTURES: ways of putting together individual pieces of data. In BASIC, you can use arrays and strings, and that's about it. If you want to store information on a group of people, so that each person's name, age, and birthdate are together, you have to fake it with three arrays.

It works, but once again, it requires you to bend your own thought processes to fit the requirements of the language.

All of this should not be taken as a blanket indictment of the language; BASIC has a lot going for it, as I've already stated. But it's not perfect, and it's not the only game in town. Whether you're a rookie or a veteran hacker, you should be aware of the options that are available to you, so that you can choose the right tool for the job.

THE POWER OF PASCAL

Pascal (named after Blaise Pascal, the 17th century mathematician, philosopher, inventor, scientist, and mystic) was invented by Swiss computer scientist Niklaus Wirth in the early seventies for the same reason that the folks at Dartmouth invented BASIC a decade earlier: He perceived a need for a language to teach beginning programmers the concepts of computer science. Wirth's Pascal, like BASIC, is relatively free of clutter: it has only 35 keywords. Many of the statements in Pascal are similar to BASIC statements, and the fundamental concepts of programming are the same in both languages. All in all, the two languages are similar enough that BASIC programmers usually have little trouble learning to program in Pascal.

But the two languages also have striking dissimilarities. Some are superficial: differences in the rules of syntax, for example. Others cut deeper, revealing the different philosophies designed into the two languages. Because of these, Pascal programmers and BASIC programmers tend to approach programming and problem solving in radically different ways. If you learn both languages, you learn two ways of thinking about problems as a bonus.

A typical BASIC program has a linear structure, resembling the network of electrical wires in a house. A Pascal program is structured in blocks, more like the rooms of a house. This kind of structure is a lot more like the way most of us naturally think, so programming in Pascal demands less of a compromise on our part.

To understand why this is so, consider again the game algorithm we developed at the beginning of this chapter. Our first cut at the problem was nothing more than a list of three things our program had to do:

```
-----  
BEGIN GAME.  
REPEAT TURN UNTIL NUMBER IS GUESSED OR  
SEVEN TURNS COMPLETED.  
END GAME.  
-----
```

We can think of each of those three statements as a **BLOCK**. In the beginning, we didn't know or care what was inside each block; we just had a general idea of what each should do. Then, through a series of refinements, we developed the details of each block until we had this:

```
-----  
BEGIN GAME:  
  GIVE INSTRUCTIONS.  
  PICK A NUMBER BETWEEN 1 AND 100.  
  SET COUNTER TO 0.  
  REPEAT TURN UNTIL NUMBER IS GUESSED OR  
  COUNTER = 7:  
    INPUT GUESS.  
    IF GUESS = NUMBER, THEN SAY SO AND  
    QUIT;  
    ELSE, IF GUESS < NUMBER, THEN SAY SO;  
    ELSE, SAY GUESS IS TOO BIG.  
    ADD 1 TO COUNTER.  
  END GAME:  
  PRINT MESSAGE.  
-----
```

Now, either one of these outlines can be easily turned into a Pascal program, as you'll see. For now, we'll use the second version as the basis for a program, since that's the one we used to build our BASIC program. (Later, we'll return to the original three-block outline, and see how that can be used to make a more MODULAR program.) Here's the program (as it would run on your computer with the KMMM Pascal compiler):

```
-----  
PROGRAM GAME (INPUT,OUTPUT);  
(* THIS PROGRAM PLAYS A NUMBER GUESSING  
GAME*)
```

```
VAR NUMBER, GUESS, COUNTER: INTEGER;
```

```
BEGIN  
WRITELN ('LET'S PLAY A GUESSING GAME. ');  
WRITELN ('I'LL PICK A NUMBER BETWEEN 1  
AND 100, ');  
WRITELN ('AND YOU TRY TO GUESS IT. ');  
WRITELN ('I'LL GIVE YOU SEVEN TRIES. ');  
NUMBER := 1 + TRUNC(RND(1)*100);  
COUNTER := 0;  
REPEAT  
  WRITELN ('WHAT'S YOUR GUESS?');  
  READLN (GUESS);  
  IF GUESS = NUMBER  
    THEN WRITELN ('YOU GOT IT!')  
    ELSE IF GUESS < NUMBER  
      THEN WRITELN ('TOO SMALL, GUESS  
      AGAIN.')    ELSE WRITELN ('TOO BIG, GUESS AGAIN. ');  
  COUNTER := COUNTER + 1;  
UNTIL (GUESS = NUMBER) OR (COUNTER = 7);  
IF GUESS <> NUMBER THEN  
  BEGIN  
    WRITELN ('I FOOLED YOU SEVEN TIMES!');  
    WRITELN ('THE NUMBER WAS',NUMBER:3);  
  END  
END.
```

You shouldn't have much trouble understanding most of this program if you compare it with algorithm and the BASIC version. But a few points are especially worth noting.

The first thing that stands out is how a Pascal program is constructed in three parts, like a recipe in a cookbook. First, the program heading, containing the name of the program and any files used (the name and description of the dish to be cooked).

Next, the declarations of variables, constants, data types (like arrays) and subprograms to be used in the program (the list of ingredients). Finally, the program statements, sandwiched between a BEGIN and an END (the cooking steps).

The reason all of the variables are declared before the statement part of the program is simple: Pascal requires that every word in a program (except keywords) must be defined or declared before it's used. That requirement allows the Pascal compiler to find many potentially dangerous typos and other bugs that could cause your program to spew incorrect answers or no answers at all. In BASIC, if you misspell a variable name, the interpreter will think you're creating another variable, causing it to give wrong answers or do something unpredictable. On the other hand, when the Pascal compiler finds a name it doesn't recognize in your program, it won't even run it until you make amends. This is an example of Pascal protecting the programmer from himself.

This kind of paternalism permeates Pascal. And, like cars that won't start until the seat belt is fastened, it's not popular with everybody. Pascal does everything it can to FORCE you to write well-structured, carefully crafted programs. No matter how you feel about it philosophically, this scheme seems to work. Programmers who learn Pascal first tend to write tighter, better organized code with fewer bugs, even when they use BASIC!

Back to the program. The text between these symbols (* . . . *) is a comment, similar to a BASIC REM. There are no line numbers in the program, because Pascal doesn't care too much about lines. Statements are separated from each other by semicolons, and they generally don't have labels. Statements can extend over several lines, and they may, in some cases have other statements inside them. (See below.) Indentation and blank lines are ignored by the compiler; Pascal programmers use them because they make programs easier for humans to read. The variables in this program are all declared to be INTEGERS, which means that they can only hold whole numbers. WRITELN (pronounced write-line) is the rough equivalent of PRINT.

This program contains two statements that have no equivalent in BASIC: REPEAT . . . UNTIL and IF . . . THEN . . . ELSE. The REPEAT . . . UNTIL statement does just what it says: it causes all of the statements between REPEAT and UNTIL to be repeated until the UNTIL condition is satisfied. IF . . . THEN . . . ELSE is like BASIC's IF . . . THEN, except that it allows you to tell the computer what to do if the condition ISN'T true; the ELSE part is optional. Notice how a BEGIN and an END are used in the last IF . . . THEN statement to specify that two things are to be done if COUNTER is not equal to 7: This kind of grouping of statements into BEGIN . . . END blocks, when used with control statements like IF . . . THEN . . . ELSE, makes short work of many programming tasks that would be very messy in BASIC.

Speaking of blocks, let's look at this same program written as a series of subprograms, or PROCEDURES:

```
-----  
PROGRAM GAME (INPUT,OUTPUT);  
(* THIS PROGRAM PLAYS A NUMBER GUESSING  
GAME *)
```

```
-----  
VAR NUMBER, GUESS, COUNTER: INTEGER;
```

```
-----  
PROCEDURE STARTGAME;  
BEGIN  
  WRITELN ('LET'S PLAY A GUESSING GAME.');
```

WRITELN ('I'LL PICK A NUMBER BETWEEN 1
AND 100,');

WRITELN ('AND YOU TRY TO GUESS IT.');

WRITELN ('I'LL GIVE YOU SEVEN TRIES.');

NUMBER := 1 + TRUNC(RND(1)*100);

COUNTER := 0;

END;

```
-----
```

```

PROCEDURE TURN;
BEGIN;
WRITELN ('WHAT'S YOUR GUESS?');
READLN (GUESS);
IF GUESS = NUMBER
  THEN WRITELN ('YOU GOT IT!')
  ELSE IF GUESS < NUMBER
    THEN WRITELN ('TOO SMALL, GUESS AGAIN.')
```

```

  ELSE WRITELN ('TOO BIG, GUESS AGAIN,');
COUNTER := COUNTER + 1;
END;
```

```

PROCEDURE ENDGAME;
BEGIN
IF GUESS <> NUMBER THEN
  BEGIN
  WRITELN ('I FOOLED YOU SEVEN TIMES!');
  WRITELN ('THE NUMBER WAS',NUMBER:3);
  END;
END;
```

```

BEGIN (* MAIN PROGRAM *)
STARTGAME;
REPEAT
  TURN
UNTIL (GUESS = NUMBER) OR (COUNTER = 7);
ENDGAME;
END.
```

Pascal procedures look like little Pascal programs, but they work like BASIC subroutines. Well, not exactly: They're more flexible than their BASIC counterparts, you call them by name instead of line number, and they don't explicitly say RETURN. But the idea is the same: When a procedure is called by name in the main program, the computer jumps to the beginning of that procedure, works its way through to the end of the procedure, and returns to the place it left off in the main program. Like everything else in Pascal, procedures must be declared before the main program. Notice how you can get a quick summary of what the program is doing by reading the short main program, which looks amazingly like our original algorithm.

Writing programs with procedures like this is almost like adding your own custom statements to the language!

There are lots of things that these examples haven't shown you about Pascal: How easy it is to make up brand new data types and functions if you aren't happy with the ones that come with the language, or how you can build complex data structures with relative ease. (By the way, there is a GOTO statement in Pascal, but most hardcore programmers consider it to be just another four-letter word.) Pascal packs a lot of power into a small package. But like BASIC, the standard Pascal that Wirth developed is showing its age. Consequently, most microcomputer versions of Pascal are modernized versions of the language.

● **UCSD Pascal.** The most popular of these is UCSD Pascal, named for the University of California at San Diego, where it was developed. Commodore promised a C-64 version of UCSD Pascal when the machine was first announced, but it's still not available at this writing. The UCSD package will include not just a Pascal compiler, but a whole new OPERATING SYSTEM, called the P-system, which makes your computer seem to have a whole different personality. (Or is it a machinality?) Using the P-system, you can create programs with a powerful screen editor (similar in many ways to a good word processor) instead of the simple but limited line-numbering BASIC editor. You can use a different (more understandable) set of commands to talk to the disk drive and printer. This operating system is the same on any computer with UCSD Pascal, no matter what brand, so you can easily convert programs written in Pascal (or other languages) for other computers to run on yours, and just as easily convert yourself so you can write Pascal programs on other computers!

● **KMMM Pascal.** Another Pascal compiler from Wilserve Industries is available for the Commodore 64. This compiler doesn't come equipped with all of the UCSD features, it isn't quite as easy to use, and the current documentation is spotty and definitely not for novices. On the other hand, KMMM Pascal produces programs that run faster than UCSD, because

it translates them into machine language rather than an intermediate language. In addition, KMMM offers excellent customer support. And most important, KMMM Pascal for the 64 is a reality, not a promise.

TWO PRACTICAL REASONS FOR CONSIDERING PASCAL:

1. For students: More schools every year (especially universities) are teaching Pascal in their introductory programming classes because it encourages the development of good structured programming techniques. In fact, the U.S. College Entrance Examination Board is offering advanced placement (AP) college credit to college freshmen who can pass an AP test on Pascal.

2. For job-seekers: Pascal is becoming increasingly popular as a programming language in the real world, and there are an ever-growing number of high-paying jobs for Pascal programmers. The U.S. Department of Defense, which always has plenty of jobs, is now moving to a Pascal-like language called ADA as its standard programming language.

All of this is not to suggest that Pascal is perfect. Many programmers think that its imposed discipline hampers the creative process. Few will deny that the extra step of compiling a program every time you make a change can be bothersome. Even its creator, Niklaus Wirth, recently designed a replacement for Pascal. (This new language, called Modula 2, maintains the spirit of Pascal while correcting some of the most often criticized features, extending its power, and incorporating some of the most recent advances in computer science. It's not yet widely available.)

But all things considered, Pascal is a powerful, relatively fast programming language that makes structured program design come easy. It's especially well-suited for designing large, complex programs where reliability is important.

THE COMING OF COMAL

If the debate between the BASIC and the Pascal camps has you hopelessly confused, there's a new language coming onto the American software scene that just might be for you. Danish Professor Borge Christensen developed Comal because he was not happy with BASIC or Pascal as introductory programming languages. COMAL enthusiasts claim that it has the best features of both languages, plus a few bonuses of its own. Because of these strengths, COMAL is now the official programming language in the schools of Denmark and Ireland, and Sweden and England may soon follow suit. A detailed look at this interesting new language would be technical and lengthy, but a quick survey of the high points should give you enough information to know whether you want to look further.

COMAL has many of the powerful statements and constructs that make structured programming easy in Pascal: IF. . . THEN. . . ELSE, REPEAT. . . UNTIL, PROCEDUREs, and so on. COMAL also shares Pascal's penchant for readability, by allowing long variable names and indented code. In this respect, COMAL goes Pascal one better by AUTOMATICALLY indenting listings to highlight the program's structure! COMAL makes file handling, printer access, machine language interfacing, and putting several program together much easier than they are in BASIC.

But COMAL isn't nearly as picky as Pascal about details of punctuation and such. It generally points out typos as soon as they're typed; in some cases, it corrects them for you! BASIC programmers who are intimidated by the idea of having to learn to use a new Pascal operating system will be reassured to know that COMAL commands are pretty much the same as the familiar BASIC bunch. And there's no need to learn a new style of editing, either: COMAL allows you to use line numbers when creating and changing programs. (But COMAL will AUTOMATICALLY assign line numbers as you build your program, and it allows you to renumber everything with a simple command.)

There is one very big strike against COMAL when compared with either BASIC or Pascal: its relative lack of popularity. There aren't a lot of COMAL programmers running around in the Western Hemisphere, so it's difficult to find a support group when you need help debugging. Since COMAL compilers are only available on a few computers COMAL programs are not very portable. If COMAL Users Group have their way, COMAL will be the next big thing in schools everywhere. The price is certainly right: COMAL is a public domain language.

ASSEMBLY LANGUAGE: BIT-TWIDDLER'S DELIGHT

In the earliest days of computers, every program had to be written in MACHINE LANGUAGE—the machine's built-in set of instructions. Writing error-free programs in machine language is difficult and slow. It requires a working knowledge of the idiosyncracies of the machine, careful attention to detail, and unnatural patience. HIGH-LEVEL LANGUAGES like BASIC, Pascal, and LOGO were developed to make life easier for programmers by hiding the inner workings of the computer and streamlining the programming process.

Why, then, would any modern programmer voluntarily write programs in machine language? While a few brave souls do it for the intellectual challenge or philosophical satisfaction, most have more practical reasons:

1. **Speed.** Programs written in the machine's native language run faster than programs written in other languages; usually MUCH faster.
2. **Space.** Programs written in machine language can be packed into much smaller areas of memory than high-level programs.
3. **Flexibility.** Some programming jobs are impossible to do in a particular programming language because of limitations of the language. Not so with machine language—if the computer is capable of doing something, it's possible to write a machine language program that tells it how. It may not be easy, but at least it's possible.

So when programs need quick reflexes, when memory is in short supply, or when esoteric features of the machine need to be tapped, programmers often resort to machine language programming, sort of. In truth, most machine-language programmers use ASSEMBLY LANGUAGE instead. Assembly language is similar to machine language, in that each instruction generally represents exactly one of the machine's primitive operations, such as "load a value into the accumulator." But in assembly language, the instructions are written in the form of MNEMONIC CODES rather than numbers. To illustrate, here's what the instruction for loading the number 1 into the accumulator looks like as a string of bits in the computer:

1010100100000001



That's the REAL machine language. It's a little more readable if you translate it from binary notation to hexadecimal (base-16):

A9 01

But the meaning still isn't obvious to anybody except the machine. The assembly language version looks like this:

:LDA #1

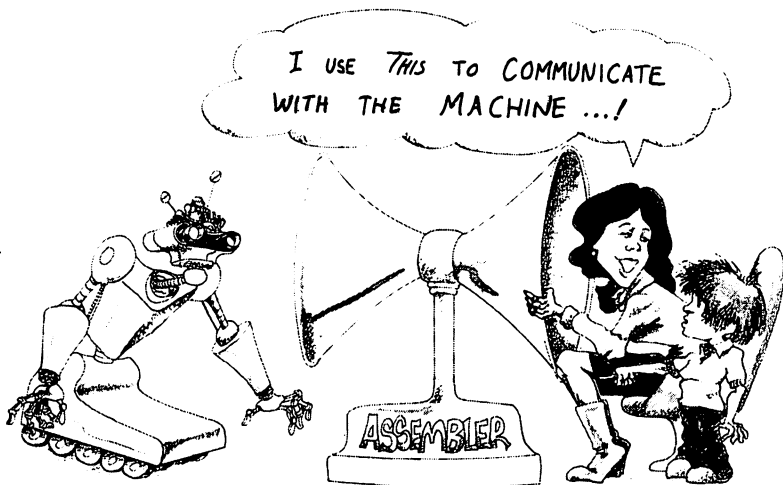
LDA is the mnemonic code for "load the accumulator," 1 means 1, and the pound sign says the 1 is a value rather than a memory address. An ASSEMBLER is a program that translates a program made up of statements like this into machine language.

If this still looks primitive and unfriendly compared to BASIC or Pascal, that's because it is. Assembly language is not for the casual computerist. But if you want to make your fortune writing lightning-fast video games, or if you want to really understand what makes your 64 tick, it may be for you.

You'll need three things to get started in machine language. First, get one of the many books about writing programs for the 6502 microprocessor. The Commodore 64 uses the 6510 chip, but these two chips use the same instruction set. Second, if you don't already have it, invest in the *Commodore 64 Programmer's Reference Guide*. This is the book that fills in all of the details of your particular machine. Finally, you'll need an assembler software package.

There are several available packages from different manufacturers, each with different strengths and weaknesses. One of the best all-around assembly-language packages on the market is PAL (Personal Assembly Language) from Pro-Line Software. One convenient feature of this one is that it can be used in conjunction with POWER, discussed earlier in the chapter. If you can't afford the PAL/POWER package, Commodore sells an excellent ASSEMBLER DEVELOPMENT SYSTEM at a bargain price. If your bank account can't even handle that,

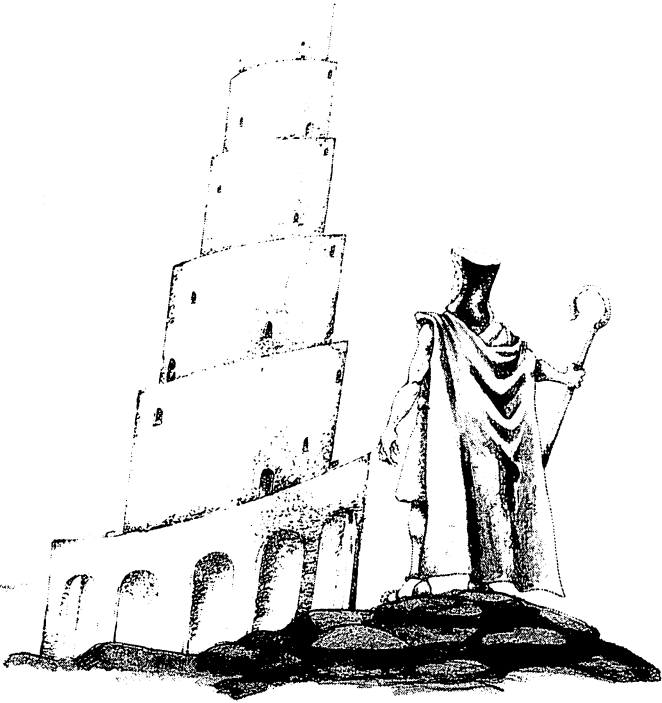
Commodore Disk Bonus Packs and Cassette Bonus Packs contain a good entry-level assembly-language monitor called SUPERMON 64.v1. If you don't have a bank account at all, borrow the January '83 issue of *Compute!* magazine—it has a type-it-yourself listing of SUPERMON 64.



THE TOWER OF BABEL: FORTH, PILOT, AND MORE

The Forth Dimension. If the speed and economy of assembly language appeal to you, but you're also drawn to the idea of the structured program design, you may be ready to go Forth. Forth is a unique figure among programming languages: it's like a cross between Logo and assembly language. Forth is a complete operating system, including an editor, that fits into a tiny part of the computer's memory. That leaves more room for your Forth program, which probably won't take up too much space, either, because Forth code is very compact. Forth programs generally run faster than programs written in other languages, but not quite as fast as machine language programs. Forth offers many of the conveniences of the "structured" languages like Logo: constructs like IF-THEN-ELSE and the ability

to define your own new statements. As a bonus, Forth is interactive, like BASIC and Logo—it carries out your commands as soon as you enter them. It's no wonder that Forth is widely used for programming video games, spreadsheet programs, compilers for other languages, even robot control programs!



But regardless of what its advocates say, Forth isn't for the casual programmer. It's a demanding language, requiring careful attention to detail. When you program in Forth, you often have to do things that other languages take care of automatically—like write your own routines to handle strings, fractional arithmetic, and complex input and output. Once you've written these routines, you can use them in any Forth programs you write. But some of us don't even want to have to write them once.

Another thing that might keep you from starting Forth is the requirement that all arithmetic be done in reverse Polish notation, or postfix. In postfix, the operators (like plus signs) come after, rather than between, the numbers they're operating on. So "3 + 2" in postfix is "3 2 +", and "(5 + 2)/3" is "5 2 + 3 /". Once you adjust to it, this is no more difficult than the normal way. Many people (and all computers) find it easier, because it eliminates the need for parentheses in complex expressions. But it does take some getting used to, and it's just another thing to have to learn.

Forth is, at the very least, a good language for good programmers who need to write fast, compact, structured code. There are Forth fanatics who argue that it is the closest thing we have yet to a perfect programming language. If you want to explore the brave new world of Forth, you can contact the Forth Interest Group (FIG), listed in the periodicals section of Appendix A, and/or play with one of the Commodore 64 Forth packages. Forth implementations for the C-64 are available from Human Engineered Software (HES) and Handik Software, and (soon) Commodore.

Pilot: A Tool for Teaching. Pilot is a language with a purpose: it was designed as a tool for teachers to create computerized study aids—drill and practice lessons, multiple choice tests, and the like. It's kind of like a simplified form of BASIC, so it's easy to learn. Because the commands are single letters (such as T, for Type—the rough equivalent of BASIC's PRINT) it's possible to create lengthy lessons with a minimum of typing. (The trade-off is that Pilot programs are hard to read until you memorize the command abbreviations.)

● **VANILLA PILOT**, from Tamarack Software (distributed by Computer Marketing Services), is an especially easy-to-learn version of Pilot at an affordable price. In addition to the text-manipulation commands of standard Pilot, Vanilla Pilot has LOGO-like turtle graphics commands that allow you to make color line-drawings on the screen. In addition, it has commands for joystick control, you can write programs that can be used by young learners who aren't keyboard-literate.

Commodore's Pilot for the 64 is a more expensive, more comprehensive version of the language. The graphics are based on Cartesian, rather than turtle geometry. For example, this series of commands erases the screen and draws a box:

```
G:E;P50,50;D150,50,D150,100;D50,50
```

In addition to commands like these for making drawings, Commodore Pilot has commands for easily creating and moving sprites around the screen in animated displays. (But alas, no joystick commands.) Other commands simplify the music and sound synthesis operations.

Both versions of Pilot come with self-teaching manuals, so you shouldn't need a separate book to learn the language. (The Vanilla Pilot manual is probably simpler for beginners, if for no other reason than the language it talks about is simpler.) For business programming or video games, Pilot is a poor language choice. But if you're interested in writing educational programs, Pilot may be for you.

LOGO revisited: not for children only. Chapter 6 introduced LOGO as a language for learning. Many enthusiasts see LOGO as the multipurpose language of the future. It's got many strengths: it's easy to learn, easy to use, powerful, extensible (you can add new commands to it), and popular. Efficiency is NOT one of LOGO's selling points. Like BASIC and other interpreted languages, it runs slowly. And the language takes a lion's share of your computer's memory, leaving precious little room for big programs. But when speed and size aren't major concerns, LOGO is hard to beat.

THE LAST ONE: Beyond programming languages. Finally, there's a new kind of tool emerging for people who want to write their own custom software without bothering with the messy details of programming. Called PROGRAM GENERATORS, these software tools are really programs that write programs. THE LAST ONE is a program generator that's available for most microcomputers, including the Commodore 64. (The pretentious name is based on the questionable assumption that it's the last piece of software you'll ever buy.)

THE LAST ONE is a big and powerful program that really can write stand-alone BASIC programs, but it can't do it all by itself. You'll need to figure out *exactly* what you want it to do, just as you would if you were working with a human programmer. Then you'll need to tell the program-programmer by picking from alternatives in numerous program menus. These menus allow you to methodically create a "flowchart" outlining the basic algorithm. Then TLO asks questions about each step of your flowchart until it has all the details it needs (screen design, branch destinations, file details, etc.) to build a program. What if you decide to change the program later? Although it's possible to change the BASIC program that you've created, it's easier to prompt TLO to change the flowchart and generate a whole new program. Of course, that means another round of menus and questions.

Responding to all of these menus and nit-picky questions can be tedious—it may take several hours of interacting with TLO to satisfy its appetite for information on a large programming project. It pretty much forces you to systematically design your program; hacking just won't work. TLO also requires that you have a rudimentary understanding of the logical flow of programs and the nature of data files. But this material is explained in the manual in an easy-to-digest form. The manual contains a thorough tutorial that takes you step-by-step through the program-design process with a minimum of technical jargon (and just a few errors).

If you're considering this program or one like it, look over the documentation and see if it speaks to you. If possible, work with the tutorial to get a feel for what the program does. Ask yourself if you're willing to go through this process to create custom programs. THE LAST ONE Might be a valuable time-and-money-saver for you if (a) you can use lots of file-oriented programs like mailing lists, business report generators, or educational drills; (b) you have, or are willing to acquire, a basic understanding of file manipulation and program flow; and (c) you don't mind doing some of the work yourself, but you don't want to get bogged down in the details of writing programs.

Languages, Languages, and more Languages.

There are hundreds of other programming languages. Most of these aren't available directly on the Commodore 64. (You probably wouldn't want most of them on the C-64 even if they were.) But if you didn't find what you were looking for here, keep reading. The next chapter will discuss CP/M and other tools that can open up the door to all kinds of other languages for you and your computer.

CHAPTER 9

The Imagination Machine: Today and Tomorrow

Tomorrow never knows . . .

—The Beatles

If you've stayed with me all the way through this book, you've covered a lot of ground. You've learned how to get your computer up and running, and how to equip it with the necessary hardware attachments. You've learned how to use your computer as a tool, a teacher, and a toy. You've seen the power of software to magically transform your blank TV screen into a financial worksheet, a textbook page, a spaceship porthole, or a programmer's toy.

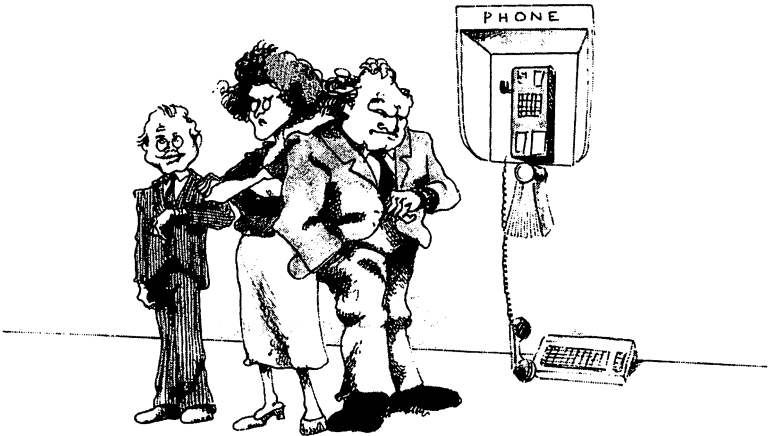
If you believe that software is the key to the computer's versatility, you're right. But there's more! Your amazingly malleable machine can be transformed in even more ways with a few extra pieces of hardware. If you start feeling confined by the walls of your little 8-bit computer, you can attach a modem and use your C-64 to talk to other computers and computer users all over the country. If you're just getting a little bored with the Commodore way of looking at things, you can plug a new software or hardware brain into the box so that you temporarily have a whole new computer! Feeling creative? You can hook up your computer to a keyboard and make music, or attach a plotter to make your computer draw the pictures you see in your mind, or connect a speech synthesizer to your computer so it can recite poetry to you! Or if you're sick of the whole thing and want to go where there are no machines for a couple of weeks, you can teach your computer to turn the lights off and on while you're gone so that it will still be there when you get back.

It's only fair to warn you that some of these things haven't made it to the marketplace as I write these words; they exist only as prototypes and promises from Commodore and other

companies. In the competitive world of microcomputer marketing, promises are broken almost as often as they are kept, and many of the most interesting products are totally unexpected. But we won't let that keep us from peeking cautiously into your computer's future.

THE MODEM: A WINDOW INTO THE WIRED WORLD

Many experts believe that the real future of your home computer is rooted in its ability to reach beyond the walls of your home. From this point of view, the computer is more like a telephone or a television than it is a stereo or a calculator. Whatever the future holds, there's little doubt that our communication patterns are changing. With our massive network of phone lines, TV cables, and communication satellites, we've created an incredibly complex planetary nervous system. This TELECOMMUNICATION system gives us almost instant access to information from all over the world, through our telephones, radios, televisions, and now, our computers. All of these devices are capable of bringing electrically transmitted information into our homes, each in a different form. We're all at least vaguely aware of the ways TV and telephones affect our communication patterns; computers are something else again.



The modem. By itself your computer can't communicate with anything but you. To make it talk to the world, you have to hook it to the phone line that runs into your house. But the phone line was designed to carry ANALOG (continuous) voice signals, not a DIGITAL stream-of-bits generated by a computer. This minor problem can be solved with a modem—a hardware device that stands between your computer and the phone line, translating digital to analog and vice-versa.

There are a lot of modems on the market; they vary considerably in features, design, and price. Modems come in two varieties: acoustic modems and direct-connect modems. Acoustic modems have a cradle that holds the handset of the phone; they translate the computer's signals into beeps that the phone receiver can hear, and pick up the noises sent across the wires with a microphone before translating them into bits again. Direct-connect modems bypass the handset altogether, sending their signals directly into the handset cord or the phone jack, which is directly connected to the modem. This arrangement is quieter and more reliable than the acoustic method, but it requires a modularized phone and/or phone jack.

Modems also differ in their speed of transmission. Most inexpensive modems for home computers have a transmission rate of 300 BPS (bits per second); more expensive modems using a different technique have bit rates of 1200. To give you a perspective, 300 BPS is considerably faster than you can type, and somewhere near the speed that most of us read comfortably. But it's nowhere near as fast as your computer can display information, and it can be annoyingly slow if you're in a hurry. If you're using your modem for professional, time-is-money, purposes, you'll probably want to go for a big baud rate. The rest of us can get by just fine with an inexpensive 300 BPS modem.

There are lots of other features that distinguish modems, such as auto-dial, which allows the modem to automatically dial any phone number you type in on the keyboard; memory, which allows the modem to store phone numbers for future reference; and auto-answer, which allows your computer to ANSWER the phone and talk to other computers WITHOUT YOUR HELP!

Each of these features has its place—and its price. Unless you're firmly committed to communicating by computer, to the tune of several hundred dollars (including the special interface you're likely to need to make the modem fit your C-64), you'll probably want to start with one of the inexpensive modems designed to hook directly into your machine. A likely candidate is Commodore's 1600 Mode (VICMODEM); it's easy to use with the C-64, and it doesn't cost a bundle. Another inexpensive C-64 modem with similar features is HESMODEM, made by HES. If you want to pay a little more to auto-dial or auto-answer, ask for Commodore's model 1650.

Terminal software. It takes more than a modem to make your computer communicate with the world. Your 64 is a full-fledged computer, prepared to do your bidding when you turn it on. You have to fool it into thinking it's just a lowly computer **TERMINAL** before you can use it as one, and that takes software. This kind of program, is called either a **TERMINAL PROGRAM** or (confusingly) a **MODEM PROGRAM**. A good one takes care of most or all of the nitty-gritty details of establishing communication with other machines. (And there are *many* details!) A really good one will allow your computer to serve both as a terminal and a computer, so that you can (1) **DOWNLOAD** programs from other computers directly into your computer's memory, (2) **UPLOAD** programs from your computer's memory to other computers via the modem/phone connection, or (3) have the communication printed on paper in addition to being displayed on the screen.

Many modems, including the Commodore modems and Hesmodem, are sold with a terminal program as part of the package. If you don't like the one that comes with your modem, you might be able to find a terminal program in a public-domain collection. Or for something extra, consider one of the commercial terminal programs that are sold separately from modem hardware.

For example, three excellent modem programs are available from Midwest Micro Associates. All three of these programs offer the luxury of **SMOOTH SCROLLING**, so that the text moves smoothly up the screen instead of jerking from line to

line. The least expensive, '64 TERMINAL (which happens to be the program that is supplied with Hesmodem), comes on cassette. This machine language program lets you temporarily store received information in a BUFFER so that you can review it on the screen or print it later. (This can save connect-time and long-distance phone line charges.) The accompanying documentation is excellent. '64 TERMINAL PLUS comes on a disk, and adds uploading, downloading, and disk commands to '64 Terminal. SUPERTERM, the top of the line, is an expensive do-it-all terminal program for people with professional communication needs. Among other things, it allows your computer to disguise itself as any of 42 popular terminals, an important feature if you'll be communicating with computers with obscure protocols.

Another noteworthy modem program for the 64, COMMANDER ULTRA TERMINAL-64, is available in a convenient cartridge format from Creative Equipment. This well-documented program doesn't have smooth scrolling, but it allows you to select a screen color combination that's easy on your eyes. It also continually displays the time elapsed since the program was started, a useful feature when you're using a pay-by-the-minute utility. The Commander program has options for routing text to the printer and/or a disk at the same time it's being displayed on the screen. Text saved on disks can be reviewed on the screen or printed after you've disconnected from the remote computer.

Once you've got your modem hardware and software, what can you use them for? There are plenty of possibilities; let's look at a few.

THE STRANGE WORLD OF INFORMATION UTILITIES

Probably the most popular use for modems outside of the workplace is to provide access to COMPUSERVE, THE SOURCE, DELPHI, DOW JONES, and/or other commercial information utilities. These utilities are nothing more than large computers available for your use on a subscription and/or pay-by-the-hour

basis. When you're ON-LINE with (connected to) one of these computers, you have access to computing power that your micro never dreamed of, and databases that would fill your bathtub with diskettes. Most of these utilities are relatively new, and still searching for a solid customer base. The services offered are amazingly diverse, and growing and changing all of the time. Here's a sampling:

News. Many of these utilities offer up-to-the-minute news stories, straight from the wire services. Why?

TV and radio provide a degree of immediacy by presenting the most important news stories as they happen. But a news director decides which stories are important, and his priorities might be different than yours. Your only real choice is to turn the station off. With newspapers and magazines, you can skip the parts that don't interest you, and get more detailed coverage of those topics that are important to you. But searching through a newspaper for articles on a particular topic is an inefficient and error-prone process. And printed news is almost always old news.

News by computer can be more up-to-date than TV or radio broadcasts, and you don't have to sit through endless commentaries on the latest Washington cocktail party gossip unless you want to. After browsing through the headlines, you can choose to get more details on the stories that interest you. With some news services, you can even ask for all stories on a particular topic, and receive a list of headlines tailored to your exact needs. It's like having a constantly-revised personalized newspaper with an index!

This kind of thing isn't for everybody; we don't all have the need to know it all right now. But for many people this can be a powerful tool for their work, and for others it's a near necessity. I met a Presbyterian minister who used the Delphi news service to keep abreast of the latest happenings at a critical meeting of his church's general assembly that he couldn't attend in person; he simply asked about all stories indexed under the word "Presbyterian." I also met a visually handicapped student who uses that same news service as his main source

of national news. His computer is equipped with a speech synthesizer, so every word that appears on the screen is "spoken" by the computer.

Business and economic information. Most of these utilities offer current news on business, finance, and economics. The Dow Jones News/Retrieval service specializes in this kind of thing, offering Wall Street Journal Highlights online, current-to-the-hour stock quotations, and investment services, in addition to standard news service.

Reviews. Many of the subscription services offer reviews of current movies, plays, and television programs, so that you can ask that the critics say before you spend your time and money forming your own opinion.

Research. If you're in pursuit of more serious knowledge, you can use an on-line encyclopedia to look up Gandhi or gonorrhoea. And if you misspell the word you're looking for, you're likely to get help anyway. Here's what the Kussmaul Encyclopedia on Delphi would say if we asked for "gonorrhia":

```
-----  
Unrecognized entry /gonorrhia/ - scanning . . .  
GONNORRHIA  
GONNORRHI  
GONNORRH  
Do you want GONNORRHEA?  
-----
```

Some utilities also offer access to data bases related to specific fields of inquiry, from Aquaculture to World Aluminum Abstracts. It's possible to track down very specific and obscure pieces of information in minutes with one of these, if you know how to ask the right question. The largest information retrieval utility, DIALOG, has over 150 data bases covering every major subject. Knowledge Index is a subject of Dialog designed for low-cost access by personal computer users on evenings and weekends. BSR AFTER DARK is a similar service.

Finally, it's possible to get human answers to your questions through utilities with features like Delphi's Oracle. You just

type in your question, and somebody on the other end does her best to come up with a solid answer within a few days. The answer comes to you via electronic mail (explained below).

Shopping. Many of the utilities offer services like Compu-store, which allows you to search a catalog/data base for just about any material possession that money will legally buy. You can browse until you find something you want, order it, and pay for it automatically via credit card. Similar services for travellers let you peruse airline schedules and reserve your tickets by computer.

Banking. Pay your bills in the comfort of your own computer room, without licking any stamps. The computer asks you how much to pay to each of your accounts and takes care of the details for you. No cash, please.

Games. Of course, there are games, ranging from simple guessing games for younger children to complex adventure games. Video games are out, because color graphics and animation haven't been incorporated into these systems. 🐾 But these systems do offer a kind of multiplayer game that you can't play on your home computer alone. In CompuServe's MEGAWARS, for example, you join a perpetual interstellar battle. Megawars is like a souped-up version of the classic Star Trek game (see Chapter 6), except that the other space ships, good and bad, are controlled, not by the computer, but by other players sitting in front of computer screens somewhere! The action is fast and furious, and the rules are complex, so experienced players tend to fare better than rookies.

Bulletin boards. These are just what you'd expect—places to leave messages, places to read messages left by others. A plea for advice on buying a word processor; a help request from a weary dragon slaying adventurer looking for a key to the dungeon; and of course, the personals—they're all here. (At least one marriage has resulted from a meeting at the Source!)

🐾 There are exceptions; see the next section.

Electronic mail. Like bulletin boards, except more private. You can type in a message to any user of the system, and deposit it in her mailbox. The next time she logs on to the system, she's greeted with a message that there's a piece of mail waiting for her. She can read it at her leisure, and respond to it, destroy it, or save it for future reference. If you don't know who to write to, you can search the user data base for people with similar interests, proximity, or whatever. Electronic mail is a lot faster than any other form of mail, but not as demanding or immediate as the telephone. Some think that these systems mark the beginning of the end of the post office. Does that mean electronic junk mail?

Electronic conferences. Earlier I mentioned a couple of people I met who use Delphi's news retrieval service. I "met" these two people in a conference on Delphi. I was snooping around one of the bulletin boards when my typing was interrupted by the message, Collie would like to talk to you. After I indicated that that was OK with me, I found myself conversing by keyboard with somebody 2000 miles away. Before long, we were joined by somebody else at least that far away from both of us. We "talked" about Delphi, Compuserve, the Source, our homes, this book, and other things, without seeing a face or hearing a voice (except for Howard, who heard the voice of his Votrax speech synthesizer quoting us). And yet, after an hour, I felt like I'd made a couple of new friends. It's not the same as sharing pie and ice cream with the neighbors, but it offers a lot more human involvement than watching *Dallas*.

Miscellaneous get togethers. The Source has something called Parti, that's sort of a free-for-all cross between electronic conferencing and electronic mail. Compuserve has a CB simulation, where you can eavesdrop or participate in scintillating exchanges like this, from channel 34:

The bottom line. These services don't come cheap. Most information utilities charge an initial membership fee; all of them charge for use by the hour, typically much less for evenings and weekends. Many have additional charges for special services like data base research, banking, and personal file storage. The good news is that if you're in the urban 80 percent of the population, you can probably connect to most of these services by dialing a local phone number. And if you're lucky, you'll get a free temporary membership in one or more of these services when you buy your modem.

If you're curious about this kind of thing, I recommend you take advantage of any "first hour free" offers that you can. An hour isn't enough to explore any of these systems thoroughly. In fact, you may spend most of your time orienting yourself to the logical geography of the system. Compuserve, the Source, and Dow Jones—the biggies—can be especially intimidating for beginners because of their size and diversity of features. And with maybe 100 other users TIMESHARING on the system with you, you sometimes find yourself waiting several seconds for the computer to respond to your requests. But an hour on line should give you a feel for the service, and, whatever you decide, provide a fascinating experience. (See Appendix A for details on contacting any of these utilities, as well as the services listed in the next section.)

MORE TIMESHARING AND NETWORKING

Services like the Source and Compuserve are interesting, fun, and occasionally very useful. They're heralded by many as a giant step into the future, when most of our shopping, banking, research, and recreation will be done on-line. If that's true, we'll have to do some hard thinking about the old issues of privacy and security in a new context. The vendors of these services go to great pains to protect the privacy and integrity of their customers. But the potential for abuse of this new technology definitely exists, and that's enough to deter lots of people from giving up cash and joining the on-line generation.

If concerns about privacy, poverty, or apathy keep you from hooking up with the big-time on-line utilities, you still may be able to use a modem to your advantage.

Informal networks. Not all timesharing services are big commercial ventures. The ELECTRONIC INFORMATION EXCHANGE SYSTEM (EIES), operating out of the New Jersey Institute of Technology, is a small but important information network that has no financial ties to Reader's Digest (owners of the Source). More than 1,200 geographically- and intellectually-diverse users tap into this network to exchange information and ideas through electronic mail and various ongoing conferences. The staff of the *Whole Earth Software Review* rely heavily on EIES as an information resource.

Game lines. Multiplayer games are one of the most popular features of the big utilities. It was inevitable, then, that a special on-line gaming house would happen. The pioneer here is Chicago's GAMEMASTER, offering a variety of games of competition between man and machine or between man and man. (Sometimes it's difficult to tell one from the other.)

Specialized bulletin boards. If the idea of on-line bulletin boards intrigues you, but not enough to pay for the Source or CompuServe, you have alternatives. All over the country small, special-interest bulletin boards offer their services for little or no charge. Some are operated by computer user groups, others by special interest organizations, still others by businesses. Even in my little town of 40,000 there are two on-line services run by local computer stores, providing facts on products and programs, and serving as information clearinghouses. Some other examples (access information is in Appendix A):

- **CLEO**, an on-line directory of computer-related jobs, free to job hunters.
- **IF**, a free computerized magazine from the Imagination Factory of Anaheim, CA, specializing in video-related subjects.
- **PEACENET**, a project of the Disarmament Resource Center in San Francisco, provides information on upcoming events, legislative updates, and other activities related to the peace movement.

- **Commodore User Group bulletin boards**, located in various basements and garages around the continent. (The October, 1982, issue of *Commodore Magazine* has some tips on starting your own.)
- **CP/M User Group bulletin boards**, described later in the chapter.

Bringing your work home. If you spend part or all of your working hours in front of a computer terminal, you might be able to do some or all of your work at home, with the help of your modem. (Of course, that may mean giving up office parties and coffee breaks.) This can be technically tricky if your work computer doesn't use a standard set of protocols; you might have to get a technician to help you over the rough spots. It can also be tricky politically; many bosses shudder at the idea of not being able to keep an eye on their underlings. But as more progressive businesses continue to show that telecomputing can increase productivity as well as employee morale, attitudes are beginning to change.

Information transfer. If your modem allows for uploading and downloading, you can use it with your computer to transfer information—data, programs, recipes, whatever—directly to anybody else with comparable equipment. This can be faster and more reliable than sending diskettes, cassettes, or paper by mail.

Videotex. If you're disappointed that the big information utilities transfer only text, then Videotex might be for you. Videotex, an interactive electronic system that allows for the transmission of text and pictures, is on its way. Prestel, the British Post Office's videotex system, has been operational since the late seventies, and similar systems are being developed in America now.

Until recently, the development of American videotex has been hindered by the lack of standards for transmission and display of graphic information. But a standard now exists, and software is available (from Avcor, of Toronto, Ontario, Canada) to make the Commodore 64 into a bona-fide videotex terminal. By the

time you read this, VIEWTRON videotex service should be available to American subscribers from Viewdata Corporation. The experts can't agree as to whether this kind of service will catch on in America; it's up to you to decide!

GIVING YOUR 64 THE SOUL OF A NEW MACHINE

Imagine this interchange in a record store:

"Do you have the latest album by the Cybernetic Chauvinists?"

"That depends. What kind of stereo do you have?"

"ThunderWunder 5000."

"Sorry, but that album isn't available in that format. It may be released late this year, but nobody's sure. If you're in a big hurry, you'll need to get another stereo."

This kind of nonsense doesn't happen any more in the record industry, because it's been around long enough that a standard format has developed for record albums, and all major record companies and stereo manufacturers comply with that standard. 🐾 But the microcomputer industry is still young and scrappy, and nobody's willing to give up the systems they've developed without a fight. Most experts claim that a shakedown is inevitable, but until that happens we'll have to put up with scenes like this one when we're shopping for software. As a Commodore 64 owner, you're pretty much limited to the software that's been specifically developed for that machine, no matter how much you want that new game that's been written for the Atari.

🐾 That standard is being threatened by new technological developments even as I write this, but that's another story.

You may not be able to plug that Atari game into your 64, but you are definitely NOT limited to tailor-made 64 software. All you need to do is fool your 64 into thinking it's some other kind of a computer! There are at least three ways of doing that, ranging from a simple treatment of software hypnosis to a complete hardware surgical brain transplant!

Making your 64 into a Pet. Your 64 is a direct descendant of the Commodore Pet, with more similarities than differences, and it's not hard at all to make it forget the differences. There are several PET EMULATOR programs available from a variety of sources that can do the job in seconds. One low-cost emulator program can be found on Commodore's Bonus Disk Pack and Bonus Cassette Pack; some no-cost emulators are making the public-domain circuits. As soon as you run one of these programs (which do things like POKE values into memory to relocate the locations that control the screen) your 64 becomes a Pet, ready to run any disk or tape program written in Pet Basic 2.0. Basic 4.0 programs and programs designed for an 80 column screen probably won't work, but the vast majority of the available Pet programs will. (This is especially important if you want educational software, because the Pet has been one of the most popular school computers since the early days.)

Putting an Apple in your 64. The Commodore 64 is, in many ways, a low cost version of the Apple II. Sure, there are plenty of differences that Commodore loyalists and Apple addicts love to argue about. But the truth is that underneath the very different skins of these two machines live similar machines, right down to the microcomputer chip that runs the show. (The Commodore 6510 is almost identical to the Apple's 6502.)

Why, then, do so many people pay so much more for the Apple? That's an open question, but most analysts believe that a major part of the answer is software. The Apple has a bigger software base than almost any other computer ever made. Whatever you want to do, if it can be done with a microcomputer there's a good chance somebody's written an Apple-compatible program to do it.

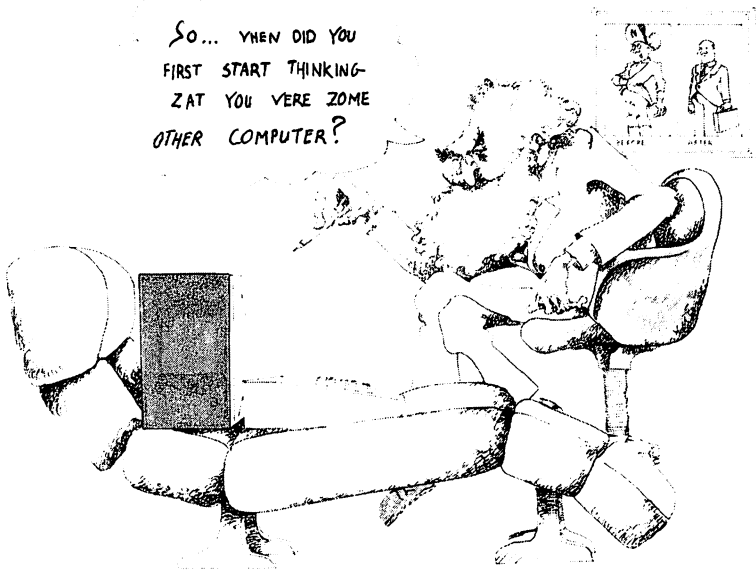
This doesn't do you much good as a 64 owner, unless you can find an Appleizer for your machine. A company called Advanced Integrated Development thought you might be interested, and has announced just such a product. Called the Amulator, it's a hardware product that you can easily install in your 64 to give it a split personality. Once that's done, your 64 will run disk-based programs written for either the 64 or the 40-column Apple without further modification!

That's no small accomplishment, either. In order to make that conversion, the Amulator has to not only reconfigure the circuitry of the computer, but it also has to convert the disk drive so that it behaves like the very different Apple disk drive. To keep it simple and affordable, the designers of the Amulator limited it to reading from Apple-formatted disks. Apple programs that require writing on disk—word processors, database managers, spreadsheets, and others—won't work properly with the Amulator: nor will programs that require an 80-column display or programs that take advantage of the enhancements built into the Apple //e. (The company plans a future upgrade to allow disk writing.) But even after you slice out those exceptions, you're still left with a big chunk of Apple software.

The Amulator hasn't yet reached the stores or my computer, so I can't report on how well it does what it does. Amulator programs like this one generally run converted programs slower than they run on their home computers, so video games may lose some of their excitement in the conversion. But the speed difference—probably in the neighborhood of 15 percent—is likely to be insignificant for most programs. If you can, give the Amulator a test drive in the store. If it's fast enough for you, it's a low cost way to expand your computer's software horizons.

CP/M: A brain transplant for your computer. While there are no real standards, most of the popular home computers do their thinking with some variant of the 6502 chip, and speak some dialect of Microsoft BASIC as their native high-level language. This BASIC serves not just as a programming language, but also as an OPERATING SYSTEM for controlling disk and file operations and performing general housekeeping.

Outside the home, it's not always that way. There are hundreds of kinds of microcomputers used for business and professional computing, with a staggering array of designs and features. Many of the newer business micros are built around a 16-bit chip rather than one of the traditional 8-bit models. (In a 16-bit computer, information is transported and stored in chunks of 16 bits rather than 8. This can make many operations faster, but the big advantage is that it allows the computer to easily access more than 64K of memory.)



But far and away, most desk-top micros are still 8-bit models. And with a few exceptions, almost all of those 8-bit computers are built with the same operating system: CP/M. Since its introduction in 1975, CP/M (Control Program for Microcomputers) has become one of the few things to be accepted as a defacto standard by most of the microcomputer industry. What is CP/M?

CP/M is an operating system: a program that takes care of many of the messiest details of computing, such as communicating between the computer and the different input and output devices, control and organization of disk files, and editing, loading and execution of programs. In effect, the operating system acts as an interface between the computer and you, accepting and acting on your commands. For example, when you type DIR to any machine running under CP/M, it responds by displaying the disk directory on the screen.

The CP/M operating system is designed to run on just about any microcomputer built around microprocessors in the 8080 family. These include the 8085 and the Z80, but not your Commodore's 6510. Which means that in order to run the CP/M operating system on your 64, you have to install another microprocessor.

Why bother? What can CP/M offer you? Not much, if your computing needs don't go beyond typical household uses. But if you want to use your computer as a professional tool, whether business or scientific, then CP/M can open up a world of useful software that your 64's built-in brain never dreamed of. Accounting, payroll, inventory, database management, word processing, special industry applications, statistics—whatever your needs, there's probably somebody ready to sell you a CP/M program to do the job. Or better yet, you just might find it in the vast library of public domain software that circulates among CP/Mers. If all else fails, you might want to write one using one or more of the many programming languages or utilities that are available through CP/M.

Commodore offers an inexpensive CP/M converter for the 64. It's easy to install: first, plug the Z-80 cartridge into the back of your machine; second, load and run the CP/M software from the accompanying disk. That's all you need to do to make your computer speak CP/M! Now, where's all that software?

Several problems are hidden in that question. In the first place, some CP/M software requires more than the 48K of memory that your 64 has when operating as a CP/M machine. Second, almost all existing CP/M software assumes that your computer

has an 80-column screen display, and your 64 definitely doesn't. Finally, CP/M software on disk has to be formatted so that your Commodore drive can read it, and most available CP/M software isn't.

The lack of memory may limit your software choices, but there's at least one way around the last two problems. It involves using a Z-80 converter made by Data 20 instead of Commodore's. The price of the Z-80 VIDEO PAK is several times the cost of Commodore CP/M; but it may be worth it if you're serious about CP/M, because it lets you choose between 40- and 80-column display, whether you're using the Z-80 processor or the Commodore's standard brain. A monochrome monitor is almost always a necessity because 80-column displays are difficult or impossible to read on color-TV screens. The Data 20 Z-80 converter doesn't include a CP/M diskette; instead, it comes with a CP/M look-alike called SB-80 that supports CP/M software.

As for the software shortage problem, the Data-20 people can supply a list of Commodore-formatted software on request. At present, most of the programs are produced by Lifeboat Associates, the manufacturers SB-20, but it's likely that other CP/M-compatible software will start showing up on Commodore-formatted disks shortly. If by chance the CP/M software you want appears first in a 40-column format, you can switch the Z-80 Video Pak to 40-column and run the Commodore CP/M software disk with it. If you can't locate the program you want in a Commodore-compatible disk format, there's still hope. Some companies specialize in converting programs to different disk formats. Watch the classified ads in the micro mags, especially the *Computer Shopper*.

Since most CP/M software is aimed at the business market, prices tend to run much higher than what you might be used to paying. But enterprising penny-pinchers can benefit from CP/M, too. Dozens of free and almost-free bulletin boards around the continent exist to provide information and program-exchanges for CP/M users. When your CP/M-compatible computer is connected to one of these bulletin boards, you can read CP/M news and notices, leave messages for other CP/Mers, or

download public-domain CP/M programs directly into your computer's memory to save on disk for later use. This is a wonderful way to get the software you need without having to worry about disk formats at all.

To take advantage of these generous services, you'll need a modem and a CP/M-compatible terminal program. One of the best available is a public domain CP/M program called MODEM7, originally written by Ward Christensen and embellished by countless unsung programmers over the years. Any CP/M modem program will work, as long as it used the "Ward Christensen Protocol" for downloading of software.

Probably the best way to locate a CP/M terminal program is to connect with other CP/M users. Your dealer might have information on a local CP/M user group. Or you can use your modem to talk to the national network of CP/M users via one of the CP/M bulletin boards. You don't need to be running in CP/M to leave messages or read text from a CP/M bulletin board. You can use the same procedure you use to connect to any bulletin board or information utility. You just won't be able to download CP/M programs. With a little persistence and luck, you'll be able to find a version of MODEM7 or the equivalent for the 64.

User's Guide, an excellent publication for CP/M and IBM users, occasionally contains lists of CP/M User Groups and bulletin boards. Issue number 3 contains a list of bulletin boards and detailed instructions for using MODEM7 to download programs. (Issue number 1, by the way, is a CP/M tutorial for beginners.) *Infoworld*, the weekly microcomputer magazine, occasionally prints a helpful column on public domain programs, CP/M and otherwise, called "Shareware."

(A final note: if you're an old friend of CP/M from some other system, you're going to be disappointed with the speed of CP/M and SB-80 on the 64. The problem is not in the operating system, but in the 1541 disk drive and interface.)

WHAT NEXT?

Once you start dabbling with hardware add-ons, you're opening up all kinds of possibilities. Here are a few of the products that have been released, or simply announced, for use with the Commodore 64. (As you read about these gizmos, remember that many products announced at press conferences or trade shows are never heard from again.)

Doubling your 64's screen width. Most "professional" computers, as opposed to home computers, display not 40, but 80 characters of information on a screen line. The 64 can be converted to an 80-column display using either software or hardware. COLOR 80, from Computer Marketing Services, is a low-cost program that converts the display to 80 columns without sacrificing color. The tiny characters can't be read on a color TV, and they're not exactly sharp on the Commodore color monitor. But that's not the fault of the software — it's due to the physical limitations of color display.

The VIDEO-PAK 80 from Data 20 is a relatively high-priced one of the most popular plug-in hardware device for converting the 64 to standard 80-column screen format. It's essentially the same as the Z-80 Pack described above, but without the Z-80 microprocessor and SB-80. It allows you to switch back and forth between the standard 40-column display and 80-column display after you get everything plugged in properly. But an 80-column display is only useful if you're connecting via modem to a computer that uses that format, or if you have software for your computer that's designed for 80-columns. With this in mind, the Data-20 package has built-in terminal-emulation software for telecommunication, and includes 80-column word processor and mailing list programs.

Number punching. The Commodore 64 keyboard is one of the best to be found on a low-cost computer. But if you're doing lots of work with numbers, you'll regret the lack of an adding-machine-style NUMERIC KEYPAD. Fortunately, an add-on numeric keypads are available from Computer Place. A simple installation procedure connects their pad by a cable

through the 64's cassette output porthole. Once it's installed, your machine will accept numeric input from either this keypad of the computer keyboard.

Pointing with a pen. A light pen is an amazing device, when you think about it. Plugged into your joystick port, it can be used to point to anyplace on the screen. The computer can tell within reasonable limits where the pen is pointing, and use that information as input to program. 🐾 A good light pen has all kinds of potential uses: as a pointing device for very young children; as an input device for disabled folks who can't handle a keyboard; as a convenient way of moving a cursor or a game piece in a variety of programs.

There are several inexpensive light pens on the market, including the EDUMATE LIGHT PEN from Futurehouse (Programmer's Institute). This pen comes with four ready-to-run BASIC programs that work directly with the pen. One is a disk menu program that lets you point to your choice of commands (delete a file, load a file, etc.) Another is a 3-D tic-tac-toe game that pits you and your light pen against the computer. A third lets you point to spots on the screen to make musical tones. But the easiest to use is a high-resolution drawing program that allows you draw or write on your TV screen by simply choosing a color, pressing F1, and moving the pen around on the screen. This program isn't a sophisticated graphics tool like Doodle (Chapter 6), Paint Magic (Chapter 6), or Koala Painter (discussed below), but it's lots of fun at a bargain price.

These four programs show a cross section of possibilities for using the light pen. They also show the weaknesses of this kind of device. You may grow tired of holding your arm out when using the drawing program, or find it irritating to switch back and forth between keyboard and light pen with the disk menu. Or you may become frustrated with the occasional inaccurate response that the light pen gives with some software. This is

🐾 It can do this because the TV picture is made by a single point-source of light that travels in rows across the screen, faster than your eye can see. The computer determines the pen location by timing when this beam passes under the pen. That's the general idea, anyway.

most noticeable in the music program, which almost fills the screen with small spots, each representing a musical note. But the program often fails to recognize the spot you're pointing to, playing a wrong note or none at all. Part of the problem is due to the physical limitations of the computer and the pen; but a well-designed program—such as the drawing program—can get around these limitations and give accurate responses.

Futurehouse promises a series of high quality educational programs that you can buy to use with the light pen. Or, if you're so inclined, you can write your own light pen software using the technical notes and sample programs in the booklet that comes with it.

Finger pointing and finger painting. Another kind of pointing device is a TOUCH TABLET—a flat panel that's pressure-sensitive, so that it can detect the presence and location of your finger or a stylus on its surface. One such device for the Commodore 64 is the KoalaPad Touch Tablet, from Koala Technologies, Inc. The KoalaPad is about the size and weight of a book, so that it can comfortably be held in your hand or lap while you work. In addition to the touch panel, it has two buttons, similar to the “fire” buttons on a joystick. The device plugs directly into the joystick port of your 64.

Like a light pen, the touch tablet has tremendous potential as an input device for people who don't like—or can't use—keyboards. Many researchers believe that touch tablets and similar devices may soon replace keyboards as the main source of microcomputer input. But any useful input device has to have software that can understand the signals it sends.

The KoalaPad comes with a disk containing a marvelous program called KOALA PAINTER to show off the potential of the pad and the computer. The KOALA PAINTER allows you to draw color pictures on your computer screen by moving your finger or stylus around on the KoalaPad. This program is simple enough for a preschooler to use, with features to satisfy a professional artist.

The menu screen lets you choose from 16 colors and eight "brushes" by simply pointing to your choice and pressing a button. The menu also offers choices between several powerful commands, the simplest of which is DRAW. Other commands include LINE, LINES, and RAYS, for connecting points with straight lines; FRAME, BOX, CIRCLE, and DISK, for putting rectangles and circles (hollow or filled with color) in your picture with two touches; FILL, for filling in a part of the screen; COPY, for repeating a part of the picture on another part of the screen; MIRROR, for creating symmetric kaleidoscopic effects; ZOOM, which magnifies part of the screen for close detail work; XCOLOR, for changing one color in your picture to another, and OOPS, for undoing the last thing you did. Other commands allow you to switch back and forth between two screen pictures and store and retrieve pictures on disk.

This laundry list of commands doesn't convey how easy and fun it is to use the program to create stunning color graphics on your computer screen. What can you do with your creations? There's no way to print color pictures on a B&W computer printer, so the instruction manual tells you how to photograph your work on the screen. Since pictures can be saved on disk, it's theoretically possible to use them in other programs of your own creation. But that requires a little bit of technical expertise and experimentation. (Hopefully the Koala people will release a utility program or subroutine to make the job easier in the future.)

KOALA PAINTER is the first in a series of KoalaWare Programs that use the touch tablet with the C-64. At this writing the only other KoalaWare program for the 64 is a clever educational game called SPIDER EATER. The spider Eater Disk comes with a thin plastic overlay that fits on the surface of the pad, turning it into a mini-piano keyboard with labeled keys. A SPIDER EATER player used these keys to respond whenever a spider crawls along a line of the musical staff on the screen. Pressing the correct key causes the SPIDER EATER to zap the spider with its tongue before it can make its way to the web at the top of the screen. Both speed and accuracy are important in this game, but the opening menu offers a choice of speeds. Note names are displayed on both the staff and the keyboard,

so even musical beginners can play the game and learn notation along the way. More advanced players interested in ear training can choose to make the spiders invisible, so that the only way to tell which key to press is by listening to the note being played. The SPIDER EATER program includes two bonuses: a music composer option that allows you to play melodies on the tiny keyboard; and crazy sounds, which lets your moving finger make bizarre music (?) that may remind you of John Cage or Pink Floyd.

SPIDER EATER may not be as impressive as KOALA PAINTER, but it's still one of the best educational games so far for the 64. Other educational, recreational, and design titles from KoalaWare are in the works. If they're as well-done as KOALA PAINTER and SPIDER EATER, the KoalaPad has a bright future.

The KoalaPad isn't the only touch-tablet you can buy for the 64. Chalkboard, Inc., has aimed at the same market with a device called the CHALKBOARD POWER PAD. There are several striking differences between these two devices. For one thing, the Power Pad is gigantic, with a full square foot of touch-sensitive surface. And because of a different design technology, the Power Pad is capable of detecting more than one touch at a time. This design difference seems to bring with it a couple of negative side effects: the Power Pad responds more slowly to touch, at least with the software I've tried. And the Power Pad, when it's in use, disables the keyboard of the 64, so that you have to use the pad for ALL inputs to the machine. This can be bothersome when it means that you have to save pictures using names like "RED-RED-BLUE" and musical compositions with names like "C-C-D."

The Chalkboard people have an impressive line of software planned for use with the Power Pad, most of it educational. The first two programs released for the 64 are LEO'S 'LECTRIC PAINTBRUSH, a simple program for drawing color pictures., and MICROMAESTRO, a music program that turns the Power Pad into an intelligent musical instrument. Each of these titles are packaged in a large plastic case that contains a software cartridge, a friendly instruction manual, and a plastic overlay

to convert the blank Power Pad into a clearly-labeled input device. A lot of thought has obviously gone into making these programs easy-to-learn. My 5-year-old son saw the MICRO-MAESTRO Piano-keyboard on the Power Pad and knew right away how to make notes, melodies, and chords, displayed on a treble clef while they're played. It didn't take him much longer to learn that the spot labeled "Play All" would play back his composition so that he could listen or add new notes. The sluggish response of the Power Pad, which might hinder a more seasoned musician, didn't bother him at all. (And it didn't bother the Power Pad when his 2-year-old sister walked in and sat in the middle of it.)

LEO'S LECTRIC PAINTBRUSH is just as easy for kids to use, with a menu of colors and commands right on the Power Pad. But it doesn't compete with KOALA PAINTER as a serious drawing tool. Chalkboard will soon be releasing a more sophisticated art program for the Power Pad called MICRO ILLUSTRATOR (the same program sold with the Apple version of Koala Pad.)

If you're trying to decide between KoalaPad and Power Pad, you'd be wise to try this program along with KOALA PAINTER. Then look at the range of available software, to see which meets your needs better. At this point, Koala's software and convenient features make it look like a better choice, but the game is still young.

(If you're a do-it-yourselfer, you'll be glad to know that both Koala and Chalkboard have released programming guides that explain in plain English and BASIC how to write programs that use touch-tablet input. Koala's Instant Programmer's Guide even comes with a disk containing a tutorial and several sample programs, including a simple sprite editor.)

An interesting Plot, and character development, too. If you like drawing pictures, you might want to consider the Commodore 1520 printer/plotter. A PLOTTER is an output device for making line-drawings and graphs on paper. This one, designed to connect directly to the Commodore 64 or the VIC-20, has four different color ball-point pens that

can draw high-resolution pictures on as 4-1/2-inch wide roll of standard adding machine paper. It also works as a very slow printer by drawing each letter individually. Commodore's LOGO (Chapter 6) has some special commands for drawing pictures with this plotter.

Musicians, take note. For music makers, Commodore promises a full-sized, organ-style keyboard that will plug into the 64 and turn SID into a full-fledged musical instrument. The add-on synthesizer will contain three additional SID chips, giving it 12 voices in all. The software will allow you to control those voices, and even save what you play in memory for later playback. (Ever play a duet with yourself?)

If Commodore doesn't come through with the goods, somebody will. Waveform Corporation has in the works a "professional quality" music keyboard for the 64 called COLORTONE KEYBOARD. This keyboard is just one of the musical hardware/software products they'll be releasing for the 64 as part of their new MUSICALC system. If their just-released MUSICALC 1 SYNTHESIZER AND SEQUENCER is any indication, this system may just be the professional tool company claims it to be. It's definitely the most complete—and complex—musical software tool yet released for the 64. At the visual center of MusiCalc 1 is the on-screen control panel, which contains over 70 variables—tempo, attack, decay, modulation, transposition, rhythm, and more—that you control with the computer's keyboard while the music plays. You can build a sound around one of the preset musical patterns that's built in, or you can create your own from scratch and record it for future use.

All of this freedom may be a little bit overwhelming if you're new to the world of electronic music—it's like sitting down in front of a new kind of musical instrument. But the book that comes with the program has a tutorial to guide you through the basics, starting with hooking a stereo to your computer to getting the best sound. If you were hoping you'd be able to play the synthesizer that came inside your Commodore 64, you should definitely look into MusiCalc 1.

The rest of the MusiCalc system is built around MusiCalc 1: MusiCalc2, the ScoreWriter, will allow you to turn your music into notes on the screen or graphics printer; MusiCalc 3, the Keyboard Maker, will let you modify the Basic MusiCalc keyboard to other musical scales; MusiCalc 4 will work with the Colortone Keyboard, allowing you to play any scale in any key, and the MusiCalc templates will give you new scores and rhythms to play along with, from African to New Wave.

The MusiCalc software is priced higher than most recreational software, so it may not be a best buy unless you're serious about music. If you're curious but poor, check out the bargain-priced MusiCalc demo disk, which plays a jukebox full of tunes to show off the music-making potential of the system.

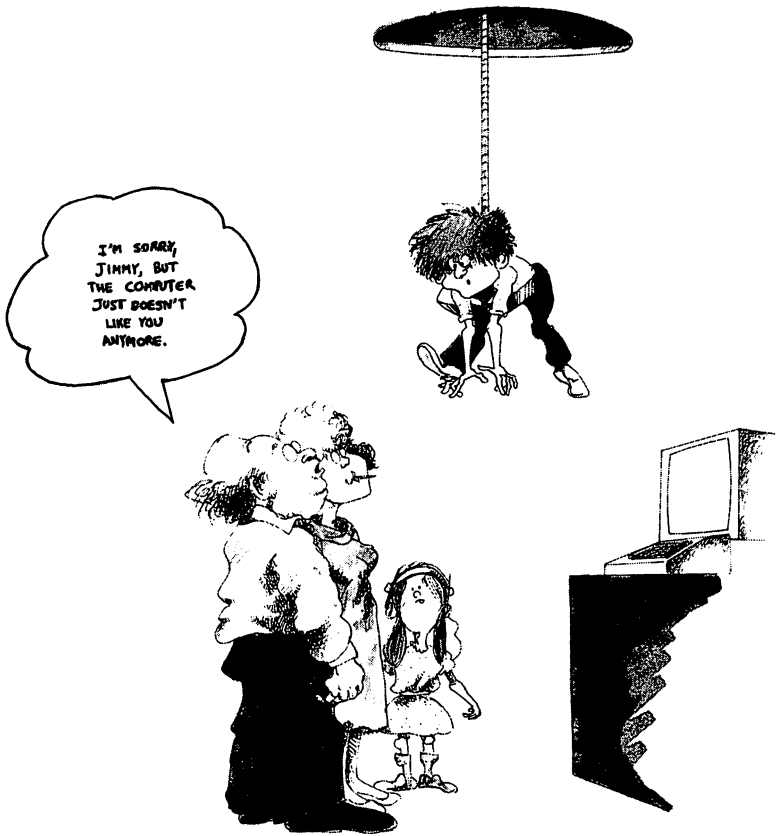
Speech! Finally, Commodore's been talking about a plug-in VOICE SYNTHESIZER called Magic Voice that will allow you to make your computer speak with BASIC commands like 10 SAY "HELLO!" Besides being fun, a speech synthesizer has great potential for opening the world of computers to preschool children, visually impaired adults, and anyone else who can't easily read text on a screen. Commodore's synthesizer will come with some software for preschool education, with more cartridges in the works.

Apparently the Commodore synthesizer is a hardware device with a fixed vocabulary and voice, both of which can be augmented with later products.

A more flexible approach to speech synthesis is used in S.A.M. (Software Automatic Mouth) from Don't Ask (distributed by Tronix). When you load and run S.A.M., you give your computer the ability to say anything that you can type using the 64's built-in sound synthesizer. S.A.M. does a surprisingly good job of translating straight English text into understandable talk. But it works even better if you type the words to be spoken in a phonetic language. S.A.M. can even be used to incorporate speech into your own programs. You can control the pitch and inflections in S.A.M.'s voice so that he can sound like a variety of characters. (S.A.M. comes with some demo programs in which he recites speeches and sings the Star Spangled Banner.) Even

in his best voice, S.A.M. sounds unmistakably mechanical and very soft-spoken. Still, it's a kick to hear your computer talk and sing to you.

Open, Sesame! This chapter wouldn't be complete without a word about using your computer to control other devices in your home. Science fiction has been telling us for years about how the home of the future will have computer-controlled lights, appliances, burglar alarms, locks, heating systems, and exotica. The future is here, if you want it.



First, the easy way. **HOME BRAIN**, from HyPertek, Inc., is a device that can be used to control or communicate with just about any electrical device in your house. You use your computer to program Homebrain for your particular household needs (or let the Hypertech consultants do it for you), and Homebrain takes over, leaving your computer free to play games or balance the budget. The Homebrain box sits tucked away on some remote wall of the house, cooking your breakfast before you get up in the morning, controlling thermostats, turning lights on when you enter a room and off when you leave, rotating the solar collector, warning you of break-ins or fires (calling the police or fire department, if you like), turning on the dishwasher when electricity is cheapest, and checking to make sure the kids made it home on time.

When you're away from home, Homebrain can make the house look lived-in, beef up the security system, even keep track of how many people come to the door. If you take your computer with you, you can talk to Homebrain via modem, changing the program or simply finding out what's been happening at home while you're gone.

Of course, Homebrain can't do all of this by itself. To detect fires, it needs to be connected to smoke detectors. To watch for burglars, the kids, or empty rooms, it uses motion detectors. Add the cost of these devices, the wiring, and programming to make Homebrain work in your house, and you have an expensive package. The company claims that it costs no more than a typical home security system, and they may be right. But that's still more than many of us can afford.

If you like to do it yourself, and you have the background skills, you can do many of the same things with your computer for a fraction of the cost of Homebrain. You'll need a low-cost relay-controlling interface like the **VIC/64 RELAY CARTRIDGE** from Computer Marketing Services. This cartridge plugs into the user port of your 64 (or a VIC), and contains 6 relay outputs and 2 octo-coupler inputs. To use it, you'll probably have to know how to design and wire some electrical relay circuits. In addition, you'll need to write programs to tell your computer how to communicate with the relay cartridge and the circuits

you connect to it. The instruction manual gives the necessary technical specifics, but it assumes you have a solid understanding of BASIC and electrical circuitry.

And so on. The story doesn't stop here. Every month brings new hardware and software products that stretch the limits of your home computer. This is especially true of the Commodore 64, a relatively new and extremely popular machine. Each month the quantity and quality of new products goes up, and there's no end in sight. That makes finishing this book especially difficult. but a line has to be drawn, so (with apologies to Harry) the book stops here.

(It wouldn't be fair, though, to leave you hanging. So keep reading, and you'll find a list of microcomputer periodicals that can keep you up-to-date on developments in the exciting field of home computing.)

Appendix A

For More Information

No book on personal computing can tell you everything. I've tried in this book to give you a balanced introduction to the Commodore 64, and home computing in general. This section is designed to point you toward further information when you need it. It's divided into three sections: (1) periodicals; (2) books; and (3) addresses of on-line utilities, software and hardware companies, and user groups.

With things changing as fast as they are, magazines are essential tools for computerists. We're in the middle of a computer magazine explosion now, with a couple of new ones appearing each month. It's hard to predict at this point which of these will survive. The ones listed here are my current favorites.

Computer books are exploding now, too. With booksellers reporting that computer books account for more than 10 percent of their sales, everybody is trying to get into the act. So by the time you read this many of these titles will have been crowded off the shelves by all the new releases; which may or may not be better.

The on-line information utility scene is even more volatile. I've included addresses and phone numbers of utilities and bulletin boards discussed in Chapter 9.

Also included are names and addresses of major software and hardware suppliers mentioned in the book, in case your dealer doesn't have what you're looking for.

Finally, a partial listing of Commodore 64 user groups, one of the best sources of information and help.

PERIODICALS FOR KEEPING UP

Most computer books, including this one, are dated before they ever reach the shelves. A good way to remedy this is to supplement your book collection with periodicals. The ones I've listed here fall into three categories: general interest magazines, Commodore-related magazines, and special interest magazines. If you're serious about computing, I recommend you subscribe to at least one in each of the first two categories, and try to get your library to carry some of the others.

General interest computer magazines. The focus here is on magazines that include a significant amount of material related to home computing, rather than industry rags like *Datamation*, *Computer World*, and *Mini-Micro Systems*.

- *InfoWorld (The Newsweekly for Microcomputer Users)*
375 Cochituate Road
Box 837
Framingham, MA
(800) 343-5730

This microcomputer weekly is more like a slick newspaper than a magazine. The latest news and gossip from the microcomputer industry, no-holds-barred software and hardware reviews (with manufacturer responses to negative reviews), in-depth analysis of industry trends, probing feature articles on anything from telecomputing to software piracy, and scrappy editorials . . . there's something for everybody here. The editorial slant is refreshingly humanistic, and the quality of the writing is generally very high, with a minimum of technobabble. If I could only subscribe to one computer publication, *InfoWorld* would be it.

- *Creative Computing*
PO Box 789-M
Morristown, NJ 07960
(800) 631-8112

This is the oldest surviving microcomputer magazine. The emphasis is on microcomputer applications and software: programming tutorials, interviews with the pros, ready-to-type-in programs, industry news and views, and lots of product evaluation (hardware, software, and books), often grouped by category. (For example, every few months they do a Consumer Report-style look at printers.) Most of the articles are well written and non-technical, and there are plenty of ads for browsing. The articles and reviews are typically weighted slightly toward Apple systems, but there is a regular Commodore column. The overall philosophy seems to be that computing is, above all, fun.

- *Byte (The Small Systems Journal)*
PO Box 372
Hancock, NH 03449

Another vintage publication from the golden age of computer hobbyists, *Byte* is one of the most widely-read of the micro mags. The emphasis is on hardware, with lots of articles on how to make your own doowah out of the spare chips you have lying around the house. But there are also plenty of articles for the serious programmer, and enough ads to make *Byte* thicker than many phone books.

- *Popular Computing*
Box 307
Martinsville, NJ 08836

This magazine is *Byte*'s little brother, aimed at regular folks with no technical background. A nice mix of breezy articles, product reviews, comparative software evaluations, tutorials, and ads. Extremely readable and informative.

- *Compute! (The Journal of Progressive Computing)*
PO Box 5406
Greensboro, NC 27403
(800)334-0868

If you're a hacker at heart, you'll love *Compute!* The magazine does have articles on applications and products (although I've yet to see a negative review). But its real emphasis is on programming, with listings of games, utilities, and applications programs for many types of computers; tutorials on secret POKEs and other programming tricks; and a kind of "Dear Abby" for confused programmers. If *Creative Computing* has an Apple bias, *Compute!* leans toward Commodore. The ads do, too.

- *Whole Earth Software Review*
Box 428
Sausalito, CA 94966
(415)331-6249

This quarterly brings the clear, incisive style of the *Whole Earth Catalogs* to the world of computers. Besides software reviews, you'll find hardware reviews, book reviews, things the manuals forgot to tell you, tips, suggestions, complaints, debates, thought-provoking articles, and lots of reader input. They accept no ads, so they don't need to pussyfoot in their reviews. The best items from this review will appear along with other material in the *Whole Earth Software Catalog*.

- *Digit (The Video/Computing Connection for Young People)*
PO Box 27958
San Diego, CA 92128

A promising new publication aimed at the biggest new group of computer users: kids. Articles, fiction, Q&A, reviews, program listings, and tutorials in a colorful package.

Commodore-related publications. Cover-to-cover Commodore coverage.

- *Commodore (The Microcomputer Magazine)* and *Commodore Power Play*, CBM, Inc.
1200 Wilson Drive
West Chester, PA 19380
(800) 345-8112

The strength and the weakness of both of these magazines is that they are published by Commodore. They're ALL Commodore, and they sometimes contain information you won't find elsewhere. But most articles tend to read like advertising blurbs, and never is heard a discouraging word about Commodore. Each magazine contains a nice mix of product announcements and descriptions (I hesitate to call them reviews); news, programs, and tutorials. *Commodore*, the bimonthly, emphasizes business, education, and serious applications, while the quarterly *Power Play* leans toward recreational and educational computing.

- *Compute!'s Gazette*, same address as *Compute!* above. This spinoff of *Compute!* is aimed strictly at beginning and intermediate Commodore users. Like its parent publication, *Gazette* contains lots of program listings and programming tips, and a nice question-and-answer section.

- *Run (the Commodore 64 and VIC-20 Magazine)*
PO Box 372
Dalton, MA 01226

Another slick publication aimed at exactly the same audience that *Gazette* is going after: you. Programs, reviews, articles, tips, ads.

- *The Midnite Software Gazette*,
635 Maple,
Mt. Zion, IL 62549.

This low-budget magazine has the folksy feel of a user group newsletter, with lots of technical tips, software reviews, and newsy pieces written in a friendly (and sometimes amateur) style. Clearly put together by Commodore fans who don't always buy the Commodore party line. The first two years of publication have been compressed into one volume. *The Whole Pet Catalog*, an interesting look at Commodore before the 64, including a big section on public domain software.

- *Torpet (The Independent Commodore User's Magazine)*
1912A Avenue Road, Suite 1
Toronto, Ontario
M5M 4A1, Canada

This is the newsletter of the biggest Commodore User Group. It doesn't have as many reviews as *Midnite*, and it's not as much fun to read. But it's unmatched as a source of information on public domain software; each issue lists the latest acquisitions of the huge Torpet library. At least one member of your local user group should subscribe, so that you can have access to that library.

- *Commodore 64 User's Group Newsletter*
PO Box 572
Glen Ellyn, IL 60137

Newer and rougher than either of the two previous entries, with less useful information per issue. But the focus is on the 64.

Special interest publications. Odds and ends

- *Computer Shopper (the nationwide marketplace for computer equipment)*
PO Box F
Titusville, FL 32780

Basically a monthly newspaper of classified ads, with a few articles for filler. *Computer Shopper* is a good place to buy and sell second-hand equipment.

- *User's Guide*
PO Box 3050
Stanford, CA 94305

An outstanding bimonthly devoted to CP/M and related software (including public domain stuff), *User's Guide* is an attempt to provide the kind of documentation that should have come with the software in the first place. A well-written and well-edited mixture of tutorials, reference guides, resource lists, and articles. The first issue is a great introductory CP/M tutorial.

- *Comal Catalyst*
Suite 5-315, 5130 E. Charleston
Las Vegas, Nevada 89122

The voice of American Comal enthusiasts. If you're one, you should subscribe.

- *The On-Line Telephone Directory*
Box 10005
Kansas City, MO 64111-9990

A quarterly telecomputing newsletter containing phone numbers for all kinds of bulletin boards, with precious little explanatory information for novices.

- *Directory of On-Line Data Bases*, by Ed Ruth, N. Cuadra, David M. Abels, and Judith Wanger. Cuadra Associates. Kind of a periodical book, this one is too expensive for most homes; look for it in your library.

Microcomputer periodical indices. If you really want to stay on top of the field, you'll need to read several periodicals a month. Even if you can do that, though, it's difficult to remember what you've seen and where you've seen it. A cross-references index to microcomputer periodicals is a valuable tool for finding the information you need without burying yourself in irrelevant data. Three good ones are:

- *Periodical Guide for Computerists*
Applegate Computer Enterprises
PO Box 288
Applegate, OR 97530
- *Microcomputer Index*
2464 El Camino Real, Suite 247
Santa Clara, CA 95051
- *LAMP (Literature Analysis of Microcomputer Publications)*
200 Route 17
Mahwah, NJ 07430

The Periodical Guide for Computerists isn't as slick or professional as the other two, but it's considerably more affordable. The other two are priced more for libraries and businesses than for hobbyists.

GOOD BOOKS

These are the books I would use if I were going to try to teach myself about the topics they cover. These kinds of decisions are subjective; I suggest you browse before you buy.

Commodore 64 books. The first wave.

- *Commodore 64 User's Guide*. Howard W. Sams & Co. This is the book that came with your computer.
- *Commodore 64 Programmer's Reference Guide*. Howard W. Sams & Co. This is the book that SHOULD HAVE come with your computer. An essential book for anyone seriously interested in programming the 64 in BASIC or machine language. Sections were written with beginners in mind, but most of the book was designed as a reference for serious programmers.
- *The Elementary Commodore 64*, by William B. Sanders. DATAMOST, Inc. A light readable self-teaching guide to BASIC programming on the 64, with lots of cute programs to type in. Watch for a sequel.
- *Kids and the Commodore 64*, by Edward H. Carlson. DATA-MOST. The Commodore 64 version of Carlson's immensely popular *Kids and* series. Aimed at a seventh-grade reader (with chapter notes for parents and teachers); this is an excellent book for introducing kids to BASIC programming.
- *More Than 32 BASIC Programs for the Commodore 64*, by Tom Rugg, Phil Feldman, and Gene Moore dilithium Press. This is one of those books for people who like to type in programs. There are programs for everybody here, from home applications and games to graphics and math. A distinguishing feature of this book is the thorough documentation, including suggested changes you can make to the programs. (If you DON'T like to type, you can buy a diskette of the programs with the book.)

- *PC-Documate*, from SMA, 3700 Computer Drive, Raleigh, NC 27609. Not a book, but a useful reference template for 64 BASIC programmers that fits around the keyboard. More details on BASIC commands, sound, and graphics than this book's crib sheet.

General how-to computer books. Useful applications, programming, hardware tinkering, and money making! None of these books was written specifically for the 64, but all are useful anyway.

- *Writing in the Computer Age*, by Andrew Flugelman and Jeremy Joan Hewes. From selection to style, this book has a lot to say about writing with a word processor.

- *VisiCalc—Home and Office Companion*, by David M. Castlewitz and Lawrence J. Chisasky. Osborne/McGraw Hill. Lots of ideas here on how to use your spreadsheet program, whatever brand. (Of course, the exact templates will need a little bit of modification.)

- *32 VisiCalc Worksheets*, by Ted Lewis. dilithium. Another book full of interesting uses for spreadsheets, with Listings showing the exact contents of each cell.

- *Background Math for a Computer World*, by Ruth Ashley. John Wiley and Sons. A friendly little book that gives just what it promises.

- *Elementary BASIC* by Howard Ledgard and Andrew Singer. A must for Sherlock Holmes fans; Doctor Watson recounts Holmes's adventures with the analytic engine, in which he uses BASIC to solve crimes. Besides being entertaining, the book does a good job of teaching BASIC. (A Pascal version is also available.)

- *Learning with Logo*, by Dan Watt. McGraw Hill. A good LOGO tutorial by one of the experts.

- *Pascal Primer*, by David Fox and Mitchell Waite. Howard W. Sams & Co., Inc. An easy-to-read introduction to UCSD Pascal

that doesn't pretend to cover everything, but does a good job with what it does.

- *Oh! Pascal!* by Doug Cooper and Michael Clancy. W. W. Norton & Co. This book teaches standard Pascal as defined by Wirth, not UCSD Pascal, so you'll have to make some adjustments for your system. But for readability, thoroughness, and personality, *Oh! Pascal!* is hard to beat. It also has an excellent survey of data structures, the building blocks of computer science.
- *Comal Handbook*, by Len Lindsay. Reston Publishing Company, Inc. Organized more like a reference book than a tutorial, but if you know BASIC you should be able to pick up Comal from the many sample programs here.
- *Starting FORTH*, by Leo Brodie. Prentice Hall. The clearest, most readable introduction yet to Fig FORTH.
- *6502 Assembly Language Programming*, by Lance Leventhal. McGraw Hill. One of the better books on the subject.
- *CP/M Primer*, by Stephen M. Murtha and Mitchell Waite. A good introduction to CP/M. (Another is the first issue of *User's Guide*, listed in periodicals.)
- *Computer Language Reference Guide*, by Harry L. Helms, Jr. Howard W. Sans & Co. A neat little book that will help you to understand the main points of several computer languages, so you can make sense out of programs written in languages you don't know.
- *Program Design and Construction*, by David Higgins. Prentice Hall. Learn how to design programs that work. Very clear introduction to Warnier diagrams as a tool of program design. Examples are in BASIC, but the principles apply to all languages.
- *Program Design*. Various Authors. *Byte Books*. A series of articles (reprinted from *Byte*) covering all aspects of program design. Includes a condensed version of the Higgins book listed above.

- *Microcomputers for External Control Devices*, by James A. Gupton, Jr. dilithium Press. Want to use your computer to control your solar heating system or create a basement automated factory? This book is a good place to start.

- *How to Make Money with Your Microcomputer*, by Carl Townsend and Merl Miller. dilithium Press. From repairing to writing, there are lots of good ideas here for the enterprising reader.

- *Computer Careers*, by the editors of *Consumer Guide*. Fawcett. Where the jobs are and how to get them.

General computer books. A sampling of books that tell you what and why rather than how.

- *Katie and the Computer*, by Fred D'Ignazio and Stan Gillman. Creative Computing Press. A charming children's book about Katie's adventure when she falls into a computer. Computer literacy for the under-eight crowd.

- *Exploring the World of the Personal Computer*, by Jack M. Nilles. Prentice Hall, Inc. A thought-provoking study of the impact of the personal computer on our society; and it's not all pretty (or ugly). A good way to prepare yourself for the future we're about to enter.

- *Mindstorms: Children, Computers, and Powerful Ideas*, by Seymour Papert. Basic Books. If this utopian vision from the MIT designers of LOGO doesn't inspire you to add that language to your children's environment, then nothing will. Every teacher should read this; parents can benefit too.

- *Computer Power and Human Reason (From Judgement to Calculation)*, by Joseph Weizenbaum. W. H. Freeman & Co. Another MIT computer scientist speaks out on the things computers shouldn't do, even if they can. Probably more important now than when it was first published in the seventies.

- *The Compleat Computer*, by Dennie Van Tassel and Cynthia L. Van Tassel. SRA, Inc. A collection of articles, stories, pictures, cartoons (including *Doonesbury*), and trivia about computers in our lives and our future. Informative and fun.

ADDRESSES AND PHONE NUMBERS

Software and hardware companies. Where to write or call if your local dealer doesn't have what you want.

On-line information utilities.

- *BSR After Dark*, 1200 Route 7, Latham, NY 12110, (800)833-4707
- *Compuserve, Inc.*, 5000 Arlington Center Blvd., PO Box 20212, Columbus, OH 43220, (614)457-8650.
- *Delphi*, 3 Blackstone Street, Cambridge, MA 02139, (800)544-4005.
- *Dow Jones News Retrieval*, PO Box 300, Princeton, NJ 08540 (800)222-0081 or (609)452-7040.
- *Electronic Information Exchange System (EIES)*, New Jersey Institute of Technology, 323 High Street, Newark, NJ 07102
- *Knowledge Index*, Dialog Information Services, Inc., 3460 Hillview Avenue, Palo Alto, CA 94304, (800)528-6050, ext. 415.
- *Newsnet*, 945 Haverford Road, Bryn Mawr, PA 19010, (800)345-1301 or (215)527-8030.
- *Source Telecomputing*, 1616 Anderson Road, McLean, VA 22102, (800)336-3330 or (703)734-7500.

On-Line bulletin boards. The ones mentioned in Chapter 9, plus several Commodore Bulletin Boards. Phone numbers are modem numbers unless otherwise specified.

- *CLEO* (714)722-8868 by modem.
- *IF*, the Imagination Factory, Anaheim, CA.
- *Peacenet, the Disarmament Resource Center*, San Francisco, CA. (415)896-0893. (People line: (415)495-0526.)

Company List

Advanced Integrated Development
5901 John Martin Drive
Minneapolis, MN 55430
(800)328-8831 or (612)561-1645

Artworx Software Company, Inc.
150 North Main Street
Fairport, NY 14550
(800)828-6573

Avalon Hill Game Company
4517 Harford Road
Baltimore, MD 21214
(800)638-9292 or (301)254-5300

AB Computers
252 Bethlehem Pike
Colmar, PA 18915
(215)822-7727

Behavioral Engineering
230 Mount Hermon Road, Suite 207
Scotts Valley, CA 95066
(408)438-5649

Broderbund Software
1938 Fourth Street
San Rafael, CA 94901
(415)456-6424

Chalkboard, Inc.
3772 Pleasantdale Rd.
Atlanta, GA 30340
(800)241-3989 or (404)496-0101

Comm*Data Computer House, Inc.
320 Summit Avenue
Milford, MI 94901
(313)685-0113

Commodore Business Machines, Inc.
1200 Wilson Drive
West Chester, PA 19380
(215)431-9100

Computer Marketing Services, Inc.
300 W. Marlton Pike
Cherry Hill, NJ 08002
(609)795-9480

Computer Place
23914 Crenshaw Blvd.
Torrance, CA 90505
(213)325-4754

Computer Software Associates
50 Teed Drive
Randolph, MA 02368
(800)343-1078

Continental Software
1123 South Hindry Ave.
Los Angeles, CA 90045
(213) 417-8031

Counterpoint Software, Inc.
4005 West 65th Street
Minneapolis, MN 55435
(800)328-1223 or (612)926-7888

Creative Equipment
6864 West Flagler St.
Miami, FL 33144
(305)261-7866

Creative Software, Inc.
230 East Caribbean Drive
Sunnyvale, CA 94086
(408)745-1655

Data 20, Inc.
23011 Moulton Parkway Suite B10
Laguna Hills, CA 92653
(714)770-2366

Data Equipment Supply Corporation
8315 Firestone Blvd.
Downey, CA 89241

DATAMOST, Inc.
20660 Nordhoff Street
Chatsworth, CA 91311
(818) 709-1202

Datasoft, Inc.
9421 Winnetka Avenue
Chatsworth, CA 91311
(818) 701-5161

Don't Ask Computer Software
2265 Westwood Blvd., Suite B-150
Los Angeles, CA 90064
(213)397-8811

The Einstein Corporation
11340 West Olympic Blvd.
Los Angeles, CA 90064
(213)477-6733

Electronic Arts
2755 Campus Drive
San Mateo, CA 94403
(415)571-7171

Epyx Computer Software
1043 Kiel Court
Sunnyvale, CA 94089
(408)745-0700

Human Engineered Software (HesWare)
150 North Hill Drive
Brisbane, CA 94005
(415)468-1111 or (800)227-6703

Infocom, Inc.
55 Wheeler Street
Cambridge, MA 02138
(617)492-1031

Koala Technologies Corporation
3100 Patrick Henry Drive
Santa Clara, CA
(408)986-8866 or (800)227-6703

Lifeboat Associates
1651 Third Avenue
New York, NY 10028
(212)860-0300

Lightning Software
PO Box 11725
Palo Alto, CA 94306
(415)327-3280

Midwest Micro, Inc.
311 West 72nd Street
Kansas City, MO 64114
(816)333-7200

Mirage Concepts, Inc.
2519 W. Shaw - No. 106
Fresno, CA 93711
(209)227-8369

Omni Unlimited
105 South Los Robles
Pasadena, CA
(213)795-6664

Pro-Line Software
755 The Queensway East
Mississauga, Ontario, Canada L4Y4C5
(416)273-6350

Professional Software, Inc.
51 Fremont Street
Needham, MA 02194
(617)444-5224

Programmer's Institute
PO Box 3470
Chapel Hill, NC 27514
(800)334-SOFT or (919)967-0861

Quick Brown Fox
548 Broadway, Suite 4F
New York, NY 10012
(212)925-8290

Sierra On-Line, Inc.
Coarsegold, CA 93614
(209) 683-6858

Sirius Software
10364 Rockingham Drive
Sacramento, CA 95827
(916)366-1195

Spinnaker Software Corporation
215 First Street
Cambridge, MA 02142
(800)323-8088

Sublogic Corporation
713 Edgebrook Drive
Champaign, IL 61820
(217)359-8482 or (800)637-4983

Synapse Software
5221 Central Avenue, #200
Richmond, CA 94804
(415)527-7751

Taylormade Software
PO Box 5574
Lincoln, NE 68505
(402)464-9051

Timeworks, Inc.
PO Box 321
Deerfield, IL 60015
(312)291-9200

Tronix
8295 South La Cienega Blvd.
Inglewood, CA 90301
(213)215-0529

TOTL Software, Inc.
1555 Third Avenue
Walnut Creek, CA 94596
(415)943-7877

United Microware Industries (UMI)
3503-C Temple Avenue
Pomona, CA 91768
(714)594-1351

Wadsworth Electronic Publishing Co.
10 Davis Drive
Belmont, CA 94002
(415)595-2350

Wilserv Industries
PO Box 456
Bellmawr, NJ 08031
(609)227-8696

Commodore User Groups

Commodore User Groups. If there's one in your area, look into it. It could be your best single source of information and support. If there isn't, you might still benefit from joining a national organization. There's one specifically for C-64 users called, appropriately enough, The Commodore 64 Users Group. In return for annual dues, comparable to the price of a magazine subscription, you'll get a monthly newsletter, a bi-monthly magazine, access to public-domain software, and a possible source of help and advice. The address is:

The Commodore 64 Users Group
P.O. Box 572
Glen Ellyn, IL 60137
(313) 790-4320

The following list of local Commodore user groups is reprinted with permission from *Commodore (the Microcomputer Magazine)*, Issue 26, pp 115-121 (address listed earlier in this appendix). For a more up-to-date list, consult the current issue of that magazine or *Power Play*.

ALABAMA

Huntsville PET Users Club
9002 Berclair Road
Huntsville, AL 35802
Contact: Hal Carey
Meetings: Every 2nd Thursday

Riverchase Commodore Users Group
617 Grove St.
Birmingham, AL 35209
(205) 988-1078
Ken Browning

Wiregras Micro-Computer Society
Commodore SIG
109 Key Bend Rd.
Enterprise, AL 36330
(205) 347-7564
Bill Brown

ALASKA

COMPOOH-T
c/o Box 118
Old Harbor, AK 99643
(907) 286-2213

ARIZONA

Catalina Commodore Computer Club
2012 Avenida Guillermo
Tucson, AZ 85710
(602) 296-6766
Geroge Pope
1st Tues. 7:30 p.m.

Central Arizona PET People
842 W. Calle del Norte
Chandler, AZ 85224
(602) 899-3622
Roy Schahrer

ACUG
c/o Home Computer Service
2028 W. Camelback Rd.
Phoenix, AZ 85015
(602) 249-1186
Dan Deacon
First Wed. of month

Arizona VIC 20-64 Users Club
232 W. 9th Place North
Mesa, AZ 85201
Donald Kipp

Arizona VIC & 64 Users
904 W. Marlboro Circle
Chandler, AZ 85224
(602) 963-6149
Tom Monson

ARKANSAS

Commodore/PET Users Club
Conway Middle School
Davis Street
Conway, AR 72032
Contact: Geneva Bowlin

Booneville 64 Club
c/o A. R. Hederich
Elementary School
401 W. 5th St.
Booneville, AR 72927
Mary Taff

The Siloam Commodore
Computer Club
P.O. Box 88
Siloam Springs, AR 72761
(501) 524-5624
Ken Emanuelson

CALIFORNIA

SCPUG Southern California
PET Users Group
c/o Data Equipment Supply Corp.
8315 Firestone Blvd.
Downey, CA 90241
(213) 923-9361
Meetings: 1st Tuesday of each month

Valley Computer Club
1913 Booth Road
Ceres, CA 95307

PUB of Silicon Valley
22355 Rancho Ventura Road
Cupertino, CA 95014

Lincoln Computer Club
750 E. Yosemite
Manteca, CA 95336
John Fung, Advisor

PET on the Air
525 Crestlake Drive
San Francisco, CA 94132
Max J. Babin, Secretary

PALS (Pets Around)
Livermore Society
886 South K
Livermore, CA 94550
(415) 449-1084
Every third Wednesday 7:30 p.m.
Contact: J. Johnson

SPHINX
7615 Leviston Ave.
El Cerrito, CA 94530
(415) 527-9286
Bill MacCracken

San Diego PUG
c/o D. Costarakis
3562 Union Street
San Diego, CA 92119
(619) 235-7626
7 a.m.-4 p.m.

Walnut Creet PET Users Club
1815 Ygnacio Valley Road
Walnut Creek, CA 94596

Jurupa Wizards
8700 Galena St.
Riverside, CA 92509
(714) 781-1731
Walter J. Scott

The Commodore Connection
2301 Mission St.
Santa Cruz, CA 95060
(408) 425-8054
Bud Massey

San Fernando Valley
Commodore Users Group
21208 Nashville
Chatsworth, CA 91311
(818) 709-4736
Tom Lynch, 2nd Wed. 7:30

VACUUM
277 E. 10th Ave.
Chico, CA 95926
(916) 891-8085
Mike Casella
2nd Monday of month

South Bay Commodore Users Group
1402 W. 218th St.
Torrance, CA 90501
Contact: Earl Evans

SLO VIC 20/64 Computer Club
1766 9th St.
Los Osos, CA 93402

The Diamond Bar R.O.P. Users Club
2644 Amelgado
Hacienda Hgts., CA 91745
(213) 333-2645
Don McIntosh

Commodore Interest Association
c/o Computer Data
14660 La Paz Dr.
Victorville, CA 92392
Mark Finley

Computer Barn Computer Club
319 Main St., Suite #2
Salinas, CA 93901
(408) 757-0788
S. Mark Vanderbilt

Humboldt Commodore Group
P.O. Box 570
Arcata, CA 95521
R. Turner

Napa Valley Commodore
Computer Club
c/o Liberty Computerware
2680 Jefferson St.
Napa, CA 94558
(707) 252-6281
Mick Winter
1st & 3rd Mon. of month

S.D. East County C-64 User Group
6353 Lake Apopka Place
San Diego, CA 92119
(619) 698-7814
Linda Schwartz

Commodore Users Group
4237 Pulmeria Ct.
Santa Maria, CA 93455
(805) 937-4174
Gilbert Vela

Bay Area Home Computer Asso.
Walnut Creek Group
1406 N. Broadway at Cypress
Walnut Creek, CA 94596
Wil Cossel
Sat 11 a.m. to 3 p.m.

Amateurs and Artesians Computing
P.O. Box 682
Cobb, CA 95426

20/64 Users Group
P.O. Box 18473
San Jose, CA 95158
Don Cracraft
1st Sunday, 6 p.m.,
Mercury Savings

The Valley Computer Club
2006 Magnolia Blvd.
Burbank, CA 91506
1st Wed. 7 p.m.

The Commodore Tech. Users of
Orange Co.
P.O. Box 1497
Costa Mesa, CA 92626
(714) 731-5195
Roger Fisher

C-64 West Orange County
Users Group
P.O. Box 1457
Huntington Beach, CA 92647
(714) 842-4484
Philip Putman
2nd & 4th Tues. of month

Antelope Valley Commodore
Users Group
POB 4436
Lancaster, CA 93539
(805) 942-2626
James Haner
1st Saturday

Diablo Valley Commodore
Users Group
762 Ruth Dr.
Pleasant Hill, CA 94523
(415) 671-0145
Ben Braver
2nd & 4th Thurs. 7:30 p.m.

Commodore Connection
11652 Valverde Ave.
Riverside, CA 92505
(714) 689-7447
Tony Alvarez

COLORADO

VICKIMPET Users Group
4 Waring Lane, Greenwood Village
Littleton, CO 80121
Contact: Louis Roehrs

Colorado Commodore Computer Club
2187 S. Golden Ct.
Denver, CO 80227
(303) 986-0577
Jack Moss, Meet: 2nd Wed.

CONNECTICUT

John F. Garbarino
Skiff Lane Masons Island
Mystic, CT 06355
(203) 536-9789

Commodore User Club
Wethersfield High School
411 Wolcott Hill Road
Wethersfield, CT 06109
Contact: Daniel G. Spaneas

New London County
Commodore Club
Doolittle Road
Preston, CT 06360
Contact: Dr. Walter Doolittle

FLORIDA

Jacksonville Area
PET Society
401 Monument Road, #177
Jacksonville, FL 32211

South Florida
PET Users Group
Dave Young
7170 S.W. 11th
West Hollywood, FL 33023
(305) 987-6982

Bay Commodore Users Group
c/o Gulf Coast Computer Exchange
241 N. Tyndall Pkwy.
P.O. Box 6215
Panama City, FL 32401
(904) 785-6441
Richard Scofield

Gainesville Commodore Users Club
3604-20A SW 31st Dr.
Gainesville, FL 32608
Louis Wallace

64 Users Group
P.O. Box 561689
Miami, FL 33156
(305) 274-3501
Eydie Sloane

Brandon Users Group
108 Anglewood Dr.
Brandon, FL 33511
(813) 685-5138
Paul Daugherty

Brandon Commodore Users Group
414 E. Lumsden Rd.
Brandon, FL 33511

Gainesville Commodore Users Group
Santa Fe Community College
Gainesville, FL 32602
James E. Birdsell

Commodore Computer Club
P.O. Box 21138
St. Petersburg, FL 33742

Commodore Users Group
545 E. Park Ave., Apt. #2
Tallahassee, FL 32301
(904) 224-6286
Jim Neill

The Commodore Connection
P.O. Box 6684
West Palm Beach, FL 33405

El Shift OH
P.O. Box 548
Cocoa, FL 32922
Mike Schnoke
Sat. Mornings/every 4 to 6 weeks

Miami 20/64
12911 S.W. 49th St.
Miami, FL 33175
(305) 226-1185

Tampa Bay Commodore
Computer Club
10208 N. 30th St.
Tampa, FL 33612
(813) 977-0877

GEORGIA

Golden Isles Commodore Users Club
Bldg. 68, FLETC
Glynco, GA 31524
Richard L. Young

HAWAII

Commodore Users Group of Honolulu
c/o PSH
824 Bannister St.
Honolulu, HI
(808) 848-2088
3rd Fri. every month

20/64 Hawaii
P.O. Box 966
Kailua, HI 96734
Wes Goodpaster

Commodore Users Group of Honolulu
1626 Wilder #701
Honolulu, HI 96822
(808) 848-2088
(808) 944-9380
Jay Calvin

IDAHO

GHS Computer Club
c/o Grangeville High School
910 S. D St.
Grangeville, ID 83530
Don Kissinger

S.R.H.S. Computer Club
c/o Salmon River H.S.
Riggins, ID 83549
Barney Foster

Eagle Rock Commodore Users Group
900 S. Emerson
Idaho Falls, ID 83401
Nancy J. Picker

ILLINOIS

Shelly Wernikoff
2731 N. Milwaukee Avenue
Chicago, IL 60647

VIC-20/64 Users Support Group
c/o David R. Tarvin
114 S. Clark Street
Pana, IL 62557
(217) 562-4568

Central Illinois PET User Group
635 Maple
Mt. Zion, IL 62549
(217) 864-5320
Contact: Jim Oldfield

ASM/TED User Group
200 S. Century
Rantoul, IL 61866
(217) 893-4577
Contact: Brant Anderson

PET VIC Club (PVC)
40 S. Lincoln
Mundelein, IL 60060
Contact: Paul Schmidt, President

Rockford Area PET Users Group
1608 Benton Street
Rockford, IL 61107

Commodore Users Club
1707 East Main St.
Olney, IL 62450
Contact: David E. Lawless

Chicago Commodore 64
Users & Exchange Group
P.O. Box 14233
Chicago, IL 60614
Jim Robinson

Fox Valley PET Users Group
833 Willow St.
Lake in the Hills, IL 60102
(312) 683-7321
Art DeKneef

The Commodore 64 Users Group
P.O. Box 572
Glen Ellyn, IL 60137
(312) 790-4320
Gud Pagnotta

RAP 64/VIC Regional
Assoc. of Programmers
10721 S. Lamon
Oak Lawn, IL 60453
Bob Hughes

The Kankakee Hackers
RR #1, Box 279
St. Anne, IL 60964
(815) 933-4407
Rich Westerman

WIPUG
Rt. 5, Box 75
Quincy, IL 62301
(217) 656-3671
Edward Mills

INDIANA

PET/64 Users
10136 E. 96th St.
Indianapolis, IN 46256
(317) 842-6353
Jerry Brinson

Cardinal Sales
6225 Coffman Road
Indianapolis, IN 46268
(317) 298-9650
Contact: Carol Wheeler

CHUG (Commodore Hardware
Users Group)
12104 Meadow Lane
Oaklandon, IN 46236
Contact: Ted Powell

Northern Indiana Commodore
Enthusiasts
927 S. 26th St.
South Bend, IN 46615
Eric R. Bean

Commodore Users Group
1020 Michigan Ave.
Logansport, IN 46947
(219) 722-5205
Mark Bender

Computer Workshop VIC-20/64 Club
282 S. 600 W.
Hebron, IN 46341
(219) 988-4535
Mary O'Bringer

The National Science Clubs of America
Commodore Users Division
7704 Taft St.
Merrillville, IN 46410
Brian Lepley or Tom Vlasic

Commodore Computer Club
3814 Terra Trace
Evansville, IN 47711
(812) 477-0739
John Patrick, President

IOWA

Commodore User Group
114 8th St.
Ames, IA 50010

Quad City Commodore Club
1721 Grant St.
Bettendorf, IA 52722
(319) 355-2641
John Yigas

Commodore Users Group
965 2nd St.
Marion, IA 52302
(319) 377-5506
Vern Rotert
3rd Sun. of month

Siouxland Commodore Club
2700 Sheridan St.
Sioux City, IA 51104
(712) 258-7903
Gary Johnson
1st & 3rd Monday of month

421 W. 6th St.
Waterloo, IA 50702
(319) 232-1062
Frederick Volker

Commodore Computers Users
Group of Iowa
Box 3140
Des Moines, IA 50316
(515) 263-0963 or (515) 287-1378
Laura Miller

KANSAS

Wichita Area PET Users Group
2231 Bullinger
Wichita, KS 67204
(316) 838-0518
Contact: Mel Zandler

Kansas Commodore Computer Club
101 S. Burch
Olathe, KS 66061
Contact: Paul B. Howard

Commodore Users Group
6050 S. 183 St. West
Viola, KS 67149
Walter Lounsbery

Walnut Valley Commodore
User Group
1003 S. 2nd St.
Arkansas City, KS 67005
Bob Morris

KENTUCKY

Louisville Users of Commodore KY.
(LUCKY)
c/o Computer Showroom
1247 Hurstbourne
Louisville, KY 40222
2nd Monday

LOUISIANA

Franklin Parish Computer Club
#3 Fair Ave.
Winnisboro, LA 71295
James D. Mays, Sr.

NOVA
917 Gordon St.
New Orleans, LA 70117
(504) 948-7643
Kenneth McGruder, Sr.

MARYLAND

Assoc. of Personal Computer Users
5014 Rodman Road
Bethesda, MD 20016

Blue TUSK
700 East Joppa Road
Baltimore, MD 21204
Contact: Jim Hauff

House of Commodore
8835 Satyr Hill Road
Baltimore, MD 21234
Contact: Ernest J. Fischer

Long Lines Computer Club
323 N. Charles St., Rm. 201
Baltimore, MD 21201
Gene Moff

VIC & 64 Users Group
The Boyds Connection
21000 Clarksburg Rd.
Boyd, MD 20841
(301) 428-3174
Tom DeReggi

Hagerstown Users Group
1201-B Marshall St.
Hagerstown, MD 21740
(301) 790-0968
Greg Stewart
1st & 3rd Friday of month 6:30 p.m.

Rockville VIC/64 Users Group
5112 Parklawn Terrace, Apt. #103
Rockville, MD 20853
(301) 231-7823
Tom Pounds

The Compucats' Commodore
Computer Club
680 W. Bel Air Ave.
Aberdeen, MD 21001
(301) 272-0472
Betty Jane Schueler

Westinghouse BWI
Commodore User Group
Attn: L. Barron, Mail Stop 5156
P.O. Box 1693
Baltimore, MD 21203

MASSACHUSETTS

Commodore Users Club
Stoughton High School
Stoughton, MA 02072
Contact: Mike Lennon

Berkshire PET Lovers
CBM Users Group
Taconic High
Pittsfield, MA 01201

The Boston Computer Society
Three Center Plaza
Boston, MA 02108
(617) 367-8080
Mary E. McCann

Masspet Commodore Users Group
P.O. Box 307
East Taunton, MA 02718
David Rogers

Raytheon Commodore Users Group
Raytheon Company
Hartwell Rd. GRA-6
Bedford, MA 01730
John Rudy

Commodore 64 Users Group of
The Berkshires
184 Highland Ave.
Pittsfield, MA 01201
Ed Rucinski

Cape Cod 64 Users Group
358 Forrest Rd.
S. Yarmouth, MA 02664
1-800-225-7136
Jim Close
(In MA call) 1-800-352-7787

MICHIGAN

David Liem
14361 Warwick Street
Detroit, MI 48223

Commodore User Club
32303 Columbus Drive
Warren, MI 48093
Contact: Robert Steinbrecher

Commodore Users Group
c/o Family Computer
3947 W. 12 Mile Rd.
Berkley, MI 48072

South Computer Club
South Jr. High School
45201 Owen
Belleville, MI 48111
Ronald Ruppert

Commodore Users Group
c/o Eaton Rapids Medical Clinic
101 Spicerville Hwy.
Eaton Rapids, MI 48827
Albert Meinke III, M.D.

South East Michigan Pet
Users Group
Box 214
Farmington, MI 48024
Norm Eisenberg

Commodore Computer Club
4106 Eastman Rd.
Midland, MI 48640
(517) 835-5130
John Walley
9:30 p.m. Sept/May

VIC, 64, PET Users Group
8439 Arlis Rd.
Union Lake, MI 48085
(313) 363-8539
Bert Searing

ComputerTowne
35171 Grand River
Farmington, MI 48024
(313) 471-4216

Ann Arbor Commodore Users Group
Ann Arbor, MI 48103
(313) 994-4751
Art Shaw
3rd Tues. 7:30-10:00

DAB Computer Club
P.O. Box 542
Watervliet, MI 49098
(616) 463-5457
Dennis Burlingham

West Michigan Commodores
c/o R. Taber
1952 Cleveland Ave., S.W.
Wyoming, MI 49509
(616) 458-9724
Gene Traas

MINNESOTA

MUPET (Minnesota Users of PET)
P.O. Box 179
Annandale, MN 55302
c/o Jon T. Minerich

Twin Cities Commodore
Computer Club
6623 Ives Lane
Maple Grove, MN 55369
(612) 424-2425
Contact: Rollie Schmidt

MISSISSIPPI

Commodore Biloxi User Group
(ComBUG)
Universal Computer Services
3002 Hwy. 90 East
Ocean Springs, MS 39564
(601) 875-1173
John Lassen

MISSOURI

KCPUG
5214 Blue Ridge Boulevard
Kansas City, MO 64133
Contact: Rick West
(816) 356-2382

Commodore User Group of St. Louis
Box 6653
St. Louis, MO 63125-0653
Dan Weidman, New Members
1541 Swallowtail Dr.
St. Louis, MO 63125-0653

Worth County PET Users Group
Grant City, MO 64456
(816) 564-3551
David Hardy

Mid-Missouri Commodore Club
1804 Vandiver Dr.
Columbia, MO 65201
(314) 474-4511
Phil Bishop

Joplin Commodore Computers
Users Group
422 S. Florida Ave.
Joplin, MO 64801
R.D. Connely

MONTANA

Powder River Computer Club
Powder River County High School
Broadus, MT 59317
Contact: Jim Sampson

Commodore User Club
1109 West Broadway
Butte, MT 59701
Contact: Mike McCarthy

NEBRASKA

Greater Omaha Commodore 64
Users Group
2932 Leawood Dr.
Omaha, NE 68123
(402) 292-2753
Bob Quisenberry

NEVADA

Las Vegas PET Users
Suite 5-315
5130 E. Charleston Blvd.
Las Vegas, NV 89122
Gerald Hasty

NEW JERSEY

Amateur Computer Group
18 Alpine Drive
Wayne, NJ 07470

Somerset Users Club
49 Marcy Street
Somerset, NJ 08873
Contact: Robert Holzer

Educators Advisory
P.O. Box 186
Medford, NJ 08055
(609) 953-1200
John Handfield

ACGNJ PET/VIC/CBM
User Group
30 Riverview Terr.
Belle Mead, NJ 08502
(201) 359-3862
J. M. Pylka

South Jersey Commodore Computer
Users Club
46-B Monroe Park
Maple Shade, NJ 08052
(609) 667-9758
Mark Orthner
2nd Fri. of month

Parsippany Computer Group
51 Ferncliff Rd.
Morris Plains, NJ 07950
(201) 267-5231
Bob Searing

NEW HAMPSHIRE

Northern New England
Computer Society
P.O. Box 69
Berlin, NH 03570

C-64 U.S.E.R.S. User Software
Exchange Pro
Rochester, NH 03867
Paul Kyle

NEW MEXICO

Commodore Users Group
6212 Karlson, NE
Albuquerque, NM 87113
(505) 821-5812
Danny Byrne

NEW YORK

Capital District 64/VIC-20
Users Group
363 Hamilton St.
Albany, NY 12210
(518) 436-1190
Bill Pizer

Long Island PET Society
Ralph Bressler
Harborfields HS
Taylor Avenue
Greenlawn, NY 11740

PET User Club of Westchester
P.O. Box 1280
White Plains, NY 10602
Contact: Ben Meyer

Commodore Masters
25 Croton Ave.
Staten Island, NY 10301
Contact: Stephen Farkouh

SPUG
4782 Boston Post Rd.
Pelham, NY 10803
Paul Skipski

L&M Computer Club
VIC-20 & 64
4 Clinton St.
Tully, NY 13159
(315) 696-8904
Dick Mickelson

Commodore Users Group
1 Corwin Pl.
Lake Katrine, NY 12449
J. Richard Wright

VIC-20/Commodore 64
Users Group
31 Maple Dr.
Lindenhurst, NY 11757
(516) 957-1512
Pete Lobol

New York Commodore Users Group
380 Riverside Dr., 7Q
New York, NY 10025
(212) 566-6250
Ben Tunkelang

Hudson Valley Commodore Club
1 Manor Dr.
Woodstock, NY 12498
F.S. Goh
1st Wednesday of month

LIVICS (Long Island VIC Society)
20 Spyglass Lane
East Setauket, NY 11733
(516) 751-7844
Lawrence Stefani

Manhattan 64
426 West 48th
New York, NY 10036
(212) 307-6519
Charles Honce

Adirondack Commodore 64
Users Group
205 Woodlawn Ave.
Saratoga Springs, NY
(518) 584-8960
Paul Klompas

Rockland County Commodore
Users Group
P.O. Box 573
Nanuet, NY 10965
Ross Garber

New York 64 Users Group
222 Thompson St.
New York, NY 10012
(212) 673-7241
Bruce Cohen

Finger Lakes Commodore
Users Group
c/o Rose City Computer Associates
229 West Union St.
Newark, NY 14513
(315) 331-1185

NORTH CAROLINA

Amateur Radio PET Users Group
P.O. Box 30694
Raleigh, NC 27622
Contact: Hank Roth

Microcomputer Users Club
Box 17142 Bethabara Sta.
Winston-Salem, NC 27116
Joel D. Brown

Raleigh VIC-20/64 Users Group
410-D Delta Court
Cary, NC 27511
(919) 469-3862
Larry Diener

OHIO

Dayton Area PET User Group
933 Livingston Drive
Xenia, OH 45385
B. Worby, President
(513) 848-2065
J. Watson, Secretary
(513) 372-2052

Central Ohio PET Users Group
107 S. Westmoor Avenue
Columbus, OH 43204
(614) 274-6451
Contact: Philip H. Lynch

Commodore Computer Club
of Toledo
734 Donna Drive
Temperance, MI 48182
Gerald Carter

Chillicothe Commodore Users Group
P.O. Box 211
Chillicothe, OH 45601
William A. Chaney

Licking County 64 Users Group
323 Schuler St.
Newark, OH 43055
(614) 345-1327

11433 Pearl Rd.
Strongsville, OH 44136
Paul M. Warner

C.P.U. Connection
P.O. Box 42032
Brook Park, OH 44142
Danni Hudak

Commodore Users Group
18813 Harlan Dr.
Maple Heights, OH 44137
(216) 581-3099
Carl Skala

OKLAHOMA

Southwest Oklahoma
Computer Club
c/o Commodore Chapter
P.O. Box 6646
Lawton, OK 73504
1:30 at Lawton City Library

Tulsa Area Commodore Users Group
Tulsa Computer Society
P.O. Box 15238
Tulsa, OK 74112
Annette Hinshaw

Commodore Oklahoma Users Club
4000 NW 14th St.
Oklahoma City, OK 73107
(405) 943-1370
Stanley B. Dow

Commodore Users
Box 268
Oklahoma City, OK 73101
Monte Maker, President

Commodore Users of Norman
209 Brookwood
Noble, OK 73068
Matt Hager

OREGON

NW PET Users Group
John F. Jones
2134 N.E. 45th Avenue
Portland, OR 97213

Pacific Northwest Commodore
Users Group
P.O. Box 2310
Roseburg, OR 97470
(503) 672-7591
Richard Tsukiji

PENNSYLVANIA

PET User Group
Gene Beals
P.O. Box 371
Montgomeryville, PA 18936

Penn Conference Computer Club
c/o Penn Conference of SDA
720 Museum Road
Reading, PA 19611
Contact: Dan R. Knepp

PACS Commodore Users Group
LaSalle College
20th & Olney Ave.
Philadelphia, PA 19141
(215) 951-1258
Stephen Longo

Glen Schwartz
806 Avon
Philadelphia, PA 19116

Gene Planchak
4820 Anne Lane
Sharpville, PA 15150
(412) 962-9682

PPG (Pittsburgh PET Group)
c/o Joel A. Casar, DMD
2015 Garrick Drive
Pittsburgh, PA 15235
(412) 371-2882

Westmoreland Commodore
Users Club
c/o DJ & Son Electronics
Colonial Plaza
Latrobe, PA 15650
Jim Mathers

COMPSTARS
440 Manatawny St.
Pottstown, PA 19464
Larry Shupinski, Jr.
Meet at Audio Video Junction

Commodore Users Club
3021 Ben Venue Dr.
Greensburg, PA 15601
(412) 836-2224
Jim Mathers

G.R.C. User Club
300 Whitten Hollow Rd.
New Kensington, PA 15068
Bill Bolt

NADC Commodore Users Club
248 Oakdale Ave.
Horsham, PA 19044
Norman McCrary

CACC (Capitol Area Commodore
Club)
134 College Hill Rd.
Enola, PA 17025
(717) 732-2123
Lewis Buttery
Union Deposit Mall at 7 p.m.

G/C Computer Owners Group
c/o Gilbert Associates, Inc.
P.O. Box 1498
Reading, PA 19607
Extension 6472
Jo Lambert (215) 775-2600

Boeing Employees Personal
Computer Club
The Boeing Vertol Co.
P.O. Box 16858
Philadelphia, PA 19142
(215) 522-2257
Jim McLaughlin

South Central PA Commodore Club
2109 Cedar Run Dr.
Camp Hill, PA 17011
(717) 763-4219
David Persing

Main Line Commodore Users
Group (MLCUG)
c/o Main Line Computer Center
1046 General Allen Lane
West Chester, PA 19380
(215) 388-1581
Emil Volcheck

Commodore Users Group
781 Dick Ave.
Warminster, PA 18974
Matt Matulaitis

RHODE ISLAND

Irving B. Silverman, CPA
160 Taunton Ave.
E. Providence, RI 02914
Contact: Michelle Chavanne

Newport VIC/64 Users
10 Maitland Ct.
Newport, RI 02840
(401) 849-2684
Dr. Matt McConeghy

Commodore Users Group
c/o Data-Co.
978 Tiogue Ave.
Coventry, RI 02816
(401) 828-7385
Victor Moffett

SOUTH CAROLINA

Beaufort Technical College
100 S. Ribaut Rd.
Beaufort, SC 29902
Dean of Instruction

Computer Users Society
of Greenville
Horizon Records-Home Computers
347 S. Pleasantburg Dr.
Greenville, SC 29607
(803) 235-7922
Bo Jeanes

Commodore Computer Club
of Columbia
318 Quincannon Dr.
Columbia, SC 29210
Buster White Sect./Treas.

Spartanburg Commodore
Users Group
803 Lucerne Dr.
Spartanburg, SC 29302
(803) 582-5897
James Pasley

SOUTH DAKOTA

PET User Group
515 South Duff
Mitchell, SD 57301
(605) 996-8277
Contact: Jim Dallas

VIC/64 Users Club
608 West 5th
Pierre, SD 57501
(605) 224-4863
Larry Lundeen

TENNESSEE

River City Computer
Hobbyists
Memphis, TN 38128
1st Mon. at Main Library

Nashville Commodore Users Group
P.O. Box 121282
Nashville, TN 37212
3rd Thurs. at Cumberland Mus

Commodore User Club
Metro Computer Center
1800 Dayton Blvd.
Chattanooga, TN 37405
Mondays 7:30 p.m.

Metro-Knoxville 64 Users Club
7405 Oxmoor Rd., Rt. #20
Knoxville, TN 37921
(615) 938-3773
Ed Pritchard

Memphis Commodore Users Group
2476 Ridvers Ave.
Memphis, TN 38127
(901) 358-5823
Harry Ewart

TEXAS

SCOPE

1020 Summit Circle
Carrollton, TX 75006

PET Users

2001 Bryan Tower
Suite 3800
Dallas, TX 75201

Larry Williams

P.O. Box 652
San Antonio, TX 78293

PET User Group

John Bowen
Texas A & M
Microcomputer Club
Texas A & M, TX 77843

CHUG (Commodore Houston

Users Group)
8738 Wildforest
Houston, TX 77088
(713) 999-3650
Contact: John Walker

Corpus Christi Commodores

3650 Topeka St.
Corpus Christi, TX 78411
(512) 852-7665
Bob McKelvy

Commodore Users Group

5326 Cameron Rd.
Austin, TX 78723
(512) 459-1220
Dr. Jerry D. Frazee

VIC Users Group

3817 64th Dr.
Lubbock, TX 79413

Southeast Houston VIC

Users Group
11423 Kirk Valley Dr.
Houston, TX 77089
(713) 481-6653

64 Users Group

2421 Midnight Circle
Plano, TX 75075
S.G. Grodin

Savid Computer Club

312 West Alabama, Suite 2
Houston, TX 77006
Davi Jordan, Chairman

Gulf Coast Commodore Users Group

P.O. Box 128
Corpus Christi, TX 78403
(512) 887-4577
Lawrence Hernandez

Mid-Cities Commodore Club

413 Chisolm Trail
Hurst, TX 76053
Garry Wordelman

Mid-Cities Commodore Club

413 Chisolm Trail
Hurst, TX 76053
Bruce Nelson

Interface Computer Club

814 North Sabinas
San Antonio, TX 78207
M. E. Garza, President

UTAH

Utah PUG

Jack Fleck
2236 Washington Blvd.
Ogden, UT 84401

The Commodore Users Club

742 Taylor Avenue
Ogden, UT 84404
Contact: Todd Woods Kap, President
David J. Shreeve, Vice President

Northern Utah VIC & 64

Users Group
P.O. Box 533
Garland, UT 84312
David Sanders

The Commodore Users Group

652 West 700 North
Clearfield, UT 84015
(801) 776-3950
Rodney Keller, Richard Brenchly

VIRGINIA

Northern VA PET Users
Bob Karpen
2045 Eakins Court
Reston, VA 22091
(803) 860-9116

Dale City Commodore User Group
P.O. Box 2004
Dale City, VA 22193
(703) 680-2270
James Hogler

Tidewater Commodore Users Group
4917 Westgrove Rd.
Virginia Beach, VA 23455
Fred Monson

Fredericksburg Area
Computer Enthusiasts
P.O. Box 324
Locust Grove, VA 22508
(703) 972-7195
Michael Parker

Commonwealth 20/64
Users Group
1773 Walnwright Dr.
Reston, VA 22090
(703) 471-6325
Tal Carawan, Jr.

Peninsula Commodore 64
Users Group
124 Burnham Place
Newport News, VA 23606
(804) 595-7315
Richard G. Wilmoth

Norfolk Users Group
1030 West 43rd St. B-4
Norfolk, VA 23508
489-8292
Larry Pearson

WASHINGTON

NW PET Users Group
2565 Dexter N. 3203
Seattle, WA 98109
Contact: Richard Ball

PET Users Group
c/o Kenneth Tong
1800 Taylor Ave. N102
Seattle, WA 98102

Whidbey Island Commodore
Computer Club
947 N. Burroughs Ave.
Oak Harbor, WA 98277
Michael D. Clark

Central Washington
Commodore Users Group
1222 S. 1st St.
Yakima, WA 98902
Tim McElroy

Blue Mountain Commodore
Users Club
667 Canary Dr.
Walla Walla, WA 99362
(509) 525-5452
Keith Rodue

Spokane Commodore User Group
N. 4311 Whitehouse
Spokane, WA 99205
(509) 328-1464
Stan White

WEST VIRGINIA

Personal Computer Club
P.O. Box 1301
Charleston, WV 25325
Cam Cravens

WISCONSIN

Sewpus
c/o Theodore J. Polozynski
P.O. Box 21851
Milwaukee, WI 53221

Waukesha Area Commodore
User Group (WACUG)
256½ W. Broadway
Waukesha, WI 53186
Contact: Walter Sadler
(414) 547-9391

Commodore User Group
1130 Elm Grove St.
Elm Grove, WI 53122
Tony Hunter

Commodore 64 Software
Exchange Group
P.O. Box 224
Oregon, WI 53575
E.J. Rosenberg

C.L.U.B. 84
6156 Douglas Ave.
Caledonia, WI 53108
(414) 835-4645 pm
Jack White
2nd Sat every month 10:00 am

VIC-20 & 64 User Group
522 West Bergen Dr.
Milwaukee, WI 53217
(414) 476-8125
Mr. Wachtl

Menomonie Area Commodore
Users Group
510 12th St.
Menomonie, WI 54751
(715) 235-4987
Mike Williams

C.U.S.S.H.
3614 Sovereign Dr.
Racine, WI 53406
(414) 554-0156
Tim Tremmel
3rd Saturday of month

Madison Area Commodore
Users Group
1552 Park St.
Middleton, WI 53562
(608) 831-4852
John Carvin
3rd Thurs. each month

WYOMING

Commodore Users Club
c/o Video Station
670 North 3rd #B
Laramie, WY 82070
Pamela Nash

CANADA

Toronto PET Users Group, Inc.
1912A Avenue Rd., Ste. 1
Toronto, Ontario, Canada
M5M 4A1
(416) 782-8900
or call (416) 782-9252
Contact: Chris Bennett

PET Users Club
c/o Mr. Brown
Valley Heights Secondary School
Box 159
Langton, Ont. N0E 1G0

Vancouver PET Users Group
P.O. Box 91164
West Vancouver, British Columbia
Canada V7V 3N6

CCCC (Canadian Commodore
Computer Club)
c/o Strictly Commodore
47 Coachwood Place
Calgary, Alberta, Canada
T3H 1E1
Contact: Roger Olanson

W.P.U.G.
9-300 Enniskillen Ave.
Winnipeg, Manitoba R2V 0H9
Larry Neufeld

Arva Hackers
Medway High School
Arva, Ontario N0M 1C0
D. Lerch

Nova Scotia Commodore
Computer Users Group
66 Landrace CRes.
Dartmouth, N.S. B2W 2P9
Andrew Cornwall

Bonnyville VIC Cursors
Box 2100
Bonnyville, Alberta T0A 0L0
(403) 826-3992
Ed Wittchen

Commodore Users Club of Sudbury
938 Brookfield Ave.
Sudbury, Ontario
P3A 4K4

PET Educators Group
P.O. Box 454
Station A
Windsor, Ontario
N9A 6L7

COMVIC
P.O. Box 1688
St. Laurent
Montreal, Quebec
H4L 4Z2

Calgary Commodore Users Group
37 Castleridge Dr., N.E.
Calgary, Alberta
T3J 1P4
John Hazard

GERMANY

Kettenberg 24
D 5880 Lueden Scheid
West Germany
Rudi Ferrari

ITALY

Commodore 64 Club
Universita di Studi shan
V. Avigliana 13/1
10138 TORINO
ITALY

KOREA

Commodore Users Club
K.P.O. Box 1437
Seoul, Korea
Contact: S. K. Cha

MEXICO

Asociacion De Usuarios
Commodore
c/o Alejandro Lopez
Arechiga
Holbein 174-6° Piso
Mexico 18, D.F.

Club de Usuarios Commodore
Sigma del Norte
Mol del Valle, Local 44
Garza Garcia, N.L. 66220

NEW ZEALAND

Commodore Users Group
Meet at VHF Clubrooms
Hazel Ave.
Mount Roskill
3rd Wed. of month, 7:30 pm
Roger Altena 278-5262

E.R. Kennedy
c/o New Zealand Synthetic
Fuels Corp. Ltd.
Private Bag
New Plymouth

PUERTO RICO

CUG of Puerto Rico
RFD #1, Box 13
San Juan, PR 00914
Ken Burch

VIC 20 User Group
655 Hernandez St.
Miramar, PR 00907
Robert Morales, Jr.

INDEX

A

A.B. Computers	242	BASIC 19, 56, 67, 133, 137, 138, 194, 248, 250, 252, 268	
accumulator	96	BASIC 2.0	116
acoustic modems	280	BASIC 4.0	116, 157
ADA	267	basic arithmetic	57
address	30	BASIC commands	67
advanced arithmetic	60	BASIC interpreter	257
Adventure	164	Basic Rule Of Law and Order	70
adventure and fantasy games	163	Beatles, The	278
adventure games	200	Behavioral Engineering, Inc.	190
Algebra Arcade	187	Benji Discovery Series	191
algorithm	249	Benji's Space Rescue	172
ANALOG	280	binary	65, 271
AND statement	90	binary digit	65
Apple	226	binary search	249
Apple II	158	bit	65, 270
appointment book	242	Blade of Blackpoole	176
Aristotle	207	block	116, 261
arithmetic	57	Boilerplating	218
arithmetic skills	185	books	136
array	146	Boolean algebra	96
artificial intelligence	194	border color code	62
Artworx Software		bork, alfred	258
Company, Inc.	166, 185	bricklin, dan	226
ASC	145	Bridge 4.0	166
assembler	271	Broderbund Software	166, 221
assembler development system	271	buffer	282
assembly language ...	95, 254, 269, 270	bug	73
asterisk	114	bulletin boards	285, 289
Atari	158	business	19, 22
Attack Of The Mutant Camels	171	business programs	193
Avalon Hill Game Company	166	buying	42
average	80	byte	30, 61, 64, 65

B

backup	103, 132
backup copies	113, 125
Bank Street College of Education	221
Bank Street Writer	221

C

C-64 Wedge	121
CalcResult	226, 234
call subroutine	138
Canada	193
canned programs	97

DATAMOST	203
datasette	36, 102, 108
Datasoft	170
David's Midnight Magic	167
Deadline	172
debugging	132, 137
decimal numbers	66
decisions	89
dedicted microcomputer system	208
delating disk files	119
delay	86
delete	54
Delphi	282
Delphi news service	283
device number	112, 117, 126, 152
Diary-64	241
digital	280
dim	147
direct-connect modems	280
directory	114, 115
disk	108
disk directory	115
disk drive error channel	123
disk drive performance test	124
disk name	115, 118
diskette	33, 45, 110
diskette care	125
documentation	131
Don't Ask Computer Software	187
Doodle	203
DOS 5.1	121
DOS Wedge	121, 253
dot matrix printer	40
Dow Jones	282
Dow Jones News/Retrieval	284
drive number	119
Dungeons and Dragons	169
Dungeons Of The Algebra	
Dragons	177, 186
Dunzhin	173

E

Early Games For Young Children	181
Early Games Music	191

Easy Mail	241
Easy Script	209, 215, 219, 221, 241
edit mode	212
editing	54
editor	272
80 column display	224, 296
Einstein Corporation	190
electronic conferences	286
Electronic Information	
Exchange System	289
electronic mail	286
embedded format commands	216
embedded format instructions	216
END	141
endless loop	73
English Monster	180
English programs	193
Epyx Software	170
execute	70
exponentiation	60
external memory device	32

F

Facemaker	183
fields	238
FIG	274
files	238
filing cabinet	237
Flex-File 2.1	242
Flight Simulator	176
floppy disk	33
floppy disk drive	32, 37
flowchart	252
Flower Power	185
footer	218
footnoting	222
FOR	82
FOR-NEXT loop	81
foreign languages	193
form letters	21, 240
format the disk	110
formula	227, 229
Forth	272

forward	195	high-level language	95
fraction fever	186	high-level languages	269
Frankston, Bob	226	history programs	193
Frogger	167, 174	home	51, 196
Ft. Apocalypse	177	household helpers	207
function keys	55, 134		
functions	61		

G

game lines	289
game paddles	33
games	160
GasCalc	232
Generic BASIC	153
geography programs	193
GET	149
GET#	152
global search and replace	212
Gortek and The Microchips	192
GOSUB	138
GOTO	73, 138
GOTO statement	259
Grand Master	177
Grandfather's Clock	129
graphic characters	53
graphics	40, 76, 254
Gridrunner	171

H

hacker	25
Handik Software	226, 234, 241
Hangman	178
hard copies	39, 98
hardware	33
Hawkeye Pierce	13
header	111, 218
help screens	223
HESmodem	281
HESware	171, 191, 220, 236, 274
hexadecimal	271
Hey Diddle Diddle	183

I

ID number	115, 118
IF	289, 230, 243
IF THEN ELSE	230, 235, 264
IF THEN statements	89
Infocom	172
information utilities	23, 282
InfoWorld	47
initializing a disk	120
input	13, 14, 80, 149
input statements	80
input#	152
insert	54
INST/DEL	54
installing	45
insurance	47
INT	142
Intentional Educations	221
interface	41
interger variables	79

J

Jawbreaker	174
joystick	33, 41, 200
Jumpman	170

K

K	31
keyboard	30, 50
keywords	238
Kindercomp	183
KMMM Pascal	261, 266
Koalapad Touch Tablet	300

Kussmaul Encyclopedia 284

L

label 227, 228
Le Mans 178
Learning With Leeper 182
LEFT\$ 145
LET 63, 79
Letter Man 190
letter-quality printer
light pens 33
LISP 194
list 72
list-processing 194, 200
listing your program 72
listings of BASIC programs 136
load 97, 104, 106, 107, 113
loading a program from disk 113
loading from tape 104
log 107
logical file 117
logical file number 117, 126, 152
LOGO 19, 193, 194, 252, 258, 275
loop structure 73

M

M*A*S*H 13
machine language 95, 133, 257
magazines 136
magnetic field 108
Mail-Merge 214
Mailing Label 21, 240
mailing list programs 241
Master Mind 178
Mastertype 189
mathematics programs 193
matrix 148
memory 61
microcomputer 15
microprocessor 30
Microsoft Multiplan 236

MID\$ 145
Midnite Magic 176
Midwest Micro Associates 205
mileage 232
Million-Dollar Blackjack 173
Mindstorms 258
minimal BASIC 153
Mirage Concepts 224, 243
Mirage Database Manager 243
Mirage Word Processor 224
MIT 172, 194
mnemonic codes 270
modem 33, 41, 278, 280
Modula 2 267
monitor 24, 39
Monkey Math 185
monochrome 39
Moon Shuttle 170
Moondust 169
Mother Goose 67
Mr. Cool 174
Multiplan 236
Murder By The Dozen 168
Music Composer 202
Music Machine 201
Music Theory 193
Musicomp 203

N

Name Machine 238, 239
National Institute For Occupational
Safety And Health 45
nested loops 80
networks 289
NEW 68, 76
newing the disk 111
news 284
NEXT 82
Night Mission 176
non-volatile 32
numeric keypad 179, 298
numeric mode 219
numeric variables 79

O

OMNI Unlimited Computer	
Marketing Services	203
OMNIspell	220
OMNIwriter	220
on-line	283
ON . . . GOSUB	143
ON . . . GOTO	144, 156
on/off switch	44
open	111, 116, 119, 121, 126, 152
operating system	95, 266, 272
OR statement	90
output	13, 14, 148

P

p-system	266
Pac-Man	16, 167, 169
Paint Magic	203
PAL	254, 271
Papert, Seymour	193, 258
parameter	146
parameters	199
parti	286
Pascal	19, 256, 260, 268
Pascal, Blaise	260
PEACENET	289
PEEK	145, 157
PET	135, 157, 178, 292
PETspeed	258
PI	60
Piaget, Jean	193
Pilot	19, 274
pinball	167
Pinball Math	185
Pipes	170, 190
Pitfalls	24
Plato	159
Pogo Joe	173
POKE	61, 66, 157
Politics, The	207
Pong-machines	160
Pooyan	170

postfix	274
Power-64	253
power supply	44, 46
PractiCalc and PS	235
preschoolers	180
PRG	116
Primary Math Tutor	184
PRINT	68, 111, 154
PRINT#	118, 120, 121, 152
printer	33, 39, 45, 126
Professional Blackjack	173
Pro-Line Software	253
problem solving	248
procedure	197
procedures	199, 264
Professional Software	219
program	14, 67
program design	248
program file	101
Programs For The Very Young	180
proofreading programs	208
PS	236
psuedocode	252
public domain	178, 184, 193
public domain software	135
public libraries	179

Q

Q-Bert	174
Quick Brown Fox	224
Quick Brown Fox, Inc.	224
Quizagon	168
QWERTY keyboard	50

R

Random Access Device	37
Random Access Memory	30
random number	142
Read Only Memory	32
read statements	150
read/write head	37

reading, writing, and arithmetic	184
recipe file	26
records	238
recreational and educational	
software	159
recursion	199
relative values	233
REM statements	100
remarks	100
renaissance	178
renaming a disk file	120
repeat	196
replicate a formula	233
Repton	175
research assistant	2.0 241
reset	150
restore	55, 62
return	55, 138
Reverse Polish Notation	274
reverse video	52, 76
reviews	131
RF modulator	38
Richvale Communications	223
right	195
right justification	216
RIGHT\$	145
RND	142
ROM	32, 95
ROM cartridge	32, 97
RUN	69
run/stop	55, 62, 73, 107
running your program	69

S

Sammy Lightfoot	174
SAVE	98, 103, 112
Save New York	170
saving a program	101
saving on tape	103
saving the program on disk	112
SB-80	296
Scarborough Systems, Inc.	189
science programs	193

scratch	119
screen color code	62
screen editing	256
screenplay	173
Script 64	223
scrolls	51, 216
search	236
sectors	37
semicolon	75
sequence	73
sequential access	36
sequential access device	37
sequentially	33
Serpentine	167
Shamus	177
shopping	285
SID	61, 98
Sierra On-Line	174, 182
sign language	193
silicon	15
silicon chip	14, 30
Simon's BASIC	256
Sirius Software	175, 189
smooth scrolling	281
Snake Byte	175
Softsync, Inc.	202
software	33
software buying	130
Softwave	185
sort	236, 237
sound	200, 254
sound effect program	100, 128, 160
sound interface device	61, 99
Source, The	282, 286
Space Rescue	191
Space War	160
SPC	75
spreadsheets	226
speech synthesizer	33, 278
Spelling Checker	208
Spellright Plus	220
spikes	44
Spinnaker Software	176, 183, 186
spreadsheet program	21
sprite	149

sprite editors	255	Terminal Software	281
sprites	200	textbook BASIC	153
Squish 'Em	175	The Complete Personal	
standard input device	30	Accountant	245
standard output device	30	The Einstein Memory Trainer	190
Star Trek	178	The Home Accountant	245
Starcross	172	The Name Machine	213
statement	69	Through The Looking Glass	29
statement numbers	69	TI\$	63
Status	153	Timesharing	288
stepwise refinement	249	Timeworks Personal Computer	
stereo system	33	Software	192
stop	141	Timeworks, Inc.	177, 186, 242
strategy games	163	timing mechanism	86
string	69, 156	Tooth Invaders	180
string array	148	Toronto PET User Group	135
string functions	145	TORPET	193
string variables	79	TOTL 2.6	222
SubLOGIC, Inc.	176	TOTL Label 2.6	222, 241
subprograms	138, 264	TOTL Software	241
subroutines	138, 265	TOTL Text	222
subscript	146	Touch Tablet	300
sum	230	Touch Typing Tutor	189
Super Expander 64	255	tracks	
Supermon 64.V1	272	Trashman	170
Superterm	282	TRS-80	158
Survivor	177	true decoders	40
Suspended	172	TSI Maze Man	168
switching modes	53	Turtle Graphics	194, 274
Synapse Software	176	TV set	30
syntax error	52, 56, 57, 70	Twain, Mark	97
SYS	144, 157	Type Attack	189
		Type Right	188
		typing	21, 208
		typing tutorials	188

T

tab	75
table	148
Tamarack Software	274
tape	102
Taylormade Software	185, 189, 193
technology programs	193
telecommunication	23, 279
Telegard	166
Temple of Apshai	170

U

U.S. Department of Defense	267
UCSD Pascal	266
UMI (United Microwave	
Industries, Inc.)	177
upgrades	132
upper-case/graphic mode	53

upper/lower-case mode	53
user group	23, 43, 136
user guide	297
USR	144, 157
utilities	135
utility programs	253

V

validating a disk	120
value	63
Vanilla Pilot	274
variable	62
variable names	62
variables	78
verify	104, 197, 113
VIC	192
VIC-20	158
VIC/64 relay cartridge	307
VICmodem	281
video games	16, 160
video support package	255
Videotex	290
Viewtron	291
Visible Solar System	191
Visicalc	226
voice synthesizer	305
voice-recognition	33
volatile	32

W

Wall Street	177, 192
Warriors of Ras	173
Wepeco Software	187
Whole PET Catalog	135
Wilserv Industries	266
Winchester Disk Drive	38
windows	235
wire services	283
Wirth	267
Wirth, Niklaus	260
Witness	172

word machine	209, 210, 222
word procesor	21, 208
word processing	208
word processing station	208
Word Race	187
Word Wrap	212
Wordpro	220
Wordpro 3+	219
words	30
Work, Henry Clay	129
write-permit tabs	108
write-protect notch	110, 125

Z

Z-80 Video Pak	296
Zork	172

*** BASIC KEYWORDS ***

Notes

FUNCTIONS:

ABS
ATN
CHR\$
COS
EXP
FN
FRE
INT
LEFT\$
LEN
LOG
MID\$
PEEK
POS
RIGHT\$
RND
SGN
SIN
SPC
SQR
STATUS
STR\$
TAB
TAN
TI
TI\$
USR
VAL

COMMANDS:

CONT
LIST
LOAD
NEW
RUN
SAVE
VERIFY

STATEMENTS:

CLOSE
CLR
CMD
DATA
DEF
DIM
END
FOR
GET
GET#
GOSUB
GOTO
IF
INPUT
INPUT#
LET
NEXT
ON
POKE
PRINT
PRINT#
READ
REM
RESTORE
RETURN
STEP
STOP
SYS
THEN
TO
WAIT

(TURN ON DISK BEFORE COMPUTER)

TO LOAD AND RUN A PROGRAM:

Type LOAD "Programname",8
Type RUN

OR (first program in directory)

Type LOAD "*,8

OR (for some canned programs)

Type LOAD "Programname",8,1
or LOAD "*,8,1

TO VIEW DISK DIRECTORY:

Type LOAD "\$",8
Type LIST

TO SAVE A PROGRAM:

Type SAVE "programname",8
(DISK MUST BE FORMATTED)

TO FORMAT A DISK:

Type OPEN 15,8,15
Type PRINT#15, "NEW:diskname,id"
Type CLOSE 15

** USING THE DATASETTE **

TO LOAD & RUN A PROGRAM:

Type LOAD "programname"
Press PLAY on Datasette
Press **C** when found
(repeat until loaded)
Press STOP on Datasette
Type RUN

OR, if first program on tape

Press SHIFT RUN
Press PLAY on Datasette
Press STOP on Datasette
when it stops

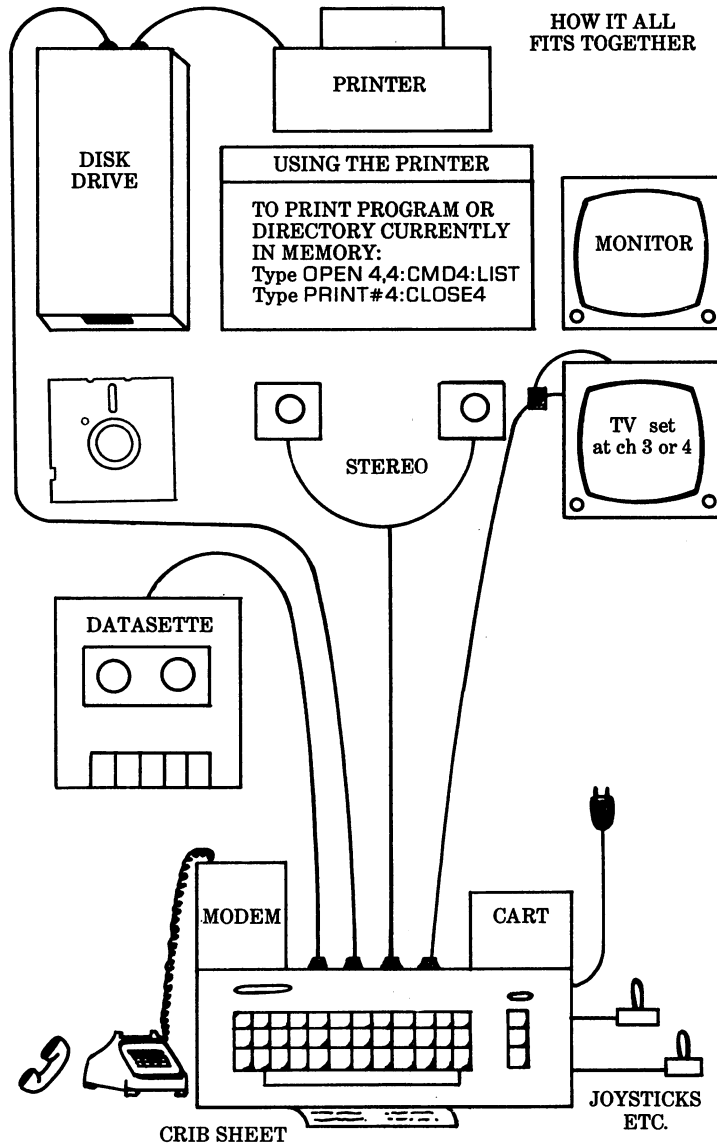
TO SAVE A PROGRAM:

Type SAVE "programname"
Press PLAY REC on Datasette
Press STOP on Datasette
when it stops

** USING THE MODEM **

Plug in modem
Switch modem to 0
Turn on computer
LOAD & RUN terminal prog
Plug phone handset cord
into modem
Dial phone number
Respond to screen prompts

Keep this sheet under your computer
for quick reference.



** CONTROLLING THE SCREEN **

TO CHANGE MODES (UPPER/LOWER CASE <> UPPER/GRAPHICS)
Press SHIFT **C**

TO CHANGE BACKGROUND COLOR:

Type POKE 53281, num

TO CHANGE BORDER COLOR:

Type POKE 53280, num

TO CHANGE CHAR COLOR:

Press key combination shown:

COLOR * num * key comb.

black	*	0	*	CTRL	1
white	*	1	*	CTRL	2
red	*	2	*	CTRL	3
cyan	*	3	*	CTRL	4
purple	*	4	*	CTRL	5
green	*	5	*	CTRL	6
blue	*	6	*	CTRL	7
yellow	*	7	*	CTRL	8
orange	*	8	*	C	1
brown	*	9	*	C	2
lt red	*	10	*	C	3
gray 1	*	11	*	C	4
gray 2	*	12	*	C	5
lt grn	*	13	*	C	6
lt blu	*	14	*	C	7
gray 3	*	15	*	C	8

TO CLEAR SCREEN:

Press SHIFT CLR

** USING A CARTRIDGE **

Turn computer off before
inserting or removing cartridge

** KEYBOARD REMINDERS **

Press RETURN after typing or
editing each line
TO DELETE chars, press DEL.
TO INSERT, press SHIFT INST
USE CRSR keys for editing

TO STOP A RUNNING PROGRAM (OR GET A STUCK PROGRAM UNSTUCK)

Press STOP
OR if that doesn't work
Press STOP RESTORE twice
OR as a last resort
Turn computer off and on

TO CLEAR PROGRAM FROM MEMORY

Type NEW

THE COMMODORE 64 HOME COMPANION

The Commodore 64

HOME COMPANION

The book every C-64 owner should have!

You own a Commodore 64. But you're a little unsure of what it can really do for you. You tried reading the manual that came with the computer and that only got you more confused. Well, take heart — for all is not black. DATAMOST to the rescue!

The Commodore 64 Home Companion is actually three books in one. If you are looking for a computer-as-home-appliance instruction manual, this is it. If you want to learn BASIC programming, this is it. And if your desire is to know everything about hardware, software, programming, etc., then this is it! Think of this book as your friend — always ready and willing to help you learn more about your Commodore 64.

- How to buy, hook-up & take care of your C-64.
- "Hands on" lessons (take your C-64 for a drive).
- Commodore BASIC programming.
- Specific programs you can use to make your C-64 do what you want it to.
- Three ways to learn!

ISBN 0-88190-294-2



 **DATAMOST**TM
INC.

20660 Nordhoff St., Chatsworth, CA 91311-6152, (818) 709-1202