micro COMPUTER books

Len Lyons
with Joe Campbell and Herb Moore

Using the Commodore 64™

A Step-by-Step Guide to:

- ☐ Word Processing
- ☐ Telecommunications
- ☐ Database Management
- ☐ Programming and Graphics
- ☐ Financial Spreadsheets
- ☐ Education, Games . . . and More

Using the Commodore 64

Using the Commodore 64

Len Lyons
with Joe Campbell and Herb Moore

ADDISON-WESLEY PUBLISHING COMPANY
Reading, Massachusetts • Menlo Park, California
Wokingham, Berkshire • Amsterdam • Don Mills, Ontario • Sydney

Book design and production: Greg Johnson,
The Book Department, Inc.

Technical art: Paul S. Foti,
Boston Graphics Inc.

ISBN 0–201–05156–7

ABCDEFGHIJ–HA–8987654

Library of Congress Cataloging in Publication Data

Lyons, Leonard.
Using the Commodore 64.

Bibliography: p. 293
Includes index.
1. Commodore 64 (Computer) I. Campbell, Joe, 1944– . II. Moore, Herb, 1944– . III. Title.
QA76.8.C64L95 1984 001.64 84-2847
ISBN 0-201-05156-7

Table of Contents

This book has a modest goal: to help its readers get the most out of the Commodore 64 computer. It is dedicated to the memory of someone whose goal was more noble:

Michael D. Evans

He helped others get the most out of their lives.

Acknowledgments

Although writers usually work alone, they must generally acknowledge the assistance of many others. This book is no exception: Writing it was a lonely task, but it would have been a far more difficult one without the cooperation of those mentioned here.

The first thank-you's go to Herb Moore and Joe Campbell, who contributed Chapters 4 and 8 respectively. I feel fortunate to have had two experienced writers provide chapters in their fields of special expertise. Additional thanks to Joe, for his advice on the concept of this book when it was in the planning stages.

An author could not ask for a more supportive and capable publisher. Thanks to Tom Bell for taking this project on board; to David Miller and Peggy McCauley, who saw it through to completion; and to Janice Byer for her thorough and constructive copy editing.

I am grateful to Dick McKnight, former president of the Commodore Users' Group of the Boston Computer Society, for his careful reading of the manuscript and thoughtful suggestions.

I am deeply grateful as well to the late David A. Rogers for his unerring advice on many aspects of this book. Dave was a Commodore aficionado, whose expertise in all aspects of Commodore hardware and software was (in my experience) unmatched. Dave was the president of the MASSPET Commodore Users' Group and also operated a Commodore electronic bulletin board service that received calls from across the country and Canada. His knowledge of the subject was matched only by his enthusiasm for it.

It takes more than professional help to finish a book. Thankfully, my wife Maxine and our children, Gila and Ami, were able to bear with the author for the difficult months (which included a cross-country move) required to complete the work. For them, I am grateful as always.

Preface

This book is written for people who expect computers to make their lives more productive, more exciting, and more fun. In other words, it's about more than writing computer programs. It also shows you how to use your Commodore 64 as a word processor, financial adviser, filing system, telecommunications center, learning tool, and game machine.

This book also attempts to remedy a well-known shortcoming of the Commodore computer line: the company's inability to provide easy-to-follow, readable users' manuals with their products. Despite Commodore's excellent hardware and software, which are arguably the best dollar value in the home computer market today, Commodore's manuals are conspicuously inadequate. The VIC-1541 disk drive manual, for example, has become notorious for confusing even experienced computer users. And the manual that comes with THE MANAGER, which is perhaps the best filing system you can buy for under $50 for any microcomputer, fails to mention some essential steps in commonplace procedures.

Even a good manual, however, is basically a reference guide to the software's commands. By contrast, in this book the emphasis is on explaining what the software can do for you, and providing exercises, or "tutorials," in using it.

In the introduction I outline the idea of a computer *system*, explaining how its component parts are interrelated. I also emphasize the importance of *applications*, the ways that computers can be used to expand and improve upon your work and play.

Part One of the book is about what you can do with your computer system before purchasing any additional software. This includes using the keyboard, the external "memory" devices (the C2N Datassette cassette tape recorder and the VIC-1541 disk drive), and a printer. There are also chapters that explain what programs are and how to write them; one chapter deals specifically with how to program graphics and sound. If you're a good enough programmer, there's much you can do with the Commodore 64 without purchasing additional software. Becoming *that* good a programmer, however, takes most people a year or more of regular study and practice.

Part Two is designed to help you get some computing into your life right away. These chapters are devoted to word processing, financial management and projections, file management (that is, record-keeping), telecommunications, and games and educational programs. Each chapter explains the advantages of computerizing the tasks involved, and then describes in detail how to use the best software available for the job.

Of course the "best" really means what's most useful for the vast majority of people. For several reasons, I discuss Commodore's own software whenever possible. In 1983, the company released several good software packages in the basic applications areas. Although Commodore had little software to offer early in the 64's life, these new programs are powerful, relatively easy to use, and inexpensive (especially in contrast to comparable software for other home computers). Another virtue of Commodore products is that they are likely to be available and supported by the manufacturer so long as Commodore is in the home computer business.

There are a few instances, however, where we mention other manufacturers' software. The financial management software discussed in Chapter 6, Easy CALC RESULT, was a Swedish product when it was selected. Recently, however, Commodore purchased the right to distribute this program as its own EASY-CALC 64. Thus, you can find this software under both titles. The telecommunications software discussed in Chapter 8 includes both Commodore's own 64 TERM that comes with the VICMODEM, and a program called SMART TERMINAL 64 PLUS, published by a New Haven, Connecticut, company. Since Commodore has no software that can *upload* and *download* (these terms are explained in Chapter 8), it was essential to find reliable software that offers these powerful features.

Chapter 9, on educational software and games, is more of an overview than an in-depth analysis of particular programs. By its nature, this software is easy to use, so it seemed more appropriate to sample the field than to examine single examples in depth. Commodore-related periodicals regularly describe and review educational software and games.

At the end of the book, there are a bibliography of relevant books and periodicals and a glossary, which provides quick def-

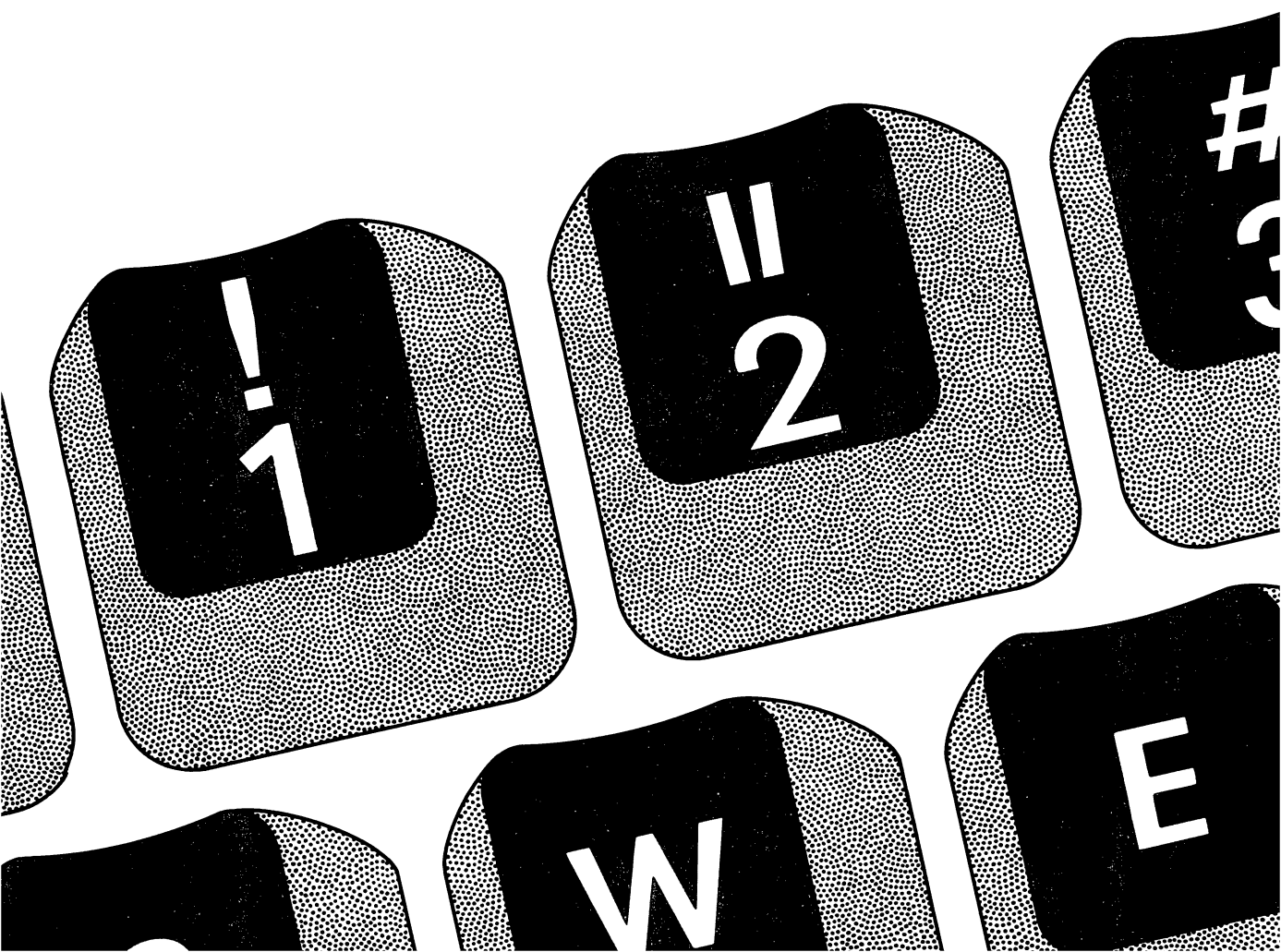
initions of technical terms that users are likely to come across in this book or in other reading.

Appendix A provides instructions on how to type program LISTings that include graphics symbols. Many published programs are printed directly from the screen and show the graphics characters produced by using the SHIFT or COMMODORE keys. Unless you know which combination of keys produces a particular character, it can be very difficult to figure out how to type the program. Appendix B is a list of Commodore users' groups. Joining a users' group is an excellent way to learn about computing. Appendix C contains a list of Commodore electronic bulletin board systems, which — like users' groups — are very worthwhile but sometimes ephemeral.

One exciting thing about the microcomputer field is that it is continually growing and expanding. The opportunities to use the computer in everyday life seem to increase monthly. The Commodore 64's popularity is a good indication that it will continue to attract those who develop and write new software. With an understanding of the computer applications described in the chapters ahead, you should be well prepared to evaluate and take advantage of those opportunities.

Len Lyons
Lexington, Massachusetts

INTRODUCTION



Note to the Reader

The most logical way to read the following chapters is consecutively. But you may want to use word processing, financial planning, or telecommunications software *before* working through Chapter 2 (on the Datassette recorder and VIC-1541 disk drive), Chapter 3 (on programming), or Chapter 4 (on programming graphics and sound).

Feel free to read the later chapters out of sequence. The chapters in Part 2 have been written with that possibility in mind. They are relatively self-contained, and any mention of material discussed earlier in the book is clearly cross-referenced. There is no need to learn to program the computer before putting it to work in one of the many applications that are the subjects of Chapters 5 through 9.

You should, however, read the introduction and Chapter 1 before going on to any other chapters. These provide a general introduction to the Commodore 64 computer system and reveal my goals and point of view.

What Computing Can Mean to You

Appreciating the Computer *System*

One of the marvelous things about the human mind is that we don't have to understand how it works to use it well. If that weren't so, we'd still be sitting around in caves scratching our heads. We know precious little about how we coordinate our bodies, perform abstract thinking, change our attitudes, and so on. There is probably no theory on any of these subjects that commands universal agreement. Yet even a child accomplishes a great deal with his or her young brain, without even being aware of the brain's existence.

Just like our minds, we can use computers to great advantage and have a lot of fun with them without understanding how they work. That was not always true, though, because computers used to be so sensitive and "stupid" that they needed highly trained personnel to tell them what to do. But no more.

Computers are no match for the human mind in most respects. But they can "process information" more accurately, tirelessly, and infinitely faster than we can. And recently, computers have begun to approach the sophistication of the mind in one respect. We can use them without understanding how they work. As computers become more sophisticated, the number of tasks we can apply them to increases and the amount of computer science we have to know decreases.

In certain respects, the impact of computer technology on our lives is analogous to the effect of automotive technology. For example, you can walk at the rate of a few miles per hour, which may be sufficient to get you around town to the post office, to the market, and home in time for lunch. Driving a car doesn't simply change how fast you can do those chores; it also changes *what* you can do. Traveling in a car at forty or fifty miles per hour, you can mail your letter, shop at the market, carry home enough food so you don't have to shop again for a week, transport a

friend across town on your way to the market, and *still* have time left over to go to the beach or get some other chores out of the way.

New technology doesn't simply increase the speed and efficiency of our work; it gives us the power to do more than we could have imagined doing without it. That is certainly true of computers, especially when your computer is part of a computer system.

Your Commodore 64 computer — we'll abbreviate that to *C-64* from now on — is contained in the frame that houses the keyboard. The frame contains the electronic wiring and microprocessor *chips* (integrated circuits printed on a silicon wafer) that make up the "brains" of the computer.

The computer system, however, includes the other components (or peripheral devices) that work with your computer, such as the VIC-1541 disk drive, the C2N Datassette cassette tape recorder, a TV set (or the Commodore 1701 monitor), a printer, and a modem. In general terms, these components are referred to as the system's *hardware*. (See Figure I–1 on page 8.)

The programs (or sets of instructions) that make the hardware do what you want it to are known as *software*. Some software — for example, the BASIC programming language interpreter — is built into the circuits of the computer. Other software comes in the form of cartridges, tapes, or diskettes that are loaded into the computer. And if you know how to program, you can create your own software by typing programs on the keyboard. Software is an essential part of your computer system; without it, the hardware is no more than an expensive paperweight or a conversation piece.

(This book uses the terms *computer* and *computer system* interchangeably. In sentences like "The computer displays the next step on your TV screen" and "The computer prints out the answer when you press RETURN" I am actually referring to the computer *system*.)

But a computer is different from an automobile and most other kinds of technology you've encountered just because it's programmable. In short, by writing your own programs, you can enable your computer to perform a variety of tasks, probably including tasks you do not yet realize can be done by the computer. You'll start thinking about these possibilities when you

read Chapters 3 and 4, which introduce you to programming and to creating graphics and sound on the computer. With these tools, you can create your own software to help you in business or school, or design your own video games complete with sound effects.

Applications and Automation

In computer jargon, the tasks you use your computer to accomplish are called *applications*, and the programs that tell the computer how to accomplish the tasks are known as *applications software*. Part 2 of this book is devoted to applications software and how to use it effectively to automate your personal and family business tasks.

Automation is a term that was once applicable only to large businesses and industry. Simply defined, automation refers to systems in which machines automatically perform tasks that were previously carried out by people. For example, to make a phone call you no longer have to crank up the phone to get the attention of the operator, who would then insert a phone jack into the proper plug. If calls were still switched manually, the phone company would need to hire half the work force in your town just to handle the switchboard. Today, you simply dial or "type in" the number, and a computer does the switching at an amazing speed.

Personal computers, like the C-64, have brought automation home. The same computer you use to play video games, to learn to program, and to teach your kids how to spell can help you make long-range financial projections, write term papers and letters, and search voluminous databases in minutes for useful information. Of course, you're not going to automate your life to the extent that Ma Bell has automated the phone system or that NASA has programmed its unmanned (*automated*) satellites. But your C-64 can greatly simplify complex, tedious tasks. And equally important — it can put useful, fascinating, and unimagined possibilities at your fingertips.

Here are a few examples. You can probably write a letter pretty quickly with an electric typewriter, but with your computer, you can revise and edit the letter without having to retype it, create modified versions of the original letter to send to several

friends or business contacts, check your spelling automatically (without proofreading), and have all the letters printed error-free, automatically, at a rate many times faster than a first-rate secretary could type it. Chapter 5 shows you how to use your computer in this way — as a *word processor.*

Using a telephone, you can call an acquaintance long distance and find out a detailed travel itinerary for his trip to Europe. Or, if he has time, he can type it out and mail it to you. But if you both have computers and *modems* (a device that sends and receives data over telephone lines), he can send your computer all the information within a couple of minutes (at the most); and you can read it on your computer's screen or print it at your leisure.

With a modem, you can also search databases to receive up-to-the-minute stock quotations, reports on ski conditions, and sports scores; you can search the contents of a library by subject matter without ever leaving your desk — or scan electronic bulletin boards for business or personal reasons. Telecommunications is the subject of Chapter 8.

Do you keep financial records of your monthly mortgage payments and household expenses? Do you know what your basic living costs would be if interest rates drop two percentage points and you refinance, but the price of fuel goes up 5 percent? What would it mean to your financial situation if you deposited the money saved in mortgage payments in an account yielding 10 percent interest? If you're like most people, the difficulty of that calculation would deter you from trying to figure it out. With a good *spreadsheet* program, such as Easy CALC RESULT, your computer will give you the answer to this and similar financial projections in seconds. Easy CALC RESULT is discussed in Chapter 6.

These examples are the tip of the proverbial iceberg. Since you own a C-64 (or you're contemplating buying one), you probably know about the colorful and challenging video games that you can play on your computer. But you might not know that there is educational, as well as entertaining, software that makes use of the C-64's exceptional sound and graphics capabilities. Games and instructional programs are covered in Chapter 9.

Although people haven't yet begun to think of computers as common, everyday household items like refrigerators or wash-

ing machines, there is reason to do so. A computer doesn't keep your food fresh or clean your laundry, but it performs another valuable service: It processes *information*, a broad category that includes numbers, words, sounds, and images. Some household appliances, such as microwave ovens, contain the same type of microelectronics as your computer. The microwave oven has been preprogrammed to have one fixed function. Thus, you may already be using "computerized" appliances in the home without realizing it. There's a big difference, though. The microwave oven has one unalterable program, but your computer is truly programmable — by you.

Although computers have innumerable applications, most people buy a computer with only one or two applications in mind. Oddly, they often fail to investigate all the other things that can be done with the same computer. Returning to our automobile analogy, that would be like buying a car to help you go to the market and the bank, *and never using it to go anywhere else!*

Naturally, if you own a car, you'll also use it to go to the beach, visit friends, take the kids to school, transport furniture, or anything else you can think of doing with it. The logic of this approach is obvious enough with cars, but then cars are not nearly as subtle, or as new, as computers. Cars are in effect extensions of our legs — they can take us farther and get us there faster. But computers are extensions of our minds, fingers, eyes, ears, memories, and our imaginations. The limits of what they can enable us to do have yet to be reached.

Fundamentally, the computer is an information processor consisting of a memory and circuitry that processes the information in memory. The peripherals — the components that are not part of the computer itself — are *input/output* devices. They are responsible for entering information into the computer's memory (INPUT) or receiving information from the computer (OUTPUT). Some peripherals — the Datassette recorder, disk drive, and modem — can perform both input and output functions.

The diagram on page 8 outlines the flow of information in a computer system with arrows representing input and output. The captions provide a brief indication of how each peripheral functions in the system, but don't expect to understand these devices fully until you've used them.

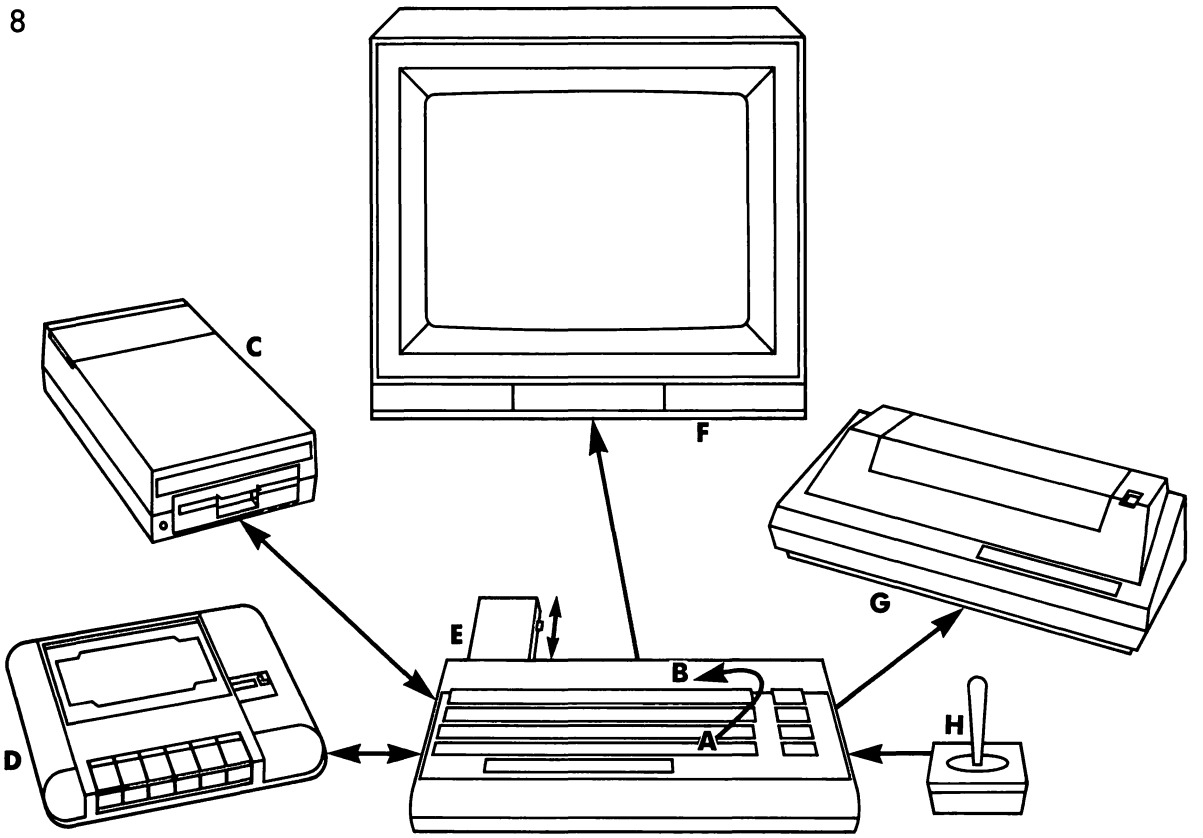


FIGURE I–1: Commodore 64 System.

A. *Keyboard:* enters data and commands into the computer's memory. (INPUT)

B. *Computer:* consists of memory to retain information and "integrated circuits" that process the information.

C. *Disk drive:* permanently stores information from the computer's memory on floppy diskettes and loads information back into the computer's memory as needed. (INPUT/OUTPUT)

D. *Datassette recorder:* same as the disk drive, except that it stores the information on tape. The recorder stores less information than the disk drive and works more slowly. (INPUT/OUTPUT)

E. *Modem:* transmits information from the computer's memory over telephone lines; or receives information over the phone lines and enters it into memory. (INPUT/OUTPUT)

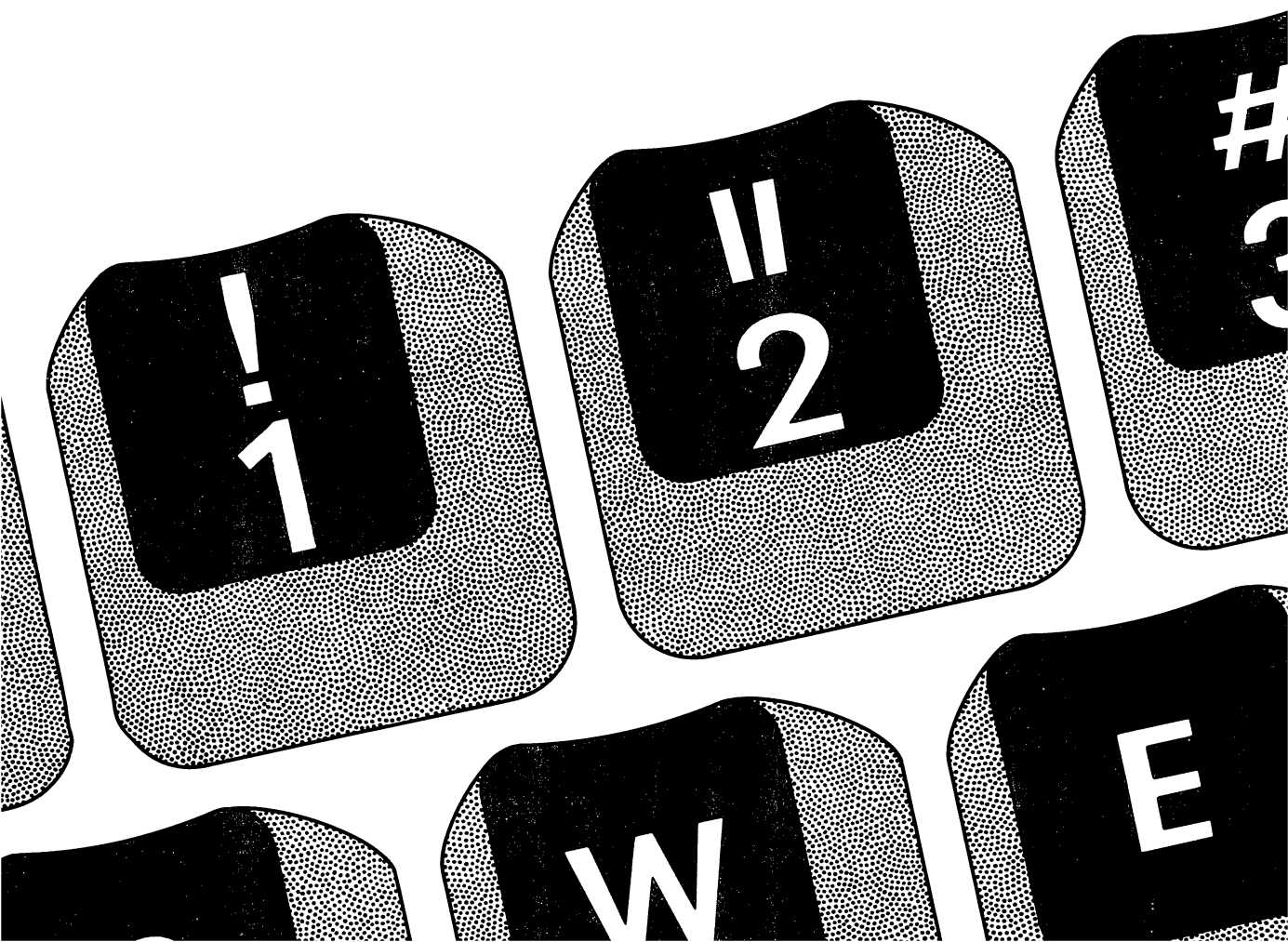
F. *TV or monitor:* displays characters or graphics from, and as instructed by, the computer's memory. It is the computer's visual output device. The Commodore 1701 monitor offers a sharper picture — with "higher resolution" — than a TV. (OUTPUT)

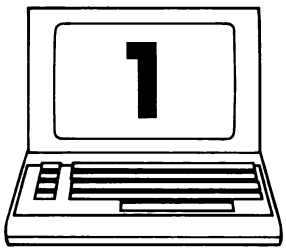
G. *Printer:* prints information from the computer's memory or from the disk drive or Datassette recorder via the computer's memory. (OUTPUT)

H. *Joystick:* sends instructions to the computer that determine the movement of game pieces on the screen. (INPUT)

1

THE COMPUTER SYSTEM AND HOW TO USE IT





Getting Acquainted with the Keyboard

This book takes a hands-on approach to learning how a computer works. In other words, you'll learn by doing. As you use your computer, you'll begin to understand how it behaves and how to *make it behave*. In the past, people who didn't have technical backgrounds were afraid of computers, perhaps because these "mysterious" machines have for so long been associated with science. But most people no longer think of computers as solely for scientific use. The C-64, and many other personal computers, are designed for use in the home, in small businesses, in the grade school classroom, and on the farm. You can make good use of your computer without any knowledge of electronics or even programming — in the same way that you can drive your car without knowing how an internal combustion engine works.

Sometimes, it helps to know what is going on behind the scenes, in the computer's memory, on the floppy diskette where your text, programs, and data are stored, and so on. As these crucial points arise, you'll find diagrams and explanations designed to give you a clear conceptual framework that removes the mystery from computing. But our starting point is always *using* the computer.

The time to start is *now*!

Booting Up

In computer jargon, turning on the computer is known as booting up. *Booting up* comes from the expression "pulling yourself up by your own bootstraps," which is what the computer seems to be doing when it starts up. Although this book avoids jargon when-

ever possible, booting up is an inescapable expression, so you may as well get comfortable with it. Basically, you can think of booting up as *starting* up your system.

The instructions for connecting the C-64 to your TV, tape recorder or disk drive, and printer are clearly stated in the Commodore 64 *User's Guide* (pp. 3–10), the manual that you found in the box with the computer. Follow the instructions carefully. Above all, don't *force* any of the plugs into their ports, because they should fit easily; also, remember to turn on the computer *last*. The list below summarizes the order of the steps for booting up the system. Since the manual already shows you how to connect the components, those instructions aren't repeated here. This chapter is devoted to *using* the computer and understanding something about how it works.

The exercises in this chapter require only steps 4 and 5, so if you find that steps 1 through 3 don't apply to you, don't give up yet.

1. Connect your disk drive or Datassette recorder to the computer and turn on the drive or recorder.
2. Connect the printer to the computer and turn on the printer.
3. Gently insert the modem into the user port in the back of the C-64.
4. Connect the TV (or monitor) to the computer and turn the TV on.
5. Turn on the computer — last!

When you boot up successfully, the information in Figure 1–1 appears on your TV screen or monitor.

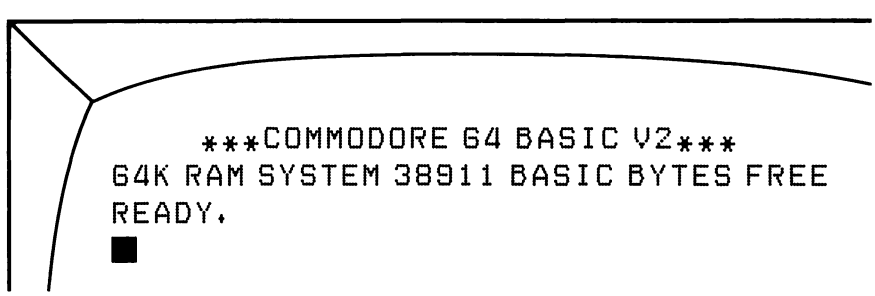


Figure 1–1: The Initial Screen.

The first line on the screen tells you that the Commodore 64 understands the BASIC programming language. BASIC is a code that the C-64, along with many other computers, is built to understand. If you know the code and use it properly, you can instruct the computer to perform as you wish. "V2" stands for Version 2.0, the version of BASIC used by the C-64. There are other versions of BASIC, but they're all very similar and their differences, at this point, are unimportant. You'll find some sample BASIC programs at the end of this chapter, and Chapter 4 provides an introduction to BASIC programming.

(Remember, though, that programming is only one of many things you can do with your computer. You can become an expert in word processing, telecommunications, and you can use the C-64 for financial planning, record-keeping, and games *without ever writing a program.*)

RAM and ROM Memory

The second line of the screen (Figure 1–1) refers to the C-64's *RAM* — or Random Access Memory. You probably know already that your computer has a memory, but it's important to realize that it actually has two different kinds of memory. The second kind is called *ROM* — or Read-Only Memory. The essential difference between them is that you can put information into the computer's RAM but not into ROM. That's what "Read-Only" means: The computer can read what's stored in ROM but can't alter or erase its contents. The computer, however, can "write" information into its RAM, read it back later on, and even erase what it writes there.

Let's illustrate the two types of memory with a simple example: telling the computer to add two numbers and to display the result on the TV screen.

Using the C-64's keyboard, type the following line:

```
PRINT 1 + 1
```

Then press the key marked RETURN. The number 2 should appear on the next line. Now, let's look more closely at each component of this line.

PRINT is a key word in the BASIC programming language; it tells the computer to *display* something on the TV screen.

The plus sign is a BASIC symbol that means just what it does in everyday arithmetic: addition. So the statement you typed means "display on the screen the sum of 1 and 1."

RETURN

Then you pressed RETURN. This is a special key that has several uses. Its two major uses are (1) telling the computer to enter information into RAM and (2) telling the computer to *execute* your instructions. The word RETURN is a holdover from typewriter keyboards, which have a key called CARRIAGE RETURN. Since the computer doesn't have a carriage, labeling a computer key RETURN is a bit of a misnomer. On some computers, it's called the ENTER key (because it enters data in memory), and somebody once suggested calling it the DO IT! key. But Commodore has stuck with the old standby.

Many Happy RETURNs

The RETURN key is used so frequently that it's worth adopting a special symbol to indicate its use. From now on: <RETURN> means "press the RETURN key."

Were you wondering how the computer knew what to do with PRINT, +, and RETURN? The answer is that their meanings are stored in ROM. The computer is *pre*programmed to interpret them in a way determined by the C-64's designers. ROM contains a set of instructions and information that is part of the computer.

But how did the computer know to display the sum of 1 and 1. Any other numbers might have been chosen instead. You could have told it to display the result of subtracting 6,789.04 from 28,409.67, or to subtract 1 from 1. RAM, in short, is "blank" memory or "work space" that exists so that you can provide the computer with information and tell the computer what to do with the information.

You might picture RAM as a configuration of pigeonholes, (like the cubbyholes used to sort mail). Each pigeonhole, or mailbox, represents one unit (or *byte*) of memory. In fact each of these mailboxes is labeled with a distinct memory *address*. When you type on the keyboard, each character (or *byte*) of information is stored at a particular memory address in RAM.

The computer, thus, uses both RAM and ROM to process your simplest request. But it also uses something more: a microprocessor — also known as the computer's Central Processing Unit (or CPU).

The CPU: The Brains Behind the System

Imagine looking at a busy train yard from the air. You'd see strings of cars traveling out of the yard and others entering the yard. Some trains would be switching tracks and moving in new directions; others would be shut down and left in temporary storage. Here and there you'd see engines decoupling from their freight cars and other engines coupling with new cars. Thinking about all this activity, you'd eventually come to the conclusion that somewhere, somebody is overseeing and coordinating the movements. Otherwise, you'd wind up with a yard full of collisions and disabling train wrecks.

In a train yard, the mastermind that coordinates the movement of trains is usually a group of people situated in the switching tower. In a computer, those "trains" are streams of data going into RAM, out to the TV screen, to the printer, and so on at a rate calculated in *nano*seconds (billionths of a second).

The brains behind all this activity is a microprocessor *chip* — an extensive amount of electrical circuitry imprinted on a silicon wafer about the size of your thumbnail. Since this chip performs the arithmetic and logic operations of the computer, it is called the CPU, or Central Processing Unit. The CPU is what makes the computer "intelligent."

The C-64's microprocessor is known as a *6510* chip. It's a slightly more versatile version of a chip that is used in many other microcomputers, most notably the Apple, Atari, and VIC-20 computers. Included in the C-64's CPU are also the 6567 Video Interface Chip (VIC), which controls the computer's output to the screen, and the 6581 Sound Interface Device (SID), which controls music and sound effects. Within the CPU is the C-64's operating system. Part of the operating system known as the KERNAL controls input and output.

At present, there are two things to keep in mind about the operating system and the KERNAL. First, the computer won't work without them. Second, you don't have to know anything *else* about them until you're ready to do some serious programming. The CPU is meant to remain behind the scenes, coordinating whatever the computer has to do to execute your commands.

Now that you're aware of the CPU and its role, let's return to the more practical issue of the computer's memory.

Measuring the Computer's Memory

We describe our own memories as "long" or "short," "good" or "bad," but we don't have any precise way of measuring them. Computers are much easier to understand than people, and the size of their memories can be measured in *bytes*. Each byte of memory is equivalent to one character (a letter, number, symbol, or blank space).

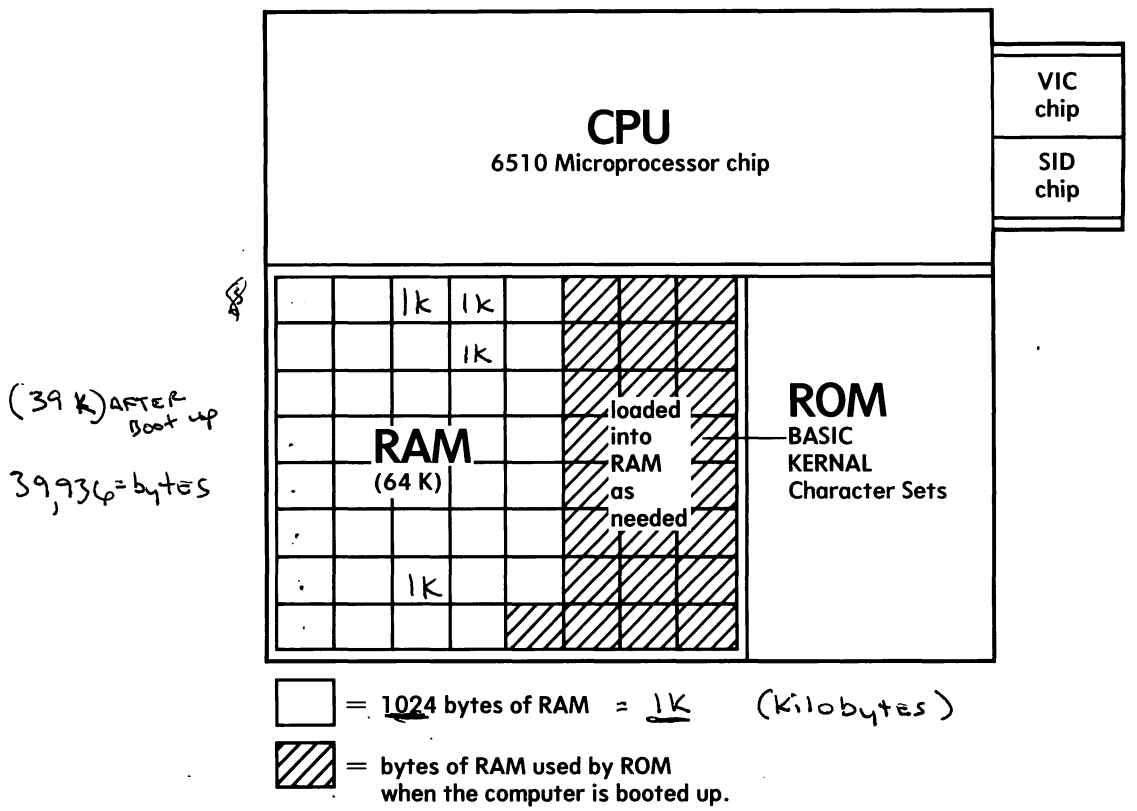
Look again at the second line of the screen in Figure 1–1:

```
64K RAM SYSTEM 38911 BASIC BYTES FREE
```

This line gives you two measurements of RAM memory: the total size of RAM, and the number of bytes you can use for writing programs in BASIC.

As you probably know, your Commodore computer is called the 64 because it has 64K (K is the abbreviation for "kilobytes") or approximately sixty-four thousand bytes of memory. (Since 1K is actually 1,024 bytes, the C-64 has exactly 65,536 bytes of memory.)

The reason that there are only 38,911 bytes available for your programs is because the C-64's operating system and the BASIC language itself use up some of those mailboxes. In short, when you boot up the computer, it loads the BASIC language interpreter (stored in ROM) into RAM. (See Figure 1–2 on page 16.) You can *un*load BASIC — remove it from RAM — to give yourself more free memory. But unless you become a very sophisticated programmer or process large amounts of data, that won't be necessary.



A more realistic depiction of RAM would include 65,536 "mailboxes" or "pigeon-holes" of memory. To simplify the Figure, each mailbox represents 1,024 (or 1K) distinct memory "addresses."

Figure 1–2: Inside the C-64.

READY and the Cursor (■)

SHIFT
↑ CSR ↓
←CSR→

On the C-64's start-up screen, represented by Figure 1–1, the illuminated, light-blue square known as the *cursor* appears just below the word READY. READY is a *prompt*; it's the computer's way of saying, "it's your turn — tell me what to do." The cursor indicates where the next character you type will appear. The word "cursor" comes from the Latin *cursus*, meaning "runner," and, true to the word's derivation, this little patch of light runs all over the screen at your command.

Practice the cursor movements below until they feel comfortable. (SHIFT-↑CRSR↓ means press the SHIFT key and ↑CRSR↓ key *simultaneously*. A hyphen between two keys *always* means "press the keys simultaneously.")

↑CRSR↓	moves cursor down
SHIFT-↑CRSR↓	moves cursor up
←CRSR→	moves cursor to the right
SHIFT-←CRSR→	moves cursor to the left

You may have noticed that when a cursor key is held down, the cursor continues to move until the key is released. Here's a key that moves the cursor directly to the upper left-hand corner of the screen, or the home position.

CLR/HOME moves the cursor to the home position

Like most keys, CLR/HOME functions differently when it's pressed simultaneously with the SHIFT key. Along with bringing the cursor home, it activates the CLeaR function, erasing everything on the screen:

SHIFT-CLR/HOME clears the screen and sends the cursor to the home position

When you press SHIFT-CLR/HOME, all the information on the screen (except for the cursor) disappears.

Now that you have some practice controlling the cursor's position on the screen, let's get to know the keyboard in earnest.

Shake Hands with Your Keyboard

The keyboard is the principal means for communicating with the computer. There are other means for telling the computer your wishes, such as joysticks and light pens, but the most common and useful work your computer will do involves the keyboard.

The C-64's keyboard resembles a typewriter's, but there are major differences. One difference is that the computer has two *character sets*. A character set is just what you'd think — a set of characters (letters, numbers, punctuation marks, and a large complement of special symbols). Typewriters have only one char-

acter set. Typing produces lower-case letters and the lower symbols on the key faces; pressing the SHIFT key with another key produces an upper-case (capital) letter or the upper symbol on the key face.

The computer keyboard also works in the "typewriter mode," but, when booted up, it starts out in the "graphics mode." You used this mode when you typed PRINT 1 + 1. You probably noticed that upper-case letters appeared, even without pressing the SHIFT key. So what happens when you do press the SHIFT key?

Try this experiment. Type the words SHIFT KEY, press the SPACE BAR twice, and then type the word PRINT *with the SHIFT key held down*. Press the SPACE BAR once after each letter key so the characters on the screen will be easier to examine. You should see the following line on your screen:

SHIFT

SHIFT KEY

These strange symbols are graphics characters used in creating pictures, screen designs, and game pieces.

Now, press the RETURN key twice. Type COMMODORE KEY, press the SPACE BAR twice, and then type PRINT *with the Commodore logo key* C= *held down*. The second line on your screen now looks like this:

C=

COMMODORE KEY 

As you can see, the graphics mode itself has two sets of graphics characters — one produced by letters typed with the SHIFT key; the other with letters typed with the COMMODORE key.

Look closely at the front of the P, R, I, N, and T keys. Compare the design on the right side of each key to what you see on the screen, and do the same for the design on the left side of each key.

The SHIFT key produces the graphics characters on the *right* of the key front; the COMMODORE key produces the graphics characters on the *left*.

To switch from the graphics mode to the typewriter mode (that is, to change character sets), press:

SHIFT C=

SHIFT-C=

As you press these keys simultaneously, watch the screen change from the graphics to the typewriter mode. Notice also that the letters pressed with the COMMODORE key still produce graphics characters. Feel free to experiment with these two modes. No matter what you type on the keyboard, you can't do any damage to the computer. Type plenty of graphics characters to get an idea of the assortment available. In Chapter 5, you'll get to see *all* the graphics characters at once — and in color.

After experimenting, clear the screen with SHIFT-CLR.

The Space Bar

Pressing the SPACE BAR moves the cursor to the right and creates a space at the cursor position. Although invisible, a space is considered a character. Later in this chapter, in fact, you'll be using spaces to display colors on the screen. Moving a CRSR key also leaves a space on the screen, but there's a major difference between them. SPACE BAR *replaces* any character at the cursor position with a space; but a CRSR key simply passes over an existing character.

Demonstrate the difference by typing PRINT on the screen. First move the cursor over PRINT with the CRSR key; then move it across PRINT with the SPACE BAR. Using the SPACE BAR *replaces* PRINT with five spaces.

Leave your finger against the SPACE BAR as the cursor moves across the line. There are 40 columns (or character-positions) on each line. When the cursor reaches the end, it *wraps around* to the next line. You don't have to hit the RETURN key as on a typewriter. The RETURN key (as noted earlier) is used to enter information into RAM or to execute your instructions.

Using the INST/DEL Key to Correct Typing Errors

Move the cursor backward by pressing the INST/DEL key. This key *deletes* the character to the immediate left of the cursor, even if the character is a space made with the space bar. For practice,

Special Keys

Use this chart for reference as you work through the book. Don't expect to understand all the summaries at first reading. The use of certain keys varies with the context. For example, in a BASIC program 3 ↑ 2 means "three squared," but when using the Wedge (see Chapter 2) ↑ PROGRAM means "load a program from a diskette and run it immediately."

The term *control key* means "a key used to control the operation of a program."

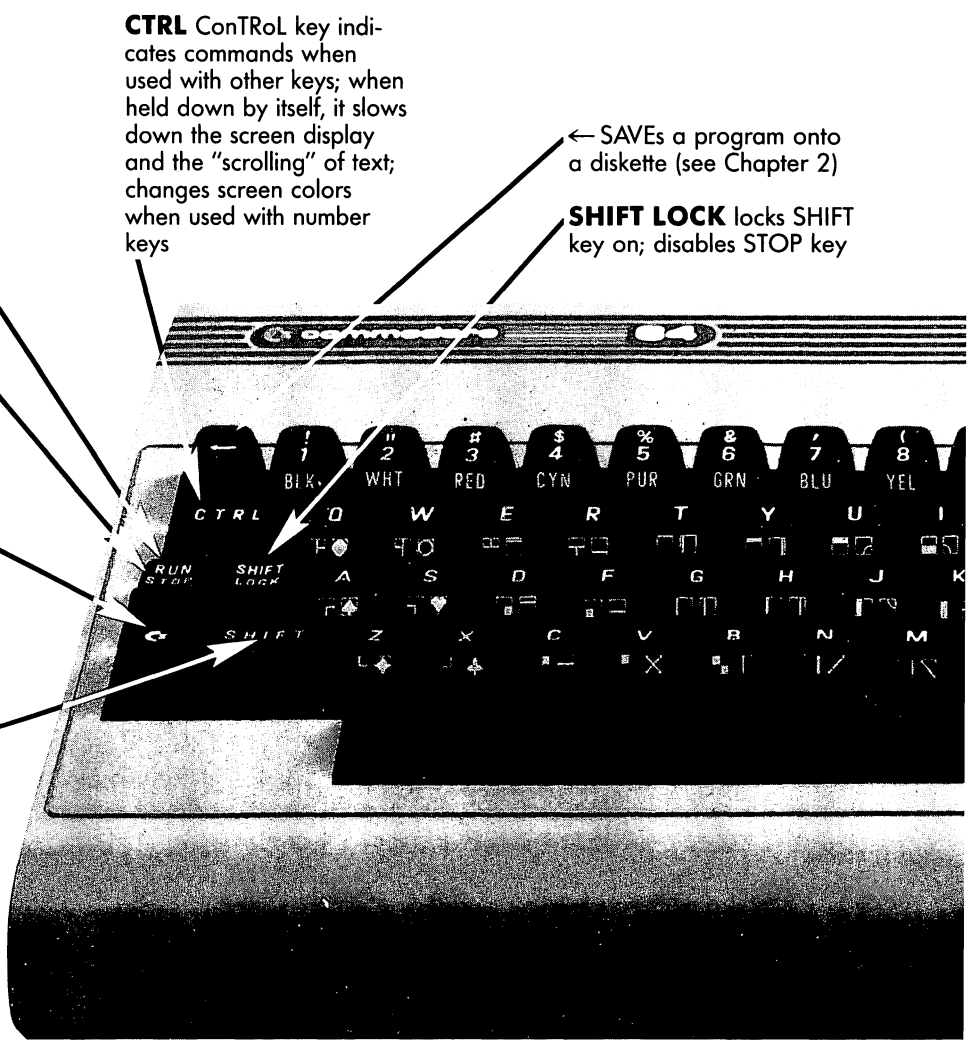


Figure 1–3: Getting to Know the Keyboard.

SHIFT-CLR clears the screen (without erasing memory) and returns the cursor home

HOME returns cursor to line 1, column 1 (upper left-hand corner) of the screen

↑ LOADs and RUNs a program from a diskette; indicates exponentiation in BASIC; and signifies data input when used with certain software

* signifies multiplication in BASIC

SHIFT-INST inserts a space at cursor position

@ signifies commands (copy a file, erase a file, etc.) to the disk drive when using the Wedge; checks Error Status of the disk drive; indicates SAVE-and-replace a program onto diskette in BASIC; a control key with some software

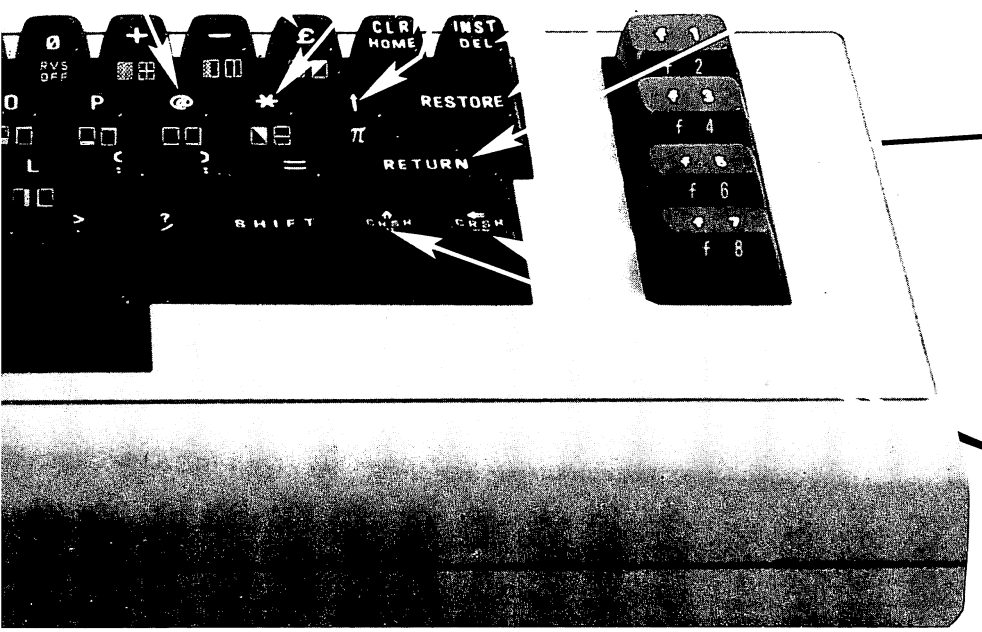
£ initiates specific actions in some software (i.e., a control key); a useful graphics symbol

DEL erases character to the immediate left of the cursor

STOP-RESTORE breaks a program (if one is running) and clears the screen without erasing the program in memory

RESTORE does nothing by itself

RETURN tells the computer to enter information into memory, or to execute a command; also returns the cursor to column 1 (left side) of the screen and moves the cursor down one line



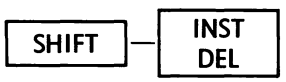
F1–F8 function keys used to implement commands or as control key in certain software; the keys are programmable in BASIC but have no function until they are programmed; F2, F4, F6, and F8 are selected by pressing the respective keys *along with* the SHIFT key

CRSR keys move cursor vertically and horizontally on the screen (see SHIFT key); can be used in BASIC programs with PRINT statements to control screen layout

Courtesy of Commodore Business Machines, Inc.

type COMMODORE 64 on the screen, and then erase it by pressing the INST/DEL key. The INST/DEL key, like the CRSR keys, repeats itself as long as you hold the key down. Move the cursor back to the home position (line 1, column 1) by holding down the INST/DEL (or *delete*) key.

Starting from the home position, type the word PRINT. Now press SHIFT-←CRSR→ to move the cursor back to column 1 (over the letter P). To change PRINT to SPRINT, you must use the INST/DEL key *along with the SHIFT key*. Simply typing the letter S in column 1 won't work! Try it, and see what happens. The S replaces the P. The computer types in *overwrite* mode: Whatever you type at a given line and column position replaces the character (if any) already there. If you've replaced P with S, move back to column 1 again and type in P.



Changing PRINT to SPRINT requires you to *insert* a letter before the P. To insert a letter, press the SHIFT key and the INST/DEL key. This INSerTs a space at the cursor. Then you can overwrite the desired character in the space. In brief:

1. Move the cursor back to the P.
2. Press SHIFT-INST/DEL.
3. Type S.

Having an *insert* mode in BASIC means that, if you make an error in writing a BASIC program, you can fix the mistake without retyping the whole line. If you had written RINT 1 + 1 in the earlier example, you could have inserted a "P" instead of retyping the command. That may seem like a negligible difference, but it will mean a lot when you have ten or more lines to correct.

(Consult the keyboard illustration (Figure 1–3) on pages 20–21 for a list of the other special-purpose keys and a brief description of what they do. The use of the special function keys varies with the software, so a complete description of what each key does is impossible.)

Getting Acquainted with the Quote Mode

Telling your computer to PRINT 1 + 1 shows you one use of BASIC's PRINT statement. An equally important way to use PRINT requires quotation marks after the word PRINT. Just as in English

grammar, direct quotes (" ") imply that everything between them should be taken literally. When you read in the newspaper about your city's leading politician — *At the luncheon, Mr. R. responded by saying, "I agree and disagree up to a point, generally speaking"* — you can conclude that these were literally his exact words.

The PRINT ". . . ." statement tells the computer to display on the screen (literally) what comes between the quotation marks. Quotes are made by pressing the SHIFT and the number 2 keys simultaneously. Try this out by typing:

PRINT "COMMODORE 64" <RETURN>

The next line should display exactly what you place between the quotes — including spaces made by the SPACE BAR.

In the quote mode, the computer also obeys many other special function keys. For some illustrations, follow the examples below:

You type: PRINT "COMMODORE
then: press the ↑CRSR↓ three times
then: 64"

The screen should look like this:

PRINT "COMMODORE QQQ 64"

In the quote mode, pressing one of the CRSR keys gives you a graphics character — in this case a Q in reversed video. When you press RETURN, the computer obeys the CRSR key literally: It moves the number 64 down three lines. The result should be:

```
COMMODORE


          64
```

This time:

You type: PRINT"
then: press the ←CRSR→ key ten times
then: COMMODORE 64"

The screen should look like this:

PRINT "■■■■■■■■■■ COMMODORE 64"

Can you predict where the words will appear? Press RETURN to confirm your prediction.

Experiment in the quote mode with the other CRSR movements, if you like, and with the graphics characters made by pressing the SHIFT and COMMODORE keys with the letter keys.

To conclude the experimenting, use the SHIFT-CLR key in the quote mode. That is, press SHIFT-CLR between the quotes, so that the line on your screen looks like this:

```
PRINT "■"
```

Then press RETURN. Now let's move on to a more colorful subject.

Flying the Commodore's Colors

The C-64's graphics capabilities are better than other computers in its price range, and even better than computers costing many times more. One reason for its superior performance is that a special microprocessor, the Video Interface Chip (or VIC), spends its time handling animated graphic images known as *sprites*. Few other computers have a single-purpose graphics chip. Sprites are discussed along with graphics and sound in Chapter 4, but to whet your appetite, you might try displaying the C-64's sixteen basic colors on the screen.

You can see the names of eight colors on the front face of the keys numbered 1 through 8. These colors are displayed when keys 1 through 8 are pressed along with the CTRL key. (The ConTRoL key is a special-purpose key that is used in various ways by the software to be discussed in later chapters. Like the SHIFT key, it is always used along with another key and does nothing when pressed by itself.) Eight more colors are obtained by pressing keys 1 through 8 along with the COMMODORE key.

Here's a method for putting a complete color chart on the screen.

- Press CTRL-9. This reverses the image so that the background of each character shows the color. This step needs to be done *only once* at the start of the exercise.

- Press CTRL-1. This turns the cursor black.
- Press the SPACE BAR and hold it down until the cursor reaches column 1 of the next line. Then,
- Press CTRL-2. This turns the cursor white.
- Press the SPACE BAR and hold it down until the cursor reaches column 1 of the next line.
- Continue the procedure for keys 3 through 8. Then repeat the cycle using the COMMODORE key instead of the CTRL key. When the exercise is over, turn off the "reversed" image by pressing CTRL-0.

You should see sixteen bars of color on your screen in the order shown below in Figure 1–4:

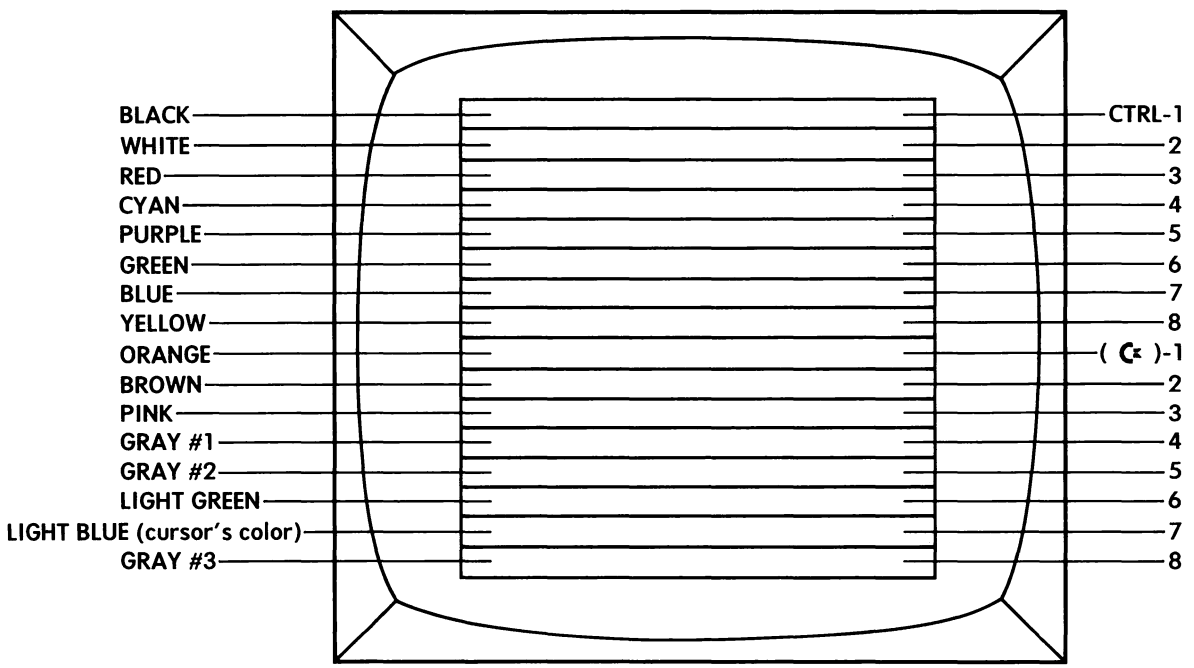
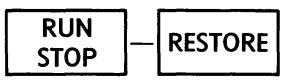


Figure 1–4: Commodore Colors.

Now that you know what the colors should be, you can fine-tune the color on your TV or monitor.



Resetting Your Colors

When changing colors on the screen, the cursor and any characters you type take on the current screen color. This can make the READY prompt (and any other characters) very hard to read. To get the cursor back to its normal light blue, press COMMODORE KEY-7, which is the code for light blue.

If you get confused during this color-changing test, or decide to start over again, you can return the screen to its normal color combination by pressing STOP and RESTORE. STOP-RESTORE has other uses, too. (See Figure 1–3 on pages 20–21.) On some C-64's, you have to hit the RESTORE key a couple of times — and forcefully — to make it respond.

A Taste of Programming

(Before beginning this section, press STOP and RESTORE together. You may have done so already after reading the box called "Resetting Your Colors." These keys return the screen to its normal condition.)

The first practice example in this book used the computer as a calculator. You typed PRINT 1 + 1 <RETURN>, and the computer displayed the number 2 on the screen. In this case you used the computer in the *immediate* mode — it executed your command immediately. Let's turn that immediate command into a *program* — an instruction (usually, a series of instructions) that is RUN at your request.

The program uses precisely the same PRINT statement, except that it has a line number in front of it. If there are several numbered steps in the program, the computer executes them in order when you give the command by typing RUN <RETURN>. A line number in front of a BASIC statement tells the C-64 that you're in *program* mode. Statements in BASIC without line numbers are in the *immediate* mode. Use only one mode at a time; otherwise, the computer won't know what you're trying to do and it will give you an *error message* by displaying ?SYNTAX ERROR on the screen.

To write the program, type line number 10 as shown below:

10 PRINT 1 + 1 <RETURN>

That's all there is to it. Notice that nothing *visible* happened when you pressed the RETURN key. Nevertheless, something important occurred behind the scenes: The computer stored your program in RAM memory. To demonstrate this, press STOP-RESTORE again, which clears the screen. Now type:

LIST <RETURN>

Your program should reappear. LIST tells the computer to display whatever program is stored in RAM. The command to execute (or run) a program in the program mode is:

RUN <RETURN>

When you run your program, the number 2 (the answer to 1 + 1) should appear on the next line. You can also run the program without listing it. Press STOP-RESTORE to clear the screen. Then type RUN <RETURN>. Did you get the number 2 on the screen again? If not, read the next section.

Every Computer's Fetish: Spelling, Syntax, and Accuracy

Computers in general are intolerant of errors. If you type in incorrect commands or data, they won't break or hold a grudge; but they won't do what you request either. The computer's fetish for accuracy is not a sign of its superior intelligence. On the contrary, it's a sign of its *lack* of intelligence.

People are smart enough to figure out what a sentence means, even if it's not punctuated correctly or if a word is misspelled. Computers aren't that smart — not yet. Since BASIC is a programming *language*, it has its own special words and grammar (or syntax). If you don't speak BASIC correctly, the computer won't be able to obey you, although it does have the decency to tell you why. In the following example, the word PRINT is deliberately misspelled. Try it yourself using the immediate mode, so you won't disturb the program in memory.

You type: PRIMT 1 + 1 <RETURN>
Computer: ?SYNTAX ERROR

The computer didn't know what you meant. Correct the spelling mistake, move the cursor to the end of the line, and then press

RETURN. You'll see how readily the computer forgives your mistakes.

If you're in program mode and using line numbers, the computer also tells you in which line the error occurred.

Clearing Memory

You've already used two methods for clearing the screen: pressing SHIFT-CLR and pressing STOP-RESTORE. Here's a third means of clearing the screen: type SYS64759 <RETURN>. This is a "SYStem command" that goes to memory location 64759 and does the same thing that STOP-RESTORE does. The *User's Guide* (first edition, 1982, p. 18) tells you that SYS64759 also erases everything in the computer's memory. It doesn't! The *User's Guide* is generally accurate, but not in this case.

The purpose of this section is to show you how you can erase everything in memory, which is necessary before writing a new program. If you fail to erase the old program, the new and old programs are combined and neither can run properly.

If you want to write a new program, clear memory first by typing NEW <RETURN>. Try it. Then type LIST <RETURN>. The computer responds with the READY prompt. It can't list your 10 PRINT 1 + 1 program because RAM memory has been erased.

Another way to clear the computer's memory is by typing SYS64738 <RETURN>. This is comparable to turning off the computer and turning it on again. On some computers this is called *resetting*. It's also known as a *warm boot*, because you're booting up the computer while it's running. Of course, you can also clear memory by physically turning off the computer and turning it on again—a *cold start*. There's nothing wrong with this in most cases, although it's inefficient and considered *gauche*. There are also times when a cold start affects the operation of peripherals, requiring you to go through other start-up procedures.

Now that you've cleared memory, let's type a more complicated program into the computer, one that instructs the computer to count from 1 to 1,000. You won't understand all the lines in the program, but you will get some action out of the computer.

After reading Chapter 3, you'll understand the program below and find that you're capable of writing more interesting programs than this one.

```
10 REM *** COUNTING PROGRAM
20 FOR N=1 TO 1000
30 PRINT N
40 NEXT N
```

After typing the program, type RUN <RETURN>. See if you can count as fast as your C-64. To slow down the count, press the CTRL key while the program is running. If you want to stop the program before it reaches 1,000, press STOP. To continue from where the program stopped, type CONT (for CONTinue) <RETURN>.

Typing Problems?

If you find yourself making frequent typing errors, start typing the program over again by pressing SHIFT-CLR, which clears the screen and returns the cursor to the home position. Then type NEW <RETURN> and start again.

If you've already typed some lines correctly, then don't start over with the NEW command. Instead, press SHIFT-CLR and then type LIST <RETURN>. The typed lines appear in consecutive order. Type over those that have mistakes. When you're through making the corrections to a line, press RETURN.

You don't have to retype the *entire* line, nor must you move the cursor to the end of the line before pressing RETURN. Pressing RETURN stores the line in RAM just as it appears on the screen, so long as the cursor is somewhere on that line.

Here's a second sample program. It allows you to interact with your computer. (It's about time you two were formally introduced.) Before typing in this program, type NEW <RETURN> to clear the previous program from memory.

```
10 REM *** GETTING ACQUAINTED
20 PRINT "WHAT'S YOUR NAME";
30 INPUT N$
40 PRINT "NICE TO MEET YOU,";N$
50 PRINT "YOU CAN CALL ME 64 FOR SHORT"
```

Make sure your typing is accurate, down to the quotation marks and punctuation. The $ is made by pressing SHIFT-4, just as it is on a typewriter.

After typing the program, type RUN <RETURN>. When the program asks for your name, type your name and <RETURN>.

Beyond getting the computer to perform for you, there's another reason for typing the practice programs above. In the next chapter, you'll learn how to SAVE programs on cassette tape and on diskettes; and you'll learn how to manage the stored information. The next chapter uses the programs you just typed as samples, although you can use any other BASIC program just as easily.

Saying Good Night

Don't leave your computer on for long periods of time when not in use. The constant image on the TV screen may burn in and affect the performance of the picture tube.

Although the computer must be turned on last, the system can be turned off in any order. *Warning*: Never turn the disk drive on or off with a diskette inside.

If your Datassette recorder or VIC-1541 disk drive is already connected to the computer and powered up, you can read on without shutting off your computer.

> Do *not* plug in the recorder or turn on the disk drive *while the computer is running.* Certain software packages — for example, Easy CALC RESULT in Chapter 6 — allow you to turn on the drive *after* the computer, but these are exceptions.

Thanks for the Memory

Why Save and Load?

Many a science fiction thriller has been based on the idea that a computer can remember everything you tell it, and then use the information against you. The films *2001: A Space Odyssey* and *War Games* use the computer's allegedly unerasable memory as a major premise in the plot. But computers of that sort are still fantasies. If you have nightmares about your C-64 attacking you, walk over to the wall socket and pull its plug. Then you won't have to lose any more sleep over that fear.

The fundamental truth about computer memory is not that it's unerasable, but that it's very easily erased. RAM and ROM memory make computers uniquely capable and useful. In fact, you couldn't program computers at all if they couldn't "remember" the series of instructions you give them. But if RAM and ROM were the *only* types of memory available to your computer, it wouldn't be half the machine it is.

One of the reasons RAM and ROM aren't sufficient can be illustrated quite easily by typing in the one-line program used in the previous chapter: 10 PRINT 1+1 <RETURN>. After running the program to make sure it's typed correctly, turn off the computer, and then turn it on again. You'll find yourself back to the initial screen display. You can use the RUN and LIST commands time after time, but all you'll get is the READY prompt. Your program is lost.

When you turn off the computer, you can be sure of three things:

1. The computer has no memory and no operating ability; and
2. when the computer is powered up again, its built-in (ROM) memory returns; *but*
3. anything that you typed into RAM prior to turning off the computer has been erased.

RAM can't retain any data or commands if the power is turned off because the information is preserved by an electrical current sending a charge to thousands of microscopically small switches. When the electric current vanishes (because you've switched the computer off), so does the information. Thus, all your hard work — whether it's a program, a term paper, or an operating budget for your business — is lost when the power is shut off.

Another limitation of RAM memory is the number of characters (or bytes) it can retain. Although there are an impressive 65,536 bytes of memory altogether, not all of these are available to you because the software (such as built-in BASIC, your word processing program, and so on) uses part of the total. And so does the operation of the disk drive, the printer, the modem, and so on. This is often referred to as the system overhead — the price paid in RAM to keep the system functioning.

If, for example, you're writing a novel or searching through a file of 1,000 names and addresses, you won't be able to fit everything you're working on in RAM at one time. The EASY SCRIPT word processor, for example, won't let you type more than about thirty double-spaced pages. After that the computer's memory will be exceeded. If you should by chance try to enter more information than RAM can hold, the computer will "forget" some of it.

Because the computer's memory is limited in size and *volatile* (subject to disappearing when the power is turned off), it is essential that you're able to save your work, store it for as long as necessary, and load it back into the computer whenever you choose.

To save, store, and load your work, you need an *external memory device*. These devices are sometimes referred to as mass storage, because they can hold large amounts, and varied types, of information.

Saving and Loading Files

The C2N Datassette recorder and the VIC-1541 disk drive are two types of *external memory devices* that serve as the computer's auxiliary memory. They store the contents of RAM on tape or on diskettes for as long as necessary, and let you *load* the information back into the computer later on. Tapes and diskettes are external memory media — the media on which information is magnetically encoded.

Loading information into the computer is no less important than saving and storing your work. If you want to use the C-64 as a word processor, financial planner, telecommunications device, filing system, home educator, and so on, you must load the software (that is, the applications programs) into memory. Although some software comes in cartridges that plug into the back of the C-64 keyboard, other programs have to be loaded into the computer from a cassette tape or from a floppy diskette. Except for writing programs and creating sound and graphics, everything you want your C-64 to do requires you to load a program — that is, software — to give the computer its instructions.

Ultimately, you'll need to do more than save and load information with your external memory device. COPYing enables you to make backup copies of valuable information and programs — in case something happens to the original. SCRATCHing enables you to delete or erase stored information that you no longer need. FORMATting prepares new diskettes to receive information. And there are other useful ways of managing the information stored in external memory.

The computer saves information in *files*, which are distinct collections of information — similar to what you'd place in a manila file folder. You can think of a tape or diskette file just the way you think of a file in your filing cabinet or desk drawer.

The pages ahead give you practice in:

- loading and saving files,
- reading the disk's directory,
- making backup copies of files,
- "scratching" (or erasing) files,
- renaming files,
- formatting new diskettes to receive files,

- testing diskettes for defects, and
- generally managing stored files efficiently.

You'll need to use many of the techniques above to handle the files generated by commercial software.

Files versus Programs

The word *files* raises a semantic issue that needs to be clarified. Most people speak of any collection of stored information — regardless of its content — as a file. Some computer scientists, however, don't like that metaphor because it blurs the distinction between programs and *data* files. Technically speaking, a data *file* is an arrangement of raw information, comparable to a list of names and phone numbers or the ingredients listed on the label of a soup can. But a *program* is a set of instructions that the computer can execute — comparable to instructions on *how to dial* long distance or a recipe for *making* the soup. Sticklers for terminology may object to talking about storing everything (including programs) as files, but as long as you recognize the distinction between programs and raw data, there's no harm done.

Don't be concerned if you're still a bit confused by the distinction between programs and data files. The concept of *files* has become slippery from overuse. The distinction will be restated and reinforced as it becomes relevant in the pages ahead.

File Types

The type of file you save depends upon what you're doing with your computer. Usually, you don't need to know the file type because the software you're using saves and loads for you. The file *type* refers to the way the computer retrieves the information in the file. The list below tells you what kind of file is created *in most cases* by the corresponding software.

Activity	*File Type*	*Management System*
Programming	Program	BASIC commands
Word processing	Sequential	Word processing software
Financial Planning	Sequential	Spreadsheet software
Record-Keeping	Relative	Filing software
Telecomputing	(Varies)	BASIC commands or software

Your Electronic Filing Cabinet

Since information is stored in files, it seems natural to think of the Datassette recorder or the VIC-1541 disk drive as electronic, high-speed, and paperless *filing cabinets*. Unlike a normal filing cabinet, you don't have to open the drawer and look for an appropriate place to jam in a stack of papers. Instead, the computer sends the information electronically through a cable and stores it for you in seconds on a magnetized disk or on tape. (The tape recorder can sometimes take minutes to load or save a file, depending upon the type of file and its length.) When you're ready, you can load the information back into the computer in seconds.

The information is stored on the magnetized surface of the disk or tape. A single disk (5¼ inches in diameter) can hold about 174k bytes (characters), or roughly four times more information than the computer's RAM (with BASIC loaded). The drive can search through that information — even locating a specific word, phrase, or figure — in seconds. Although the Datassette recorder doesn't condense information to the same degree or search it as rapidly, tape is also far more efficient than paper as a mass storage medium. It's also less prone to damage.

Tape versus Disk Memory

Comparing the relative merits of disk and tape storage is useful for those not yet committed to a system. The criteria are clear: cost, speed, memory size, software availability, and ease of use. Cost alone may be the clincher. A disk drive costs about $300 and each diskette from $3 to $4, depending on the quantity you buy. The Datassette recorder usually sells for about $75 and tapes go for as little as $2 each.

The major operational difference between the recorder and the drive is the way each retrieves the stored information. The recorder has to wind through the tape sequentially, from beginning to end. This type of searching, called *serial access*, is rather slow compared to the *random access* method of the disk drive. The disk drive's read/write head can find information quickly from anywhere on the spinning diskette, just as you can place the arm

of your record player anywhere on the surface of a record with equal speed.

A cassette-based storage system, for example, can't run database software — the type that maintains lists of customers and clients or that catalogues your record collection or your friends' addresses and phone numbers. Databases require the computer to read information from different areas of external memory with equal speed. But the recorder must plod sequentially toward the end of the tape.

The list below may help you to contrast these alternatives.

Advantages of the Datassette Recorder

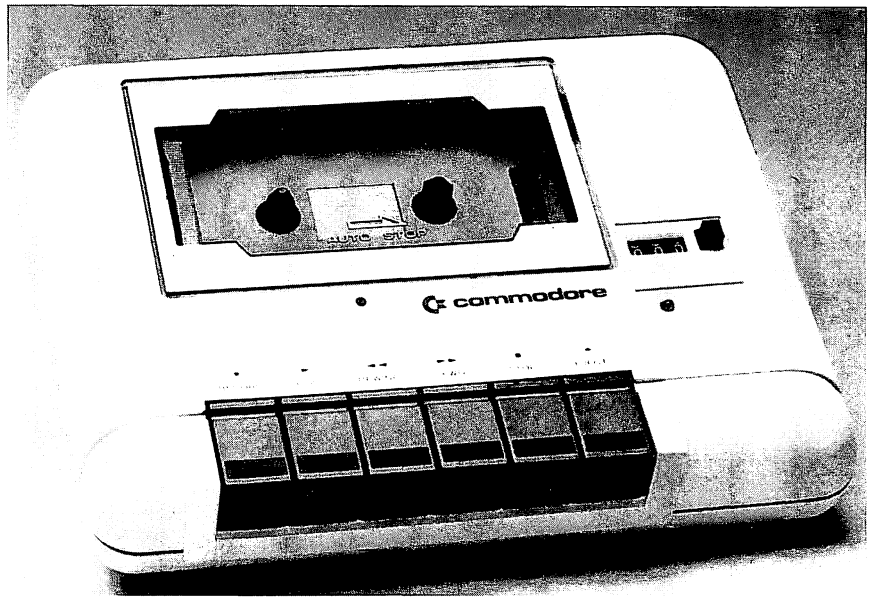
1. The recorder is less expensive than the disk drive.
2. Tapes are less prone to damage than diskettes.
3. The recorder is easy to use and comes with a clearly written, helpful manual. The drive requires more study and practice to use, and it comes with a manual that seems to make matters worse. (This chapter, however, clarifies and simplifies the use of the drive.)
4. Taped software is available commercially and through users' groups (see Appendix D).

Advantages of the VIC-1541 Disk Drive

1. The drive saves and loads files at high speed, many times faster than the recorder.
2. The memory capacity is many times larger than a tape cassette.
3. The most powerful software, especially word processing and business software, is distributed on disk and requires a disk drive for saving your work. (The tape recorder's limitations in speed, memory size, and mode of information retrieval restrict the software it can run.)

The Datassette Recorder and Some Tips on Tapes

The Datassette recorder works very much like an ordinary cassette tape recorder, except that it has no independent power supply. It's designed to plug directly into the C-64. It draws power from



Courtesy of Commodore Business Machines, Inc.

Photo 2–1: Datassette Recorder.

the computer, not from a wall socket. The cord connecting the recorder to the computer serves as both power supply and the data cord that carries the information.

In response to the right commands, information is shuttled back and forth between the Datassette recorder and the C-64 along the serial cable or data cord that connects them. More specifically, the binary patterns of 1's and 0's in the computer's memory are recorded as a series of beeps on tape. The recorder stores each file on its own section of tape. When you want to load a program back into the computer, the recorder searches the tape from beginning to end until it finds the information you've requested. It then sends the information back in the same order that it arrived.

Before attempting to use the Datassette recorder, read the *Instruction Manual*. It tells you how to connect the unit and how to care for the tapes on which your information is encoded. Make a list of the do's and don'ts of tape handling and set up a maintenance schedule for cleaning the tape head. If you want your recorder to have a long and accurate memory, you must observe the manual's cautions about exposure to dust and magnetic fields, and its maintenance procedures. You can use ordinary cassette

tapes, which are in fact preferable to the more expensive digital-quality or high-bias tapes preferred by audiophiles. The manual warns you against using tapes that are longer than thirty minutes, because they may strain the recorder's motor.

Are You Counting on Your Recorder?

Surprisingly, the *Instruction Manual* never mentions the three-digit counter located just above the recorder's SAVE light. The counter doesn't even show up on the diagram that labels the recorder's parts, but it's a very useful device.

Pressing the black button resets the counter to 000, which is what it should read when the tape is rewound. The counter enables you to find the approximate location of each file (or program) on the tape. Use the counter exactly as you would to locate a music track.

Note the number registered by the counter when you begin to save a file. Then note the number after the file's been saved. When the tapes are labeled, the position number should accompany the file's (or program's) name. For example:

TESTPROGRAM	000–005
COUNTINGPRO	005–009
CHAPTER1.HIST	009–053

Even with several files on one tape, accurate labels allow you to locate quickly the stretch of tape containing the file you want to save or load.

Another crucial reason to use the counter is to prevent accidental erasing of files. See Erase at Your Own Risk, page 40.

SAVE and VERIFY

To save a program, you first need to type one into memory. Then decide on a name for the program. The name can contain from one to sixteen characters. (See the box on pages 43–44 for a com-

plete discussion of naming files.) Lastly, you type in the SAVE command. SAVE is a BASIC command in the immediate mode — it's executed immediately after you press the RETURN key.

For a practice example, type in the sample program from Chapter 1 — the one instructing the computer to count from 1 to 1,000. Then follow these steps:

You type: SAVE "COUNTING.BAS" <RETURN>
Computer: PRESS RECORD & PLAY ON TAPE
(*Press the RECORD and PLAY buttons* simultaneously. *Computer screen goes blank momentarily.*)
Computer: OK

```
SAVING COUNTING.BAS
READY
```

(*Press the STOP button on the recorder. This releases the RECORD and PLAY buttons; it has nothing to do with saving the program.*)

Note: 1. The quotation marks around the name of the program are an essential part of the SAVE command. Do not omit them. 2. The ".BAS" in the program name is added as a reminder that the stored file is a BASIC program. 3. The screen goes blank to allow the computer to save, load, and verify faster and more efficiently. Maintaining the screen takes computer time away from these other operations.

Now, you must make sure that the right program was saved.

Whenever information is transmitted from the computer to external memory or back again, there's always the possibility of an error. Errors can be caused by a *glitch* (a sudden variation in electric current), dirt or dust on the tape, or occasionally (it seems), by some mischievous demon that inhabits your computer for a few seconds and then disappears. But BASIC provides a convenient VERIFY command to confirm that the SAVE was accurate.

The VERIFY command compares the program on tape to the program in the computer's memory. Start by rewinding your tape to the beginning of the stored program. If you used the

counter, the spot should be easy to find. Otherwise, you'll have to estimate how far back to rewind it.

If the program was saved without error (which is usually the case), the procedure goes as follows:

You type: VERIFY "COUNTING.BAS" <RETURN>
Computer: PRESS PLAY ON TAPE
(Press the PLAY button on the recorder. Computer screen goes blank momentarily.)
Computer: SEARCHING FOR COUNTING.BAS
FOUND COUNTING.BAS
(Computer screen momentarily goes blank again.)
Computer: VERIFYING
OK
(Press the STOP button on the recorder.)

If that little demon decided to pay you a visit, you'll see the message VERIFY ERROR. That means the program on tape doesn't match the one in the computer. Rewind the tape and SAVE the program again. If that doesn't work, try a different tape cassette. VERIFY the save each time until you get an OK. If nothing works, it's time to call your dealer or (if the cassette recorder has seen months of use) to clean the head.

Now save the second sample program from Chapter 1, but give this program a different name, for example: SAYHELLO.BAS. Rewind the tape and VERIFY that SAYHELLO.BAS was saved correctly.

Erase at Your Own Risk

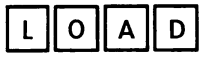
Erasing obsolete files is useful, because it makes room for new stored files on the same cassette. But there's only one way to proceed when erasing — cautiously! Once a file is erased, there's no way (as there is with the disk drive) to recover the lost program or information.

Erasing information on tape is accomplished by saving new information in the same physical location — just as you erase music on tape by recording over it. There's no special command that tells the Datassette to erase a file. Keep an up-to-date list of

your tape files and their locations. Otherwise, you might destroy one by accident. That's another good reason to use the tape counter.

It's also possible to erase information by simply pressing the PLAY and RECORD buttons simultaneously, or by running the tape through an ordinary cassette recorder and pressing PLAY and RECORD. "Bulk erasers," which demagnetize the tape's oxide coating, can clear the entire tape of stored files in seconds.

Getting LOADed



Loading a program sends the information you've stored from the tape through the data cord and into RAM. Once the program is back in the computer's memory, it can be listed by typing LIST <RETURN> or run by typing RUN <RETURN>.

Loading is *not* just for your own programs. Some of the software you use to turn your C-64 into a word processor, communications terminal, and so on, must be loaded into memory from tape. (Cartridge-based software is loaded automatically when the computer is turned on.) Here's how to LOAD a tape:

(Rewind the tape to the beginning.)
You type: LOAD "COUNTING.BAS" <RETURN>
Computer: PRESS PLAY ON TAPE
(Press PLAY on the recorder. The screen goes blank momentarily.)
Computer: SEARCHING FOR COUNTING.BAS
FOUND COUNTING.BAS
(The screen momentarily goes blank again.)
Computer: LOADING
READY
(Press the recorder's STOP button.)

There are two features of the LOAD command to keep in mind:

1. If you type LOAD <RETURN> without specifying a name for your program, the first program on the tape is loaded, assuming the tape is rewound completely. You can also load the first program by pressing SHIFT-RUN.

2. If you're planning to load a program farther on in the tape, the read/write head must read through every program that precedes it. The computer stops along the way to tell you whenever it finds a program. Then, the screen blanks out and the recorder continues. This can be time-consuming, not to mention boring. It's a good idea to check the tape label and fast-forward the recorder to the first counter position of the program you intend to load. The program then loads immediately.

Remember, as noted earlier, the LOAD command applies to programs only. Word processed (text) files, record-keeping files, and other data files can't be loaded. They can only be read from and written to. Your software takes care of these tasks for you. However, if you intend to work with data files in your programs, the Datassette recorder's *Instruction Manual* outlines the procedures for you on pages 13–15.

Free Insurance with Backup Copies

Making a backup copy of an important program is like taking out a free insurance policy. What have you got to lose? The only cost is the extra tape you've used and a few minutes of your time.

The reason for backups is that, as the manual notes, tape can be damaged in numerous ways — from exposure to dust and magnetic fields to spilled coffee. Keeping in mind Murphy's Law, (if something can go wrong, it will), the conclusion is obvious. Making backup copies of valuable programs is imperative.

There's no separate COPY command for the recorder. Making a backup copy is achieved with the SAVE command and two cassette tapes. SAVE your program on one tape, VERIFY the saved program, and then SAVE it on the second tape and VERIFY it. If the program to be backed up isn't in memory, then LOAD it into the computer, VERIFY the load, and SAVE the program to the second tape and VERIFY it.

Keep your backup tape in a safe place and use it only if necessary. If you decide to alter the original program later on, don't forget to update the backup copy.

Naming Files

If the computer's external memory is like a filing cabinet, it's natural to organize your work to be stored in files. The computer saves every product of your work in files — whether it's a BASIC program, a short story you've written with EASY SCRIPT, or a mortgage payment projection you've calculated with Easy CALC RESULT.

The computer treats a file the way you'd use a manila file folder. Each file folder is devoted to one subject and labeled with a meaningful name. (If you have a file folder of research notes for a report, you wouldn't throw in your MasterCard or Visa bills, nor would you label the folder "Tax Forms.") For example, we've named the two sample programs from Chapter 1 COUNTING.BAS and SAYHELLO.BAS

Here are three fundamental rules to follow when naming files.

1. Use from one to sixteen characters.
2. Do *not* use quotation marks (") as part of the file name.
3. Give each file on a particular diskette or tape a unique name.

Rule #1 specifies the minimum and maximum length of a file name. As a general rule, you should use at least enough characters to enable you to remember what the file contains. The letter L as a file name wouldn't tell you much; LETTER.BOB is much more informative.

Rule #2 states that you should not use quotation marks with file names because the LOAD and SAVE commands themselves use quotation marks as part of their syntax (punctuation). The computer interprets the quotation mark as part of the command, so it would confuse matters to make it part of the file name as well. The practice examples ahead will further clarify this point.

Rule #3 prevents ambiguity. Imagine that you're paging a friend at a crowded restaurant, and that you've identified him only as Bob. If there were more than one Bob in the room, there's no telling who would come to the phone. Obviously, you'd be wise to use Bob's last name, too, so he'd be identified *uniquely*. The disk drive and cassette recorder search for your files by name, so give each file its own name. Otherwise, the computer won't know which file you mean when you're saving, loading, copying, erasing and so on.

In fact, the disk drive won't allow this situation to occur. If you try to save a file with a name that's not unique, the computer responds with an error message: FILE ALREADY EXISTS. You must then choose a name that is unique.

File Types or Extenders

It's a good idea to adopt a system for annotating your files with *extenders*, which remind you of the type of file that has been stored. The extender is an extension of the file name. One convenient method is to type a period at the end of the file name and adopt a three-character code for designating the file type. .BAS would be a BASIC program file. .LTR would be a file containing

a (word processed) letter. .MOD might be information you've stored from a database with your modem. The full file names might be PROGRAM.BAS UNCLE ED.LTR, and DOWJONES.DATA. Extenders must remain within the sixteen-character limit.

Wild Cards [*] [?]

Files can also be referred to in groups by using *wild cards*, which are special symbols that get their name from the game of poker. In poker, when a card is designated as wild, it can stand for any other card you choose. The computer recognizes two symbols as wild cards: * and ?: * stands for any string of characters; ? stands for any single character.

Suppose you had a diskette that contained the following program files:

a. PROGRAM1
b. PROGRAM2
c. PROGRAM10
d. PROGRAM11
e. PROGRAM100
f. PROGRAM101

Using wild cards, you can refer to all or some of these files with only one name by a process called pattern matching. The computer matches the pattern of the full file name to the pattern specified by the wild card. Using * alone loads the first program in the disk directory.

P* refers to all the files, a through f.
PROGRAM* also refers to all the files.
PROGRAM? refers only to files a and b.
PROGRAM?? refers only to files c and d.
PROGRAM?* refers to files c through f.
PROGRAM1?? refers only to files e and f.

Wild cards can be used with some of the commands of the Disk Operating System, but you should first become skilled in managing files without them.

The VIC-1541 Disk Drive

Disk drives are high-speed memory devices that write information residing in the computer's memory onto an oxide-coated flexible (or "floppy") diskette. At your command, the drive also sends information back from the diskette into the computer. These are the drive's essential functions.

The 1541 is a "smart" peripheral. While most drives require you to load a diskette containing a Disk Operating System (or

Courtesy of Commodore Business Machines, Inc.

Photo 2–2: VIC-1541 Disk Drive.

DOS, pronounced *doss*) program, the 1541's DOS is contained in a ROM microprocessor chip *built into the drive*. This means that you don't have to boot up a separate diskette in order to save, load, copy, erase, and otherwise manage your information. The drive is ready to work with your diskettes as soon as it's powered up correctly. It also means that DOS doesn't take up space in the computer's memory, as the DOS programs in other systems do.

Commodore's DOS is *command-driven* rather than *menu-driven*. Most software for the C-64 presents you with a "menu" of choices. You simply select what you want the computer to do by pressing the key for the desired function — something like answering a multiple-choice question. Not so with Commodore's DOS.

To use Commodore's DOS, you have to remember the commands that instruct the drive. To complicate things, each command can be typed in at least three different forms. You must

also read the Error Channel after each disk drive operation to find out if anything went wrong. (Other Disk Operating Systems flash a message on the screen if there's been an error.) In short, Commodore's DOS, which handles the stored information on the 1541 drive, is not especially easy to use. Unfortunately, the VIC-1541 *User's Manual* is poorly organized, unclear, and conspicuously insensitive to the needs of beginners. The crucial pages of the manual are 1–8, which tell you how to connect the drive to the computer and how to insert the diskettes. Otherwise, you'd do well to read the rest of this chapter before attempting to decipher the VIC-1541 *User's Manual.*

The goal in this section of the book is to clarify what each DOS command does and to present in an orderly way the simplest means of implementing the command. In the practice exercises that follow, you will

- boot up the drive and the TEST/DEMO diskette that comes with it;
- read the diskette's directory;
- load and use the Wedge, a program that simplifies issuing commands to the drive;
- format a new diskette to receive information; and
- practice saving, loading, erasing, copying, and renaming files.

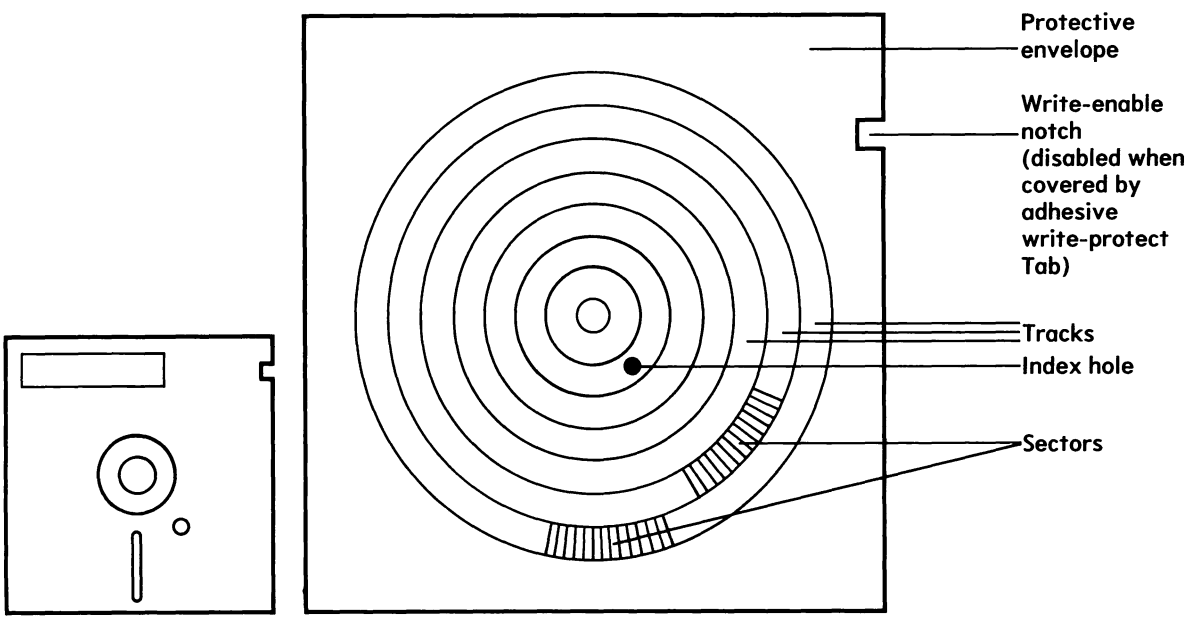
By the end of this chapter, you should be proficient enough to use the drive confidently in programming, graphics, and in conjunction with any of the software discussed in the rest of the book.

How It Works: A Brief Disk-ography

The drive writes and stores information on a circular, flexible (or floppy) disk that has a diameter of 5¼ inches. Disks are also called diskettes. The diskette spins within its black envelope inside the drive, much as a record spins on your turntable. One reason that disk memory is faster than tape is that the drive's read/write head can skip here and there around the disk at your command. This is called random access. As noted earlier in the chapter, you have the same type of instant access when you put a record on your

turntable — you can place the needle anywhere on the surface with equal speed.

Before a new diskette can be used, it must be divided into *tracks* and *sectors* so it can receive and store information in an organized way. At your command, the drive organizes the layout of the diskette in a process called *formatting*. Figure 2–1 shows you the layout of a formatted diskette. It has 35 concentric tracks, each of which is subdivided into sectors (also called *blocks*). There are 17 to 21 sectors per track. The diskette also maintains a *directory* and a *Block Availability Map* (or BAM), which organizes the different files and records where they're located on the disk.



When formatted by the VIC-1541 Disk Drive, a floppy disk is organized into:

35 tracks
683 sectors (or blocks)
256 bytes/sector

Track 18 is reserved for the directory and the Block Availability Map. The size of the files permitting, there is room for 144 distinct files in the disk directory. (To fit that many files on a disk, no file can occupy more than 4 sectors.)

There are 664 sectors free for storage, or about 170K–170,000 bytes. The exact number of bytes varies with the type of files being stored.

Figure 2–1: Diagram of a Floppy Disk (or Diskette).

For the practice exercises ahead, you should have on hand two blank 5¼-inch *single-sided, single-density* floppy diskettes. These terms refer to how information is stored on the diskette's surface. Some types of diskettes (double-sided) can record information on *both* sides, and others (double-density) can record up to twice the number of bytes per sector. The 1541 can format these other types, but it uses them as if they were single-sided, single-density.

Write-Protect Tabs

The TEST/DEMO diskette should come with an adhesive "write-protect tab" over the write-protect notch (see Figure 2–1). When the notch is covered, you can't write (save) or erase anything on that diskette. This tab prevents you from accidentally altering or destroying any of the files on the diskette. If there isn't an adhesive tab on the TEST/DEMO disk, use a tab from a box of new diskettes. The TEST/DEMO diskette is extremely valuable and should never be used for saving your own work.

Caring for Diskettes

Diskettes can be damaged by excessive heat, dirt, dust particles, and even by the magnetic fields created by such innocent devices as your telephone and television set. These are potential hazards that can alter or destroy your stored information.

That doesn't mean you should be afraid to handle them. They're designed to be used. Remember, paper is destructible, too, but you've already learned how to take care of information on paper, so it doesn't seem as intimidating. Although the manual that comes with the VIC-1541 disk drive doesn't even mention the subject of diskette care, it is vitally important. Here are some basic rules for the use and care of diskettes.

1. Don't subject diskettes to excessive heat or direct sunlight.
2. Don't use a wet or damp cloth to clean diskettes. If necessary, use a soft brush or compressed air to remove dust from the surface of the disk envelope.
3. Never touch the surface of the disk where it is exposed through the protective envelope.

4. When labeling diskettes, write on the label first and then stick the label on the envelope. If the label's already on the diskette, write with a soft felt-tip pen; do *not* use a ballpoint pen or pencil, because the hard point will damage the disk surface.
5. Don't erase a label while it's on the diskette. The eraser dust will damage the surface.
6. Don't attach anything to your diskettes with a paper clip or an elastic band.
7. Don't bend a diskette. Although they're flexible, don't try to find out *how* flexible.
8. Don't set the diskette down next to a telephone, television set, or anything that may emit a magnetic field. This includes the type of metal detectors that airline passengers walk through before boarding and that are sometimes installed in courthouses. (In most cases, authorities will hand-search computer equipment if you explain that it can be damaged by their equipment.)
9. Insert diskettes into the drive slowly and never force them if they're not seated correctly.
10. Store your diskettes in their heavy paper sleeves. A plastic case that holds a dozen diskettes costs about $4; the extra protection is worth the price.

Booting Up the VIC-1541 TEST/DEMO Disk

Turn on your system by following the instructions on pages 6–8 of the *User's Manual*. It might also help to consult pages 2–3 in the *Commodore 64 User's Guide* to clarify the locations and names of the ports at the back of the computer itself. Remember to turn on the disk drive *before* turning on the computer and *before* inserting a disk in the drive.

When you insert the TEST/DEMO diskette, make sure the drive's green light is on and that the red light is not flashing. If the red light is flashing, it means the drive is improperly connected. Don't attempt to use the drive until you have a green light. If the red light flashes despite a correct start-up procedure, the drive could be defective.

Here's the start-up procedure:

1. Turn on the TV.
2. Turn on the disk drive (before inserting the disk).
3. Turn on the computer.
4. Insert the TEST/DEMO disk in the drive and latch the drive door.

Examining the Disk Directory

L O A D

The *directory* contains a list of the files stored on the TEST/DEMO diskette. Think of the disk directory as a type of index or a table of contents. After a few months of working and playing with your C-64, you might have dozens of your own programs and files stored on a single disk. The directory reminds you of what's on each disk and the exact name of each file, so you can load it or work with it as needed. You'll consult your disk directory quite often. The directory is maintained by the computer and the drive — all you have to learn is how to display it on the screen.

To see the directory, you must first load it from the disk into the computer. Then, type LIST <RETURN> to display the directory on your TV screen.

Here's the procedure:

$ 8

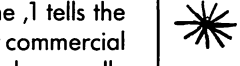
You type: LOAD "$",8 <RETURN>
Computer: SEARCHING FOR $
LOADING
READY
You type: LIST <RETURN>

As noted in Chapter 1, accuracy is essential when typing in BASIC commands. LOAD is a BASIC command in the immediate mode. The computer executes it immediately after you press RETURN. If the computer responds with ? SYNTAX ERROR instead of with the SEARCHING FOR $, check your typing for errors. If you find an error, simply retype the line and press RETURN.

The LOAD command requires you to place quotes around the information to be loaded. The $, when used with the LOAD command, stands for the disk directory. A comma separates the information to be loaded from the device code — the code specifying the device that contains the information. The device code for the disk drive is the number 8. To summarize:

LOAD	the command
" "	punctuation surrounding the information to be loaded
$	stands for the disk directory (when used with LOAD)
,	separates the information to be loaded from the device code
8	the device code for the disk drive

> If a program is written in *machine language* (see Chapter 4), more information is required in the loading command: LOAD "PROGRAM",8,1. The ,1 tells the computer that a machine language program is being loaded. Most commercial software is written in machine language, and the software manual generally provides very specific loading instructions.



The computer responds with the display below in Figure 2–2. (Your directory may vary slightly, depending upon when your TEST/DEMO disk was purchased.)

```
0     "1541TEST/DEMO         " 2X 2A
13    "HOW TO USE"              PRG
5     "HOW PART TWO"            PRG
4     "VIC-20 WEDGE"            PRG
1     "C-64 WEDGE"              PRG
4     "DOS 5.1"                 PRG
11    "COPY/ALL" TWO DRIVES     PRG
9     "PRINTER TEST"            PRG
4     "DISK ADDR CHANGE"        PRG
4     "DIR"                     PRG
6     "VIEW BAM"                PRG
4     "CHECK DISK"              PRG
14    "DISPLAY T&S"             PRG
9     "PERFORMANCE TEST"        PRG
5     "SEQUENTIAL FILE"         PRG
13    "RANDOM FILE"             PRG
558 BLOCKS FREE.
READY
```

Figure 2–2: TEST/DEMO Directory.

The first line of the display tells you the name of the diskette — 1541 TEST/DEMO — and its identification code — 2X. Every disk is given a name and ID code when it's formatted. When the drive saves (or writes) information to a disk, it uses the disk name and ID to determine that the correct disk is in the drive. (Of course, you won't ever write anything on the TEST/DEMO diskette.) The "2A" refers to the versions of the Disk Operating System that formatted the diskette. Every Commodore 64 diskette formatted by the C-64 and VIC-1541 drive has a 2A.

Look at the first two file names in the directory:

```
13   "HOW TO USE"        PRG
 5   "HOW PART TWO"      PRG
```

HOW TO USE, the first file on the diskette, occupies 13 blocks (or sectors) of the disk's total of 664 available blocks. The letters PRG mean that the stored information is a program. As you can see, the TEST/DEMO diskette contains only programs; there are no data files. Data files are signified by the letters SEQ (sequential), REL (relative), and USR (user). As noted earlier, these files differ in the way they are read by the drive, but the differences can be ignored for the present. In most cases, the software you're using handles the data files for you.

A Quick HOW-TO

The HOW TO USE file provides information about the purpose of each of the programs on the TEST/DEMO diskette. Let's load the HOW TO USE program and read these brief descriptions. They won't tell you very much, but you won't be using most of the programs anyway. The most useful programs are explained in this chapter.

To load the HOW TO USE program, move the cursor to the line below the READY prompt:

You type: LOAD "HOW TO USE",8 <RETURN>
Computer: SEARCHING FOR HOW TO USE
LOADING
READY
You type: RUN <RETURN>

(Notice again how the LOAD command is typed. The *name* of the file to be loaded is enclosed in quotes, and the comma comes after the quotes.)

The first screen of information appears after you type RUN <RETURN>. To read the next screen, press the SPACE BAR.

At the end of HOW TO USE, the computer prompts you to press RETURN to load HOW PART TWO, which continues the explanations of the TEST/DEMO program files. At the end of HOW PART TWO, the screen is blank, except for the READY prompt.

The disk directory must now be loaded again, because it was erased when the HOW TO USE file was loaded. Do you remember how to load the directory? If not, refer to the procedure on pages 50–51.

The Wondrous Wedge and DOS 5.1

C - 6 4 W E D G E

Now examine the TEST/DEMO file called C-64 WEDGE. (The file called VIC-20 WEDGE is to be used only with the Commodore VIC-20 computer.) The purpose of this short program (notice that it takes up only one block) is simply to load the program file that follows it: DOS 5.1. It "wedges" DOS 5.1 into the computer's memory, specifically into the C-64's BASIC interpreter.

The purpose of DOS 5.1 is to provide convenient, shorthand commands for loading and managing files. It's the most universally useful program on TEST/DEMO. Since C-64 WEDGE and DOS 5.1 work together, most C-64 users refer to both of them as simply the Wedge.

Why a Wedge?

Bob Fairbairn, the programmer who created the Wedge, deserves the gratitude of every VIC-1541 disk drive user. Without the Wedge, managing your stored files on the 1541 is a complicated, cum-

bersome task. For example, simply formatting a new diskette requires you to type:

```
OPEN 15,8,15
PRINT# "NEW0:DISKNAME,ID",8
CLOSE 15
```

The Wedge reduces that to one short line:

```
@N0:DISKNAME,ID
```

The Wedge reduces reading the disk drive's error status from a *three-line program* to two key presses: @ <RETURN>. The Wedge also simplifies loading programs, copying, backing up, scratching (erasing), and renaming files of any type. It lets you read a disk directory on the screen *without* erasing the program in the computer's memory. If you're planning to use the VIC-1541 disk drive, you must learn to use the Wedge. (Unfortunately, the *User's Manual* provides only a brief explanation of how to use the Wedge.)

Load C-64 WEDGE as you would any program. Your TEST/DEMO disk should be in the drive and the drive's green light should be on. Then:

You type: LOAD "C-64 WEDGE",8 <RETURN>

When you see the READY prompt, type RUN <RETURN>. The next screen looks like the screen below in Figure 2–3:

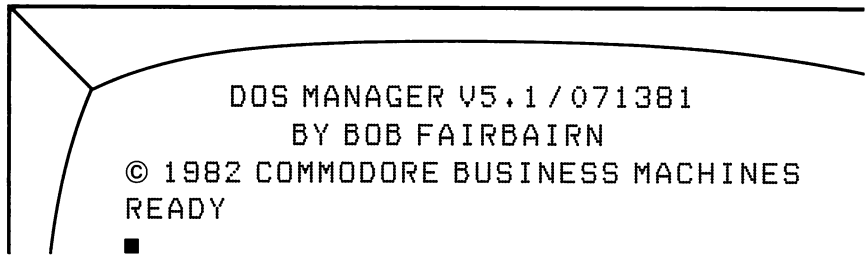


Figure 2–3: The Wedge Credit Screen.

When you ran C-64 WEDGE, it loaded the DOS 5.1 program for you. DOS 5.1 is a machine language program (see Chapter 4) which must be loaded at a predetermined memory location. That's why you should use C-64 WEDGE to load it for you. The

credit screen identifies the author and copyright holder of DOS 5.1. You do *not* need to type RUN <RETURN> to use the Wedge. It's ready to go.

Wedge Command Symbols

The Wedge simplifies managing your programs and files by providing convenient command symbols:

/	load a program or the disk directory (/$)
↑	load and RUN a program
@	prefaces any other command, such as NEW, COPY, RENAME, SCRATCH, and so on. @ also displays the directory and checks the Error Status of the drive.
>	same as @. This was the original wedge command symbol and it even resembles a wedge. The @ symbol is more convenient because it doesn't require using the SHIFT key.
←	save a program
%	load a machine language program

At the end of this chapter, you'll find a reference list of command formats that use these symbols. Don't try to use the commands, however, until you understand what each of them does.

Formatting Diskettes with the NEW Command

@ N

You cannot (and should not) use the TEST/DEMO disk for storing your own work. The purpose of the write-protect tab is to prevent you from doing so, even by accident. That means you must format a new diskette to save the files you create.

Formatting — as noted earlier — divides the diskette into tracks and sectors and creates a directory and Block Availability Map. Until it's formatted by the VIC-1541 disk drive, the diskette is useless. You can, however, format a diskette more than once, but it *erases everything on the diskette.* (The main reason for formatting an already formatted diskette is to bulk erase your files.)

The command to format a new disk is NEW, but it can be abbreviated by the letter N. The N is preceded by the Wedge command symbol @. The letter N is followed by a 0: (zero and a colon). Zero specifies a single drive. If you have two drives and want to format a disk on the second drive — device 9 — then you'd follow the N with 1: (one and a colon).

The NEW command includes giving the disk a name and a two-character identification code. This enables the drive to determine whether the correct diskette is in the drive. The name of the disk may have up to sixteen characters, including spaces. (The manual sometimes refers to the disk name as the Header.) Don't use quotation marks as part of the disk name. They may confuse the drive, because quotes are part of the NEW command when it's typed without the Wedge. As you begin to use your computer, you'll probably give your diskettes names related to the files they contain, such as CORRESPONDENCE, FINANCES, BASIC PROGRAMS, and so on. Let's call this one FIRST DISK.

Remove the TEST/DEMO disk from the drive and insert an unused single-sided, single density 5¼-inch diskette. Make sure the disk is seated properly before latching the drive door.

You type: @N0:FIRST DISK,AA <RETURN>

The drive now makes some whirring and grinding sounds as it divides your diskette into tracks and sectors. It also reserves space for a disk directory and a Block Availability Map. This takes a minute or two, so be patient. You can tell that the drive is working because the red light is on. When the formatting has been completed, the red light is turned off.

If the red light begins to flash, don't panic. Read on.

Reading the Error Status

@ RETURN

Using the Wedge, check the error status of the drive. Figure 2–4 on page 57 tells you what each part of the computer's response means. Generally, you need not be concerned with the track and sector location of the error. OK means you have nothing to worry about.

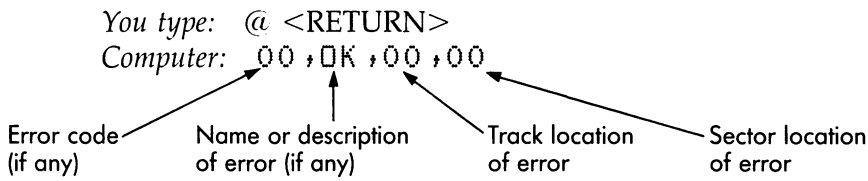


Figure 2–4: Error Code.

Checking the error status after each use of the drive is a good habit to develop. Normally, if there's been a problem, you'd see the red light flashing. But whether the red light flashes or not, you should make sure you're aware of any problems. The errors picked up by DOS involve either electronic or mechanical failures or incorrect procedures. Here are examples of each:

```
26,WRITE PROTECT ON,18,04
```

(You've tried to save a file onto a disk with a write-protect tab. *Remedy*: Remove tab or use a different disk.)

```
62,FILE NOT FOUND,00,00
```

(You've tried to load a program not stored on the disk currently in the drive. *Remedy*: Check to see that you typed the file name correctly. Check the disk directory. Your file may be on another diskette.)

The computer won't detect errors that are internal to your program or file. It detects operational errors. For a list of error messages and their remedies, see pages 43–46 of the 1541 *User's Manual*.

Displaying the Disk Directory

[@] [$]

As you've seen, the directory can be loaded like a program with the LOAD "$",8 command followed by LIST. This method has a major liability: It loads the directory into RAM, erasing the program you have in memory. The Wedge lets you abbreviate that command to /$ <RETURN>, but the same liability remains.

The real advantage to the Wedge is that it also enables you to *display* (not load) a directory on the TV screen without dis-

turbing the program in the computer's memory. Accomplish this by using the @ symbol.

You type: @$ <RETURN>

If you formatted the disk as suggested above, this display should appear on your screen:

```
0  "FIRST DISK     " AA  2A
664 BLOCKS FREE
```

The 0 is the drive unit. The 2A is the version of DOS in the drive's ROM. AA is the ID code.

You can easily prove to yourself that this display doesn't erase the computer's memory. Type NEW <RETURN> to clear memory. Then type a one-line program, for example: 10 PRINT "THIS IS A TEST" <RETURN>. Now display the directory with @ <RETURN>. Now RUN your test program. The program remains intact.

There are no files on your diskette as yet. You have an empty directory showing only the diskette's name, ID code, and the number of blocks (or sectors) available for storing your work.

SAVEing Your Work with the Wedge

The Wedge enables you to save programs quite simply, but you must also learn to save programs *without* the Wedge (see below), because you'll occasionally need to save-and-replace, which the Wedge doesn't allow. To save a program with the Wedge, though, is a simple matter. You'll use the ← key followed by the program's name. The name can be a *string* (see Glossary) of up to sixteen characters. You can use any characters except quotation marks. (For a complete discussion of naming programs and files, see the boxed discussion on pages 43–44.)

As an example, use the sample program from Chapter 1 — the one instructing the computer to count from 1 to 1,000. First, type NEW <RETURN> and then type the program into memory. Now, save the program on your formatted diskette:

You type: ←COUNTING.BAS <RETURN>
Computer: SAVING COUNTING.BAS
READY

Check the error status of the drive with @ <RETURN>.

After an error-free save, press @$ <RETURN> to see the new stored program in your directory. The directory should now include the line

```
1     "COUNTING.BAS"      PRG
```

VERIFY the SAVE

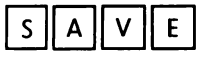
Checking the error status doesn't guarantee that the program saved on disk is the same as the program in memory. Leftover data in the drive's memory may get saved with your program and prevent it from running later on.

The purpose of the VERIFY command is to make sure you've saved what you intended to save. The VERIFY command compares the program on the diskette to the one in memory.

You type: VERIFY "COUNTING.BAS",8 <RETURN>
Computer: VERIFYING COUNTING.BAS
OK
READY

(That's assuming there's no discrepancy between the stored program and the one you typed into memory.) If some superfluous data somehow sneaked onto your diskette, the computer notifies you with the VERIFY ERROR message. If this occurs, reset the drive by pressing @I <RETURN>, and then try the save again. If that fails, type @u <RETURN>, which is equivalent to turning off the drive and turning it on again (a warm boot). If you still can't save accurately, print out your program (see Chapter 3, Listing and Printing) or copy it down on paper before shutting off the computer.

SAVEing Your Work *Without* the Wedge



Normally, the software you're using provides a command for storing your work onto disk. You'll simply type a key word, or press a single key, and the file is automatically saved. But when

writing your own programs, you can store them with BASIC's SAVE command. (Although you can accomplish the same thing with the Wedge command (←), using the BASIC SAVE command also allows you to save-and-replace.)

The SAVE command is simple. Give the program a name of up to sixteen characters, and then

You type: SAVE "PROGRAMNAME",8 <RETURN>
Computer: SAVING PROGRAMNAME
READY

You can try this with the counting program, if it's still in memory. The computer will save your program again, but this time under the name PROGRAMNAME instead of COUNTING.BAS. (You can save a single program under as many names as you like.)

When you see the READY prompt, check the error status of the disk by pressing @ <RETURN>. Then VERIFY the save. Displaying the directory should reveal a new file on the diskette:

```
1   "PROGRAMNAME"   PRG
```

SAVE-and-REPLACE with the @ Symbol

S A V E @

Writing a program, like writing a book, often feels like an endless process in which there's always room for improvement. Saving every version of a program in a separate file can take lots of disk space, so it's often preferable to save the improved version *over* the previous one. Saving-and-replacing requires a special command.

Suppose you make the COUNTING.BAS program run longer by asking the computer to count to 2,000. That means changing line 20 to:

20 FOR N=1 TO 2000 <RETURN>

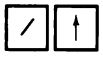
Type the line above and then use the LIST command to make sure the program is complete. Now replace the old program on disk with the new one, using the save-and-replace command.

SAVE "@0:COUNTING.BAS",8 <RETURN>

(The symbol @0: means "SAVE-AND-REPLACE IN DRIVE 0" the following program.) Remember to check the drive's error status and to VERIFY the save.

The diskette's directory reads the same as it did before, but the program it contains is the more recent one.

Getting LOADed



To run your own programs or commercial software, you have to load the desired program into the computer. You've already LOADed "C-64 WEDGE," but that was prior to using the Wedge. With the Wedge, there are two ways to load a program:

/ means LOAD

↑ means LOAD-and-RUN

For a practice example, remove COUNTING.BAS from the computer's memory by typing NEW <RETURN>. Prove to yourself that the program isn't in RAM by typing LIST <RETURN> or RUN <RETURN>, or both.

To load the program:

You type: /COUNTING.BAS <RETURN>
Computer: SEARCHING FOR COUNTING.BAS
LOADING
READY

(Check the error status with @ <RETURN>.) Using the LIST command, verify that the program has been loaded. Now, remove the program from memory by typing NEW <RETURN>.

To load and run the program

You type: ↑COUNTING.BAS <RETURN>
Computer: SEARCHING FOR COUNTING.BAS
LOADING
1
2
3
ETC.

Press CTRL to slow down the screen display. Press STOP when you've had enough.

COPYing Files to the Same Disk

@ C

This command lets you make a copy of any file on your disk and place it elsewhere (assuming there's room) on the same disk. Suppose you write a twenty-line program and then decide to create a slight variation of it. Rather than retype the whole program, you can COPY it, give it a new file name, and then revise the copy.

The COPY command requires you to state the *new* file name first. COPY can be abbreviated to C. The command takes the form: @C0:NEWNAME=0:OLDNAME <RETURN>. To paraphrase, write onto drive 0 with the name NEWNAME a copy of the file called OLDNAME.

Make a copy of the COUNTING.BAS program and call it COUNTING#2.BAS:

You type: @C0:COUNTING#2.BAS=
0:COUNTING.BAS <RETURN>
Computer: READY

Check the error status, and display the directory with the @$ command to make sure both programs are listed.

BACKING UP Files

COPYing a program isn't the same as *backing it up*. Backing up a program involves SAVEing the program onto *another* diskette for safekeeping. To back up a single program, simply LOAD it into the computer, insert a different formatted diskette, and then SAVE the program onto the other diskette.

Some people add the suffix .BAK to remind themselves that the copy is a backup. You might want to keep special diskettes reserved for backing up valuable files. Use those diskettes only when necessary — if your original is damaged, or if you revise your original and need to update the backup (with the save-and-replace command).

Procedure: If you want to back up one of your programs for practice, remove the current diskette from the drive, insert a new diskette and format it. Name this diskette BACKUP DISK,BB.

Then reinsert the FIRST DISK,AA and load the program to be backed up. Finally, insert your BACKUP DISK,BB and save the program on it. Check the error status, and VERIFY the save.

BACKING UP Diskettes with 1541 BACKUP

Since your TEST/DEMO disk is so valuable, it ought to be backed up as soon as possible. Unfortunately, neither the Wedge nor Commodore's DOS provides the means for duplicating the contents of one disk onto another (backup) disk using only one drive. (If you have two drives, you can use the COPY/ALL program on TEST/DEMO. The program is discussed briefly at the end of this chapter.) In some cases, you can LOAD a program, switch diskettes, and then SAVE the program to a backup diskette — using the method described above. But this only works for BASIC programs, and the most useful program on TEST/DEMO is DOS 5.1, a *machine language* (see Chapter 3) program. The LOAD-and-SAVE-to-another-disk method doesn't work with machine language programs.

The solution to this problem is a program called 1541 BACKUP. Reportedly, the program has been included on some TEST/DEMO diskettes. Check your TEST/DEMO directory. If the program name appears there, you're one of a fortunate minority. If it's not on TEST/DEMO, ask for it at your local Commodore users' group, or purchase a Commodore software package called **Disk Bonus Pack**.

The Disk Bonus Pack contains about forty programs, including games, simple educational programs, demonstrations, and some *utilities*. Utilities are programs that assist you in operating the computer. The Wedge itself is a utility program, and so is 1541 BACKUP. (The Disk Bonus Pack is discussed again in Chapter 9.)

The 1541 BACKUP program allows you to copy the contents of one disk to another disk using only one drive. It also duplicates machine language files, sequential and relative files (such as those created by a word processor and database management system). In short, it duplicates the contents of the original (or *source*) diskette onto the backup (or *destination*) diskette.

The program works by instructing you to swap diskettes, as described below:

- Insert the source diskette.
- The computer reads part of the source diskette.
- Insert the destination diskette.
- The computer writes what it just read onto the destination diskette.
- These steps are repeated until the backup is complete.

Two Warnings

The 1541 BACKUP program automatically *formats* the destination disk. Formatting a disk, as noted earlier, erases any information it contains. Thus, your destination disk should be a new, blank disk, or one containing obsolete files.

Some commercial software, such as the word processor EASY SCRIPT (Chapter 5), is *copy-protected.* Copy protection is intended to prevent users from duplicating a program disk and illegally reselling it (or giving it away). This would violate the copyright of those who created the software and would deprive them of potential sales. You can't duplicate a copy-protected disk with 1541 BACKUP, so don't bother trying.

Some manufacturers are adopting more aggressive measures to fight software piracy. There is an unconfirmed report that one copy-protected program *self-destructs,* if an attempt is made to copy it.

(1541 BACKUP is a BASIC program placed by Commodore in the public domain. It can be copied like any other BASIC program — and legally.)

For practice, let's make a backup copy of the TEST/DEMO disk. Start by loading 1541 BACKUP as you would load any BASIC program. Then type RUN <RETURN>. Figure 2–5 on page 65 shows a facsimile of the screen.

B
D

There are two types of backup procedures: **B** (for Block Availability Map) copies only those sectors containing information; **D** (for Duplication) makes an exact sector-by-sector copy of the source diskette, even copying blank sectors. Method B is faster, since it copies only the part of the diskette containing the files.

You type: B <RETURN>
Computer: DESTINATION
You type: BACKUP DISK <RETURN>
BB <RETURN>

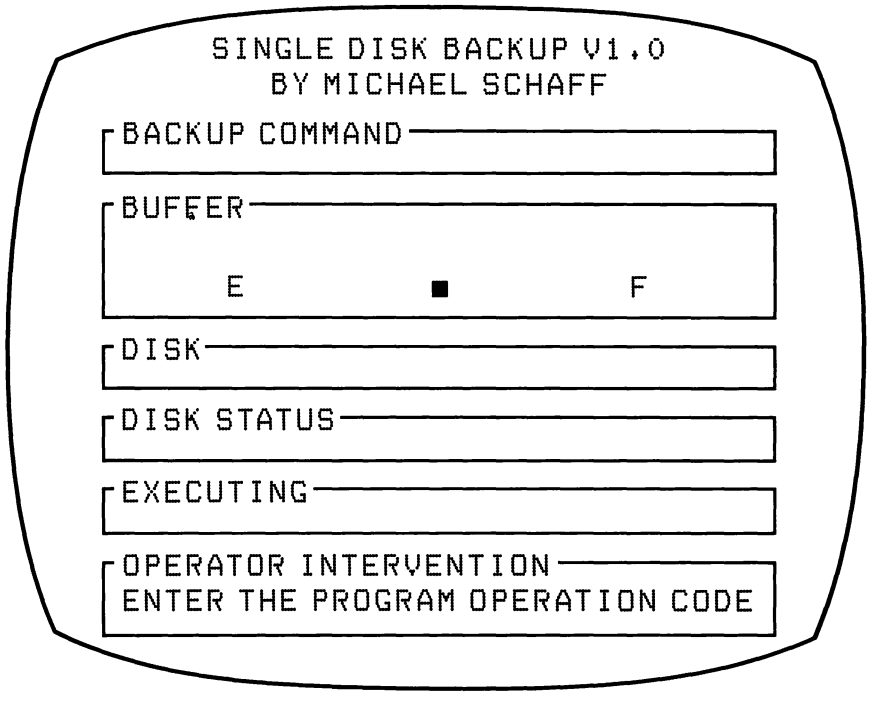


Figure 2–5: 1541 BACKUP Screen.

Notice that you must name the destination diskette and provide it with an ID code *in two separate entries.*

> *Computer:* INSERT DESTINATION DISK IN DRIVE
> Remove the disk with the 1541 BACKUP program, and insert a blank diskette. Then press RETURN.
> *Computer:* FORMATTING DESTINATION DISK

The formatting seems to make more noise than usual with this program. When completed,

> *Computer:* INSERT SOURCE DISK

It's a good idea at this point to place a write-protect tab on the source disk. (TEST/DEMO should already have one.) This is done to ensure that your source diskette won't be written on, if you mistakenly insert it instead of the destination diskette.

Insert the source disk and press RETURN. Then

```
Computer: VERIFY SOURCE DISK
     DISK
          1541 TEST/DEMO          2X
```

The computer asks you to confirm that the correct source disk has been placed in the drive. Press RETURN to continue. From now on, follow the prompts on the screen. Insert the source and destination diskettes as instructed by the prompts. Notice that the BUFFER (see Figure 2–5) acts as a gauge — from E(mpty) to F(ull) — measuring the contents of the buffer memory (a special area of memory reserved for the information on the diskette).

The disk swapping can take a long time, depending upon how full your diskette is. Backing up a full diskette takes about half an hour. Have something else to work on while this program does its job. TEST/DEMO, however, can be backed up in about five minutes. When the procedure is complete, the messages in Figure 2–6 appear in the bottom two boxes on the screen:

```
┌EXECUTING──────────────────────────────┐
│BACKUP FINISHED                        │
└───────────────────────────────────────┘

┌OPERATOR INTERVENTION──────────────────┐
│REMOVE DESTINATION DISK FROM DRIVE     │
└───────────────────────────────────────┘
```

Figure 2–6: Completed Backup.

To return to BASIC, press STOP-RESTORE. Before removing your backup copy from the drive, reLOAD C-64 WEDGE so you can continue with the practice examples in this chapter.

SCRATCHing Files

@ S

In Commodore jargon, SCRATCH means erase. Use this command with caution, because if you erase a program or file accidentally, it can be difficult, and sometimes impossible, to recover the lost information. You can only get it back by using the T&S (Track and Sector) program on the TEST/DEMO disk to alter cer-

tain bytes in the disk directory — and then only if you do so before writing anything else onto that disk. The procedure is difficult; it's a lot easier to work slowly and make sure that you don't erase the wrong file.

Using wild cards (see the box on page 44), you can erase more than one file at a time. But, for the reason just stated, this amounts to flirting with danger, and you should avoid it. Erase one file at a time.

Practice by erasing the second copy you made of the COUNTING.BAS program on FIRST DISK,AA. SCRATCH is abbreviated to S.

You type: @S0:COUNTING#2.BAS <RETURN>
Computer: READY

The results are reported when you check the error status:

You type: @ <RETURN>
Computer: 01,FILES SCRATCHED,01,00

Display the disk's directory to see that the file name has been removed.

As noted in Formatting Diskettes with the NEW Command on page 55, one quick way to erase *all* the information on your disk is to reformat it. Don't reformat a disk unless you're certain that you have no further use for its files.

RENAMEing Files

@ R

You might want to change the name of a file because you've changed its contents. Or, you may have a series of files that you want to reorder, for example, chapters in a thesis you're writing with a word processor. The RENAME command requires you to specify the *new* name first, just like the COPY command. RENAME can be abbreviated to R. To take a file called CHAPTER3 and relabel it CHAPTER7, type @R0:CHAPTER7=0:CHAPTER3 <RETURN>.

For practice, rename COUNTING.BAS to COUNT1-1000.

You type: @R0:COUNT1–1000=0:COUNTING.BAS <RETURN>

The computer doesn't display a message after the RENAME command, but you'll notice the red (busy) light go off when the drive stops working. Check the error status, and then display the directory to see the program listed with its new name.

Disk Drive ID Change

@ # 9

The device code for the disk drive is set at 8 in normal circumstances. A system with two drives, however, requires you to change one of the device codes to 9; otherwise, the computer won't know which of the drives you want to address.

To make the change, connect only one of the drives.

You type: @#9 <RETURN>

Then connect and turn on the other drive, which remains #8. The Wedge only works with drive 8. If you want to address drive 9 directly, you'll have to accomplish this with the BASIC commands. If you have two drives, experiment with the TEST/DEMO disk programs called COPY/ALL and DISK ADDR CHANGE.

LOADing Machine Language Programs

%

This command is for the technically sophisticated user. Most normal use of the C-64 involves BASIC programs or software that handles the machine language automatically. A machine language program must be loaded at a specific memory address. To load a machine language program in BASIC:

You type: LOAD "PROGRAM",8,1 <RETURN>

With the Wedge:

You type: %PROGRAM

Although the command is not difficult to implement, there are numerous situations that can prevent the program from loading correctly. Additionally, you need to specify a memory location after the program has been loaded. In any case, this isn't a command you're likely to need until you've had more experience. As

noted earlier, commercial software written in machine language usually provides an easy-to-follow loading procedure.

Booting OUT the Wedge

If for some reason you wish to remove the Wedge from memory, simply tell it to QUIT, abbreviated Q:

You type: @Q <RETURN>

Afterward, any attempt to use the Wedge results in the message ?SYNTAX ERROR.

Reference List of Wedge Commands

The following commands can be implemented after loading and running C-64 WEDGE on the TEST/DEMO disk. (Two useful commands for which there are *no* Wedge symbols are SAVE-and-REPLACE and VERIFY. However, these two BASIC commands can be used while the Wedge is in memory.)

The symbol definitions indicate whether the command applies to programs only or to both programs and (data) files. Although it's not indicated below, <RETURN> must follow each command.

←PROGRAM	SAVE a program or file
@SØ:PROGRAM	SCRATCH (erase) a program or file
/PROGRAM	LOAD a program or (/$) the disk directory
↑	LOAD and RUN a program
%MLPROGRAM	LOAD a machine language program
@NØ:NEWDISK,ID	format a new diskette — command includes disk name and ID code
@CØ:NEWCOPY=Ø:ORIGINALCOPY	make a new copy of your original program or file on the same diskette
@RØ:NEWNAME=Ø:OLDNAME	rename a program or file
@	read the error status of the disk drive
@$	display the disk directory without loading it into memory
@I	INITIALIZE the drive (reset the drive to its initial operating condition)
@#9	change the device code of the (second) disk drive to 9
@Q	ERASE the Wedge from memory

The TEST/DEMO Disk: A Quick Survey of the Remaining Programs

The rest of the programs on the TEST/DEMO disk are known collectively as disk *utilities*. They perform specific tasks that can be helpful when using the drive, but none of these programs is essential for saving and managing information. Most of the programs are *interactive*, meaning they tell you *via* messages on the screen what to do next. Some require a firm grasp of programming and are intended as aids to serious programmers.

The purpose of each program is stated below. If it seems relevant to your needs, try it out. Don't be discouraged if the results are not what you expect. Trial-and-error learning is part of using any computer. If you get stuck, go to a local users' group meeting (see Appendix B) and raise your question. (Or call a Commodore electronic bulletin board — see Chapter 8 and Appendix C — and post your question.) Someone who has already struggled down the same path will be glad to show you the way.

The COPY/ALL program allows you to copy the information from one diskette to another diskette *in another disk drive*. This program applies to you only if you have two disk drives. COPY/ALL can be used in conjunction with the Wedge command @#9, or with the DISK ADDR CHANGE program on the TEST/DEMO disk. Both of these programs temporarily change the device code of the second drive to 9.

PRINTER TEST does just what it says — tests the printer to make sure all characters (including graphics characters) are printing correctly.

DIR is an alternative means of checking the error status of the disk drive, but it's more difficult to use than @ <RETURN>. DIR can also display a directory and implement a segment of the Wedge.

VIEW BAM lets you look at the Block Availability Map. Viewing the BAM is useful for copy-protecting a diskette, a practice adapted by some software companies to prevent the illegal copying of their programs. The BAM also helps you in writing machine language programs, which require using specific memory locations.

CHECK DISK provides an easy-to-use procedure for testing the condition of diskettes. It makes sure that you can read data accurately from every sector of the disk. If there's a bad sector — one that won't store information accurately — it is flagged so that nothing is stored in it. You might want to use this program immediately after formatting a diskette with the NEW command. You can, however, use it any time without destroying your stored files because it only reads the sectors.

PERFORMANCE TEST is a more rigorous version of CHECK DISK. It both writes information in the sectors and reads it. PERFORMANCE TEST, however, destroys all the data on the disk by writing over it. *Important*: Use only with new diskettes or with those containing obsolete files.

SEQUENTIAL FILE and RANDOM FILE provide examples of both types of files for programmers who want to work with these data file types.

How to Talk to Your Computer: An Introduction to Programming

Why Bother?

In Silicon Valley, the home of the computer industry in California, there is a school administrator and computer expert who frequently speaks to users' groups and educators on the subject of programming. He begins by telling them *not* to learn to program. "Why spend the year or two it takes to become a competent programmer," he asks, "when you can simply buy inexpensive software for most applications?" Then, of course, he proceeds to tell them what programming is all about.

Why program? is a particularly relevant question for C-64 owners, because software is becoming more abundant and less expensive as the months pass. Yet programming warrants special treatment because it's not just another *applications* area. In the introduction to this book I recommend that you think of microcomputers as household appliances that process information. But programming doesn't use the computer as an appliance; instead, it tells the computer how to function as a particular kind of appliance. One popular metaphor compares computers to record players and the software to the records that are played on them. According to this metaphor, programming is *writing the music.*

There are several good answers to the question Why program? First of all, learning to program, at least in an elementary way, helps you to become computer literate. The unique thing

about computers is that they're programmable, so the best way to *understand* (and not simply to *use*) them is by writing some programs.

More specifically, programming enables you to speak directly to your computer, as you did in Chapter 1, using the commands LIST, RUN, and PRINT, and NEW, and in Chapter 2, using the commands SAVE, LOAD, and VERIFY. It also lets you write custom software to accomplish specific tasks. The sample programs in Chapter 1 demonstrate in a limited way how the computer responds to your instructions. Understanding programming often enables you to understand store-bought software better and to use it more effectively.

With experience and practice, you can create programs that do meaningful work for you. While commercial software has something for everybody, and usually includes features you don't need, your own software will be tailored to satisfy your private wish list.

Programming also requires you to think *logically* about the tasks you want your computer to perform. A programming language is really a tool for analyzing and solving the problems that computers can address.

Finally, programming is challenging, stimulating, and — when bugs aren't driving you crazy — truly enjoyable. (Bugs are errors that prevent the program from running the way it should; finding and correcting those errors is called debugging.) If you think you have computerphobia, learning to program conquers that fear. It makes you the master of your machine.

Chapter 1 (A Taste of Programming, page 26) explained the difference between the *immediate* and *program* modes. Chapters 1 and 2 illustrated the use of PRINT, RUN, LIST, NEW, SAVE, LOAD, and VERIFY. Review those chapters if you're uncertain about any of these BASIC commands. Chapter 1 also emphasizes the need for absolute accuracy in typing programs. The C-64 understands the BASIC programming language, but only if you, the programmer, use its vocabulary (which consists of sixty key words) and syntax (or grammatical rules) correctly. As noted in Chapter 1, computers aren't smart enough to know what you mean if you misspell words or make typing errors.

In this chapter, you'll learn how to speak BASIC to your computer *according to the rules of the language*. You won't be fluent

or completely versed in BASIC vocabulary — that would require at least a year of study and practice, but you will be on your way to writing some meaningful programs. As elsewhere in this book, you'll be using statements before they're explained systematically. While this departs from the style of presentation found in other books, it is consistent with the approach adopted here — that people learn by doing.

The Etymology of the Bug

Just like booting up, the terms *bug* and *debug* are pretty much inescapable elements of computer jargon. As noted in the text, a bug refers to something that prevents a program from working correctly, and debugging refers to the process of identifying and eliminating the problem.

The origin of these expressions is at least as interesting as the topic itself. As the story goes, the expression was first used shortly after World War II by computer scientists working on the Mark I, a computer with an extensive system of vacuum tubes and relays. After a program failed to run, the programmers eventually began inspecting the computer itself and found a dead moth interfering with one of the relays. When the moth was removed, the program ran.

In entering the incident in the operations log, someone wrote that the program had been debugged.

Why BASIC?

Although computers can potentially speak many languages, BASIC is the most widely used programming language for microcomputers. The word BASIC is an acronym for Beginners All-Purpose Symbolic Instruction Code, a language developed at Dartmouth College in 1964 expressly to make it easier for nonprofessionals to program computers. The language has been so successful that it's built into the ROM memory of most microcomputers, including the C-64.

BASIC makes it easier to program because it's a *high-level* language. High-level languages are those that use statements resembling everyday English. As you've seen, commands like SAVE, LOAD, and RUN do just what you'd expect. Other key words in the BASIC vocabulary — PRINT, GOTO, OPEN, CLOSE, NEXT, IF . . . THEN, and so on — are also easily grasped. Using these

words along with BASIC syntax, you can construct your programs somewhat intuitively to perform the task at hand.

Before BASIC, programming required the use of *machine languages*, which did not resemble English at all. Using machine language, which the computer understands directly, made it more difficult for the programmer to design the solution to a problem — and more difficult to fix a program that didn't run. Here are typical machine language statements written in hexadecimal — a numbering system based on 16. (The letters A–F represent numbers greater than nine.)

```
00020   027C   85   01
00021   027E   A9   80
00022   0280   85   02
00023   0282   A9   00
00024   0284   85   B1
```

Hexadecimal numbers are stored in the computer's memory as 1's and 0's, corresponding to the high and low voltages within the computer. Thus, the computer remembers statements in binary form — a number system based on 2. When the voltage is high, the value is 1; when it's low, the value is 0. (When the voltage is off, the computer loses its memory.)

What's built into the C-64 is actually a BASIC interpreter, which translates English-like BASIC statements directly into machine language.

Imagine how complicated it would be to tell the computer to PRINT 1+1 in machine language! Imagine writing a fifty-line machine language program that didn't work correctly, and then trying to locate the error! Now you know why BASIC is necessary.

There is also an intermediate language between high-level and machine languages — *assembly language*. Assembly language adopts an English-like symbol to represent a particular machine language instruction, but it's not nearly as easy to read or grasp as a high-level language. There's usually one assembly language statement for each machine language statement, which means that a given assembler program will be many times longer than the same program written in BASIC.

There are other high-level languages you can use to program the C-64, as long as you have an interpreter for those languages. The interpreter is loaded into the computer via disk, tape,

or cartridge. Some of these other languages are Pascal, FORTH, PILOT, and LOGO. PILOT and LOGO are surveyed briefly in Chapter 9. This book is not an appropriate place to discuss Pascal, which is often used for scientific applications, or FORTH, which is difficult to learn.

Your BASIC Calculator

In some programs, you'll want to tell the computer to perform a calculation for you. Using the computer as a high-powered calculator is easy, once you know the computational symbols. The C-64 performs the normal arithmetic operations: addition, subtraction, multiplication, division, and exponentiation. The symbols for these operations are:

+	addition
−	subtraction
*	multiplication
/	division
↑	exponentiation

Arithmetic computations can only be performed within legitimate BASIC statements. Typing in 2+2 = and pressing RETURN does not yield the answer. Rather, it produces a "?SYNTAX ERROR in 2" when you type RUN <RETURN>, because the computer interprets 2 as the line number of a program. Just as a quick experiment, try running 2+2 = and see what happens.

To use the C-64 as a calculator, type PRINT before the arithmetic expression you intend to solve. There are several practice examples in the *Commodore 64 User's Guide* (pages 23–26). Performing calculations is straightforward enough, so there is no need to provide another set of examples here. For practice, you may substitute different sets of numbers in the calculations, excluding division by zero (since division by zero yields an infinite number, the C-64 rejects the calculation with an error message).

Getting Scientific

Because of the way memory is structured, the computer must represent numbers greater than −999,999,999 or less than 999,999,999, and fractions smaller than .01, in *scientific notation* — by raising a decimal number to a power of 10. Watch what happens when the computer exceeds the limits of standard notation.

You type: PRINT 999999998 + 1 <RETURN>
Computer: 999999999

But now,

You type: PRINT 999999998 + 2 <RETURN>
Computer: 1E+09

The second sum requires the C-64 to translate the number into a *mantissa* ("1"), the letter E, and an exponent of 10 ("09"). In other words, the answer is, 1×10^9 or 1,000,000,000 (one billion).

To translate scientific notation into standard form (when the exponent is a positive number), move the decimal point to the right the exact number of places specified by the exponent. Fill in the number with zeros. In the example above, the decimal point follows the 1 (1.−−−−−), and it is moved to the right nine places (or nine zeros). Here's another example for comparison:

12.5×10^{12}

12.5E+12 = 12,500,000,000,000 (twelve trillion five hundred billion)
DECIMAL POINT MOVES 12 PLACES TO THE RIGHT

Now watch what happens when the number is too small to represent in standard notation.

You type: PRINT .02 − .01 <RETURN>
Computer: .01

But now

You type: PRINT .02 − .011 <RETURN>
Computer: 9E−03

With a negative exponent, the decimal point is moved to the *left* the number of places specified by the exponent. In other words,

9×10^{-3}

9.E−03 = .009 DECIMAL POINT (WHICH IS UNDERSTOOD) MOVES 3 PLACES TO THE LEFT

To check this translation, type PRINT .009 + .011, and the answer will be .02. You can also type .009 in scientific notation to perform the calculation.

When typing numbers on the computer, never use commas, as in 1,965,765. In BASIC, commas are used to space characters across the screen or to signal additional input. Using com-

mas in a number is bound to produce errors. The number above should be typed 1965765.

Parentheses are sometimes necessary to perform *mixed* calculations — those involving two or more arithmetic symbols. Normally, the expression 5+8/2 is ambiguous. It could mean "add five and eight and then divide the sum by two" — which yields 6.5. Or it could mean "add five to the quotient of eight divided by two" — which yields 9.

If no order of calculation is specified with pairs of parentheses, BASIC follows a predetermined hierarchy. Operations are performed in this order: exponentiation, multiplication and division, addition and subtraction. So typing 5+8/2 <RETURN> yields 9, because division is performed first. To indicate a different order, you must specify how you want the operations performed by using parentheses, just as you do in writing algebraic expressions.

You type: PRINT (5+8)/2 <RETURN>
Computer: 6.5

Operations on the expressions within parentheses are performed first. Thus, (5+4)*(9/3) = 27. If operators are equal in the hierarchy, and if there are no parentheses, the calculation runs from left to right.

For practice, see if you can solve the expressions below as BASIC would solve them. Try them out on the computer to check your answers.

3+5↑2
4*8−23
(16+6)*3−7
123↑((6−4)/2)
123↑(6−4)/2

BASIC Relations

To complete certain tasks, programs sometimes require a comparison of values. For example, a program that balances your checkbook ought to tell you if you're overdrawn. In other words,

the program should display a message like WARNING! MAKE A DEPOSIT SOON! if your balance falls below 0. Perhaps, in fact, you'd like the warning to come before it's too late — when the balance is less than $5. Your program must compare the balance (call it B) to the number 5. BASIC adopts several mathematical symbols that tell the computer to compare values:

<	less than
=	equal to
>	greater than
<=	less than or equal to
>=	greater than or equal to
<>	*not* equal to (greater or less than)

In our hypothetical example, we'd like to know, Is B < 5? The short program below illustrates in a limited way how the *relational operator* ("<") tells the computer what to do next.

```
10 PRINT "TYPE A NUMBER":
20 INPUT B
30 IF B<5 THEN PRINT "WARNING! MAKE A
DEPOSIT SOON!"
40 IF B>5 THEN PRINT "YOU'RE OK FOR NOW"
```

Although the INPUT and IF . . . THEN statements haven't been discussed yet, grasping them intuitively is sufficient to understand the program.

Now type RUN <RETURN>. (Reminder: Press RETURN after typing each statement in the program.) Below the word RUN, the question mark appears. The program is waiting for your input.

Computer: RUN
TYPE A NUMBER ?
You type: 4 <RETURN> (Any number less than 5 will do.)
Computer: WARNING! MAKE A DEPOSIT SOON!

For practice, you can substitute some of the other relational operators for "<" in line 10, and then run the program.

Self-Documentation with REM Statements

There's a saying that nothing in nature is wasted — everything has a purpose. This is also true of a good program. Every line of the program should bring you closer to solving the problem or accomplishing the task at hand, *except for REM statements*! REM (or REMark) statements don't cause the computer to *do* anything, although the computer remembers and stores them along with the program. Still, they have an important function: documenting the program and your own thinking.

REM statements can contain any characters at all — letters, numbers, and special symbols. Although the entire REM statement appears when you LIST the program, the computer ignores (does not attempt to execute) anything you type between REM and <RETURN>. The REM statement, however, cannot exceed a total of eighty characters (starting from the line number), because the C-64 doesn't recognize longer statements. If you need more than eighty characters to explain your thinking, use two consecutive REM statements.

In Chapter 1, REM statements gave titles to the short sample programs GETTING ACQUAINTED and COUNTING PROGRAM. More importantly, REM statements work as reminders of how the program works or what a segment of the program does. Below, line 15 has been added to the original program in Chapter 1.

```
10 REM *** COUNTING PROGRAM
15 REM * LISTS EACH NO. FROM 1-1000
20 FOR N=1 TO 1000
30 PRINT N
40 NEXT N
```

Since COUNTING PROGRAM is short and rather obvious, the REM statement may seem unnecessary. But in longer programs that may have been saved for months on disk or tape, strategically placed REM statements can save you hours of *re*thinking, trying to figure out what you did or why the program works (or doesn't yet work) in a particular way.

INPUT: Constant and Variable Data Types

Strings [I][N][P][U][T] [$]

In Chapter 1, the C-64 was described in general terms as an information processor. Although computers once functioned primarily as "number crunchers," they are no longer just high-powered adding machines. A large percentage of the information processed by computers involves *text*. Programs that handle airline reservations, word processing, record-keeping, searching databases, and so on, remember and display letters, words, and paragraphs *literally*. Literal text entered in the computer's memory is called a *string*.

Strings are enclosed in quotation marks following a PRINT statement. You can include any characters in the string, including spaces, CRSR keys, and color keys. The only prohibited character is the quotation mark, which is part of the statement syntax. You've already used a string in the sample program GETTING ACQUAINTED in Chapter 1:

```
20 PRINT "WHAT'S YOUR NAME";
```

In fact there are strings present in every line of the program, except for the opening REM statement and the END statement. WHAT'S YOUR NAME is a string *constant*: The text enclosed in quotes doesn't change when the program is rerun. Examples of string constants include:

```
ENTER ANY NUMBER:
FL#456-C**
BALANCE DUE = $34.98
```

Using PRINT statements (don't forget the quotation marks), display these strings on the screen for practice.

The string constant (including the quotation marks and the word PRINT) must not be longer than eighty characters (or two lines). If a longer string is necessary, use another PRINT statement.

Not all strings can be known prior to running a program, as the sample program in Chapter 1 illustrates. Line 30 requests the person running the program to type in his or her name:

```
30 INPUT N$
```

Since the programmer can't predict the operator's name in advance, the line requires a string *variable*. The dollar sign ($) identifies the variable as a string, and the letter N is the variable for the individual's name. *The variable names a memory location where — in this case — the person's name is stored.* Generally, the string is not a person's name. The string can be any combination of characters, including numbers and spaces.

String variable names may be up to two characters long. The first character must be a letter; the second may be a letter or a number from 0 to 9. The $ is always the last element of the variable name. Here are some typical string variables:

A$
N1$
N2$
RT$

There are two points to keep in mind regarding string (and the other types of) variables. First, as a mnemonic device, you can make string variables of more than two characters, although the computer reads only the first two characters (and the $) when it runs. Thus, NAME$ is a legitimate string variable, which has the virtue of reminding you that the variable stands for someone's name. The computer, however, reads only NA$. That means you can't use NAME1$ and NAME2$ as distinct variables.

Second, none of the letters in the variable name may form a BASIC key word. For example, DRUN$ is an illegal string variable because it contains the key word RUN. A list of BASIC key words can be found in Appendix C of the *User's Guide*.

Floating Point Numbers

In the business world, what computers do is sometimes called data processing. Data refers to the raw information that the program in some way manipulates. If a program computes a company's total revenues for the month of January, the amounts of

January's individual sales, interest income, royalty income, and so on, must be entered into the computer, so that the program can find their total. In this case, the income figures are the *data*, and the totaling of the figures and the printing of the result is the *processing*.

When specific numbers are entered as data, they are called *constants*, because their values don't change. Numbers with decimal points are called *floating point constants*, because a decimal point can appear after any digit in the number. Here are a few floating point constants:

```
-87.9768
5.
47953.87
-.9732
-1257.0756
```

BASIC also allows you to represent floating point numbers by variables. This is useful because the numbers, such as the sales figures for January, are not necessarily known in advance. The program must tell the computer to add the numbers *whatever they may be*. Floating point variable names follow the same rules as string variable names, except that the $ is dropped. Here are some typical floating point variables:

```
A
Al
FP
RA
RB
Y2
```

Integers 

BASIC offers the opportunity to work solely in integers (whole positive or negative numbers with no decimal points) if decimal numbers are unnecessary or undesirable. For example, a program that computes the average number of individuals in a club may be more pleasing to the membership if it doesn't result in a "fractional person," such as an average of 65.56 members. You can ensure that the data and the result are expressed in whole numbers by using integers.

Integer constants work just like floating point constants. The first BASIC statement you typed in Chapter 1 (PRINT 1+1) adds two integers. Integer variables follow the same rules as the other variable names, except that the percent sign (%) follows the variable, as in these examples:

A%
A3%
RF%

A program to compute the average number of members in a group for a three-year period might look something like this:

```
10 REM *** AVERAGING PROGRAM
20 PRINT "ENTER NO. OF MEMBERS IN FIRST
YEAR"
30 INPUT N1%
40 PRINT "SECOND YEAR"
50 INPUT N2%
60 PRINT "THIRD YEAR"
70 INPUT N3%
80 T%=(N1%+N2%+N3%)/3
90 PRINT "AVERAGE IS";T%
```

Assigns the value N1% + N2% + N3%/3 to the variable T% using BASIC's relational symbol "=".

T% assures that the average is expressed as a whole number.

When a solution in integers is requested, BASIC calculates the result in floating point numbers, and then drops any figures after the decimal point.

The INPUT Command

The INPUT command lets the person operating the computer supply the data necessary to execute the program. The concept of INPUT is no doubt clear, since it has overflowed the boundaries of computer jargon and has become part of everyday speech.

You've seen the INPUT command in action since Chapter 1, when the GETTING ACQUAINTED program asked you to type your name. The INPUT command always specifies an integer, floating point, or string variable. The command displays a ? on the screen (or printer) and stops the computer from executing any

more lines of the program until the required data are supplied via the keyboard.

INPUT allows you to include a prompt to the operator. Instead of

```
10 PRINT "WHAT'S YOUR NAME";
20 INPUT N$
```

you can type

```
10 INPUT "WHAT'S YOUR NAME";N$
```

More than one variable can be included in a single INPUT statement. Here's a sample program that multiplies any three numbers:

```
10 REM *** MULTIPLY ANY THREE NUMBERS
20 PRINT "ENTER ANY THREE NUMBERS"
30 INPUT "SEPARATE THEM BY COMMAS";
A,B,C
40 PRINT A;"TIMES";B;"TIMES";C; "=";
A*B*C
```

Since the final "C" in line 30 appears in column 40 (in other words, it's the fortieth character), the cursor automatically drops down to the next line. But don't forget to press RETURN anyway. This advances the cursor down another line, leaving a blank line on your screen between lines 30 and 40.

When an INPUT statement contains more than one variable, the items must be separated by commas. Remember, in BASIC numbers themselves must never contain commas. RETURN is pressed *after* the last item is entered. If RETURN is pressed after the first entry, BASIC displays (??) on the screen to let you know that more data is required by the program. If too many numbers are entered, BASIC tells you ?EXTRA IGNORED. (If alphabetic characters — string constants — are entered when the program expects a number, a ?REDO FROM START error message appears. In this case the operator has another chance to enter the correct data type.

Although INPUT statements can incorporate prompts and multiple variables on one line, it's often easier to read and interact with a program when each prompt and variable appears on its own line.

The program below illustrates the use of floating point variables and INPUT statements in a program that adds any three numbers. For practice, try experimenting with different ways of writing the program by using the options offered by the INPUT statement.

```
10 REM *** ADDING THREE NUMBERS
20 PRINT "■":REM*SHIFT-CLR PRODUCES THE
HEART
30 PRINT "TYPE A NUMBER AFTER EACH ?"
40 INPUT A
50 INPUT B
60 INPUT C
70 PRINT "YOUR NUMBERS ARE:"
80 PRINT A,B,C
90 REM COMMAS SPACE NUMBERS ACROSS SCREEN
100 PRINT
110 REM PRINT BY ITSELF LEAVES BLANK LINE
120 PRINT "THE SUM OF";A;"AND";B;
"AND";C;"=";
130 REM SEMI-COLON MEANS DISPLAY DATA
ON SAME LINE
140 PRINT A+B+C
```

Output: Formatting the Screen

Good screen design can make the information you display easier to read. A program that's hard to follow and to interact with is not a successful program, even if it accomplishes your initial objective.

BASIC offers several tools for formatting the screen; some of these are exemplified in the sample program above.

SHIFT-CLR [SHIFT]–[CLR]

In the immediate mode SHIFT-CLR erases everything on the screen. The CLR key is also programmable — within a program you can

instruct the computer to clear the screen. The technique is simple. Use a PRINT statement with SHIFT-CLR as the string between the quotes. Pressing SHIFT-CLR between the quotation marks of a PRINT statement produces a reversed video heart. There's an example in line 20 of the sample program on the opposite page.

Commas [,]

In a PRINT statement, separating the items by commas spaces the string across the screen. The comma works like the TAB key on a typewriter. The screen is divided into four 10-column widths. Each comma places the next string at the beginning of the next 10-column width. PRINT "1","2","3","4" <RETURN> yields the following:

```
1          2          3          4
```

Notice that the commas go *outside* of the quotes. There's an example in line 80 of the sample program above, although that PRINT statement involves string variables rather than constants. (*Note:* The comma has a different function when used with the INPUT statement.)

Semicolon [;]

The semicolon after a PRINT statement tells the computer *not* to skip down to the next line. Very often it makes good sense to display information on a single line. For example,

```
10 PRINT "WHAT'S YOUR NAME "
20 INPUT A$
```

is greatly improved by adding a semicolon at the end of line 10:

```
10 PRINT "WHAT'S YOUR NAME ";
20 INPUT A$
```

Run both programs to demonstrate the difference. There's a more complex example on line 120 of the sample program. Try running the program without the semicolon at the end of line 120; notice where the sum from line 140 appears.

Colon [:]

The colon allows you to add a second BASIC statement to a program line. You can string together as many statements as you wish, as long as you don't exceed the eighty character limit. Here's a short program that illustrates how statements can be combined in one line.

```
10 PRINT "TYPE A NUMBER": INPUT N:PRINT
"HALF THAT NUMBER IS";:PRINT N/2
```

When you run the program, notice also the effect of the semicolon before the last PRINT statement. It places the result of the calculation on the same line as the string.

In the sample program, the colon in line 20 appears as a REM statement explaining the reversed video heart. The colon doesn't affect the program output; it alters the way the program itself is displayed on the screen.

The SPC Command

[S][P][C][(]...[)]

Commas can space information in ten column widths across the screen, but some programs require more flexibility in screen formatting. The SPC command (the letters SPC followed by a number in parentheses) lets you control the number of spaces between characters displayed on the screen. The parentheses are filled in by a number from 0 to 255, depending upon how many spaces you want to leave between characters.

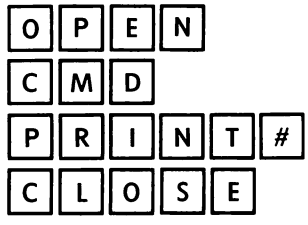
Suppose you're writing a program to compute the amount of money owed by three individuals: Smith, Jones, and Johnson. The SPaCe command can line up the names and amounts in an easily grasped layout.

```
10 PRINT "JONES" SPC(4) "SMITH" SPC(4)
"JOHNSON"
20 PRINT "$20" SPC(6) "$12.09" SPC(3)
"$37.87"
```

Run this short program and observe how the result is achieved. Notice that lining up SMITH with the amount he owes ($12.09)

requires *6* spaces to be inserted after the amount owed by Jones ($20). That's because the S in SMITH begins 4 columns after JONES, but *$20* is two characters shorter than *Jones*, so it takes 6 spaces to align the column.

LISTing and Printing



When booted up, the C-64 automatically displays information on the TV screen. But you'll want some programs to give you printed output; you'll also want to "LIST a program to the printer" — print the program itself — so you can send it to a friend or carry it around with you to debug.

LISTing programs to the printer and obtaining printed output from programs require you to open a channel of communication between the computer and the printer. Although the printer is hooked up, the computer normally sends information to the screen. Thus, BASIC commands must tell the computer to reroute information to the printer.

It's also necessary to set up a file for transmitting the information to be printed. This type of file is different from the files that are stored in external memory (discussed in Chapter 2). This type of file is a temporary grouping of data organized under a single *number* so the computer can keep track of it.

LISTing programs to the printer and obtaining printed output is a three-step process:

1. OPENing the channel and file,
2. CoMmanDing (CMD) the computer to send information to the printer instead of the screen, and
3. CLOSEing the opened file.

To use the OPEN command, you must specify the file number and device number (the device with which you wish to communicate). The file can have any number from 1 to 127; the device number of the printer is 4. The CMD and CLOSE commands refer to the same file number selected in the OPEN statement. The example below should clarify the format of the three commands.

First, practice using the printer in BASIC's immediate mode. To clear everything from memory, type

SYS64738 <RETURN>

As you may remember from Chapter 1, this is the command for a warm boot. It resets your computer, clearing RAM memory. Then, to open communications with the printer:

You type: OPEN 1,4
Computer: READY
You type: CMD1

(As usual <RETURN> must follow each command.) The printer should now print:

READY

Instead of appearing on the screen, this READY is sent to the printer. Now you know that the computer is sending output to the printer. Using the colon, the commands can also be typed on a single line:

OPEN 1,4:CMD1

However, do *not* attempt to type the command both ways, because you'll get a ?FILE OPEN error. Once the file is opened, the computer objects if you try to open it again. The computer is trying to protect you from the mistake of using the same file (file number 1) ambiguously.

For more practice in the immediate mode, type:

PRINT "PRINTER TEST"

The string should be printed instead of displayed on the screen.

Before closing the file, there is an additional step needed to "empty" the OPENed file of data that may remain. You can

do this by sending an empty data line to the file with a PRINT# statement. PRINT# is similar to PRINT, except that it's used for sending data to a device other than the display screen — such as the cassette recorder, disk drive, or printer. CLOSE the channel to the printer by typing:

```
PRINT#1
CLOSE 1
```

Now type PRINT "PRINTER TEST" again. The output returns to the screen.

LISTing Your Programs

The next step is to LIST a complete program to the printer. Assuming you've successfully CLOSEd the file used above, you can either OPEN file 1 again, or choose a different number. First type in one of the programs above as a sample, and then OPEN a printer file and use the LIST command. Your screen should look like this:

```
READY
10 PRINT "JONES" SPC(4) "SMITH" SPC(4)
"JOHNSON"
20 PRINT "$20" SPC(6) "$12.09" SPC(3) "$
37.87

READY
OPEN2,4:CMD2:LIST
```

When you press RETURN, the program listing will be printed instead of displayed on the screen. Make sure to close the file when you're through by typing PRINT#2:CLOSE2 <RETURN>.

Printed Output

The last step is to get your program to print its output. Since the example above is already typed, add these two lines and then RUN the program:

```
5 OPEN 1,4:CMD1: REM * PRINTED OUTPUT
30 PRINT#1:CLOSE1
```

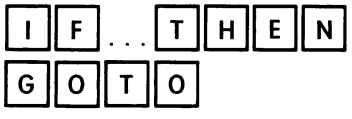
Notice that a file has to be OPENed and CLOSEd *within the program*. It's not enough that a file has been OPENed in the immediate mode. The C-64 allows you to keep ten files open simultaneously, and you should use a different file number for each purpose. Use a REM statement to indicate what each file contains.

The program's printed output looks like this:

```
JONES     SMITH      JOHNSON
$20       $12.09     $37.87
```

The printer file has been closed by line 30 so if you type LIST <RETURN>, the computer should display the program's lines on the screen.

Branching Out with IF . . . THEN and GOTO



One of the signs of human intelligence is the ability to react *conditionally* — to evaluate a situation and adjust our actions to the circumstances. IF a certain condition is met, THEN we do one thing. Otherwise, we take an alternate course of action. The computer imitates this type of human decision-making with the IF . . . THEN statement. The two words must appear together in a single BASIC line.

In the context of a program, the IF . . . THEN statement implements a technique called *branching*. Stated simply, branching allows the program to assume distinctly different directions (like the branches of a tree), depending upon whether certain conditions are met. If the condition in the IF part of the statement is met, then the computer executes the BASIC command following the THEN. If the condition isn't met, the computer moves on to the next line number.

In looking at BASIC relational symbols, we used the following example:

```
10 PRINT "TYPE A NUMBER";
20 INPUT B
```

```
30 IF B<5 THEN PRINT "WARNING! MAKE A
DEPOSIT SOON!"
40 IF B>5 THEN PRINT "YOU'RE OK FOR NOW"
```

Branching is often accomplished with the GOTO statement. GOTO enables you to tell the computer to execute a line out of sequence — to branch off to another segment of the program.

With IF . . . THEN, GOTO can cause the computer to jump to another section of the program when the IF condition is met. In effect, the GOTO statement takes the place of the THEN. When IF . . . THEN and GOTO are combined, either the word THEN or the word GOTO should be dropped from the line.

Incorrect	*Correct*
IF X=1 THEN GOTO 100	IF X=1 THEN 100
	IF X=1 GOTO 100

Here's a program that branches with the IF . . . THEN and GOTO statements:

```
5 REM *** ADDING MACHINE
10 PRINT "SELECT TWO NUMBERS"
20 PRINT
30 PRINT "SEPARATE THEM BY COMMAS"
40 PRINT
50 PRINT "WHEN YOU'RE DONE PRESS RETURN"
60 INPUT A,B
70 PRINT "NOW TELL ME THE SUM";
80 INPUT SUM
100 IF SUM=A+B THEN 150
110 PRINT "SORRY, YOU'RE WRONG"
120 PRINT
130 PRINT "TO TRY IT AGAIN TYPE RUN-RETU
RN"
140 PRINT "CONGRATULATIONS"
145 PRINT
150 PRINT "YOU DON'T NEED ANY MORE PRACT
ICE, BUT . . .":GOTO 130
```

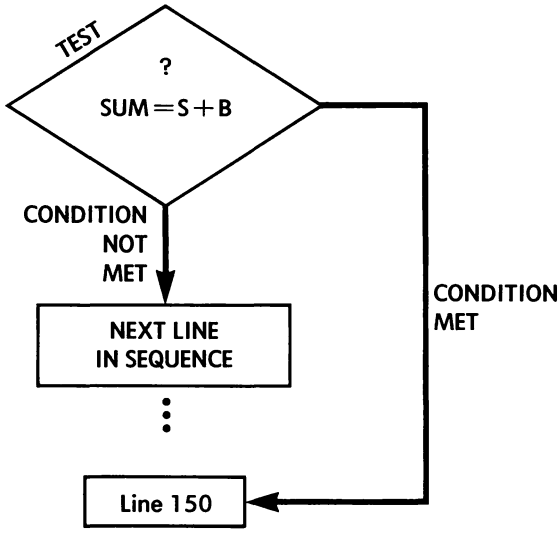


Figure 3–1: The Computer Makes a Decision.

Tips on Numbering Lines and Editing Programs

Since the computer executes instructions in numerical order, you can easily modify programs by adding statements between existing line numbers. That's why it's a good practice to increment line numbers by tens. If the lines are numbered by tens, you have room to add nine lines between any two lines of the program.

If more than nine lines are needed to alter a program, you'll have to renumber some of the existing lines. The easiest way to renumber a line is

1. LIST the program.
2. Move the cursor to the line you want to change, type the new number over the old, and press RETURN.
3. Move the cursor down below the READY prompt, type the *old* line number and press RETURN (typing a line number followed by RETURN cancels the line).

In general, when editing a program, you should LIST the program, move the cursor to the line in question, make the necessary changes, and type RETURN. You don't need to retype the entire line.

You can LIST part of a program by specifying the line numbers after the command. For example, LIST 30-200 <RETURN> LISTs all lines from line 30 through line 200. Similarly, you can RUN *parts* of a program. In long programs, these options make debugging and modifying the program much easier.

Learning to Love the Loop: Introducing FOR . . . NEXT

F O R . . . N E X T

While computers are no match for the subtlety of human intelligence, people are no match for computers in processing information by *iteration*. In plain language, iteration means repetition, as in the word "re*iterate*." Programmers refer to repeatedly executed lines as a *loop*. The diagram below makes it clear why. The simplest type of loop is created by a GOTO statement, which (as we saw in the preceding section) can make the computer jump to a line out of sequence. If GOTO specifies a line that's already been executed, it creates a loop.

The simplest example is a loop from which there's no exit:

```
10 PRINT "THIS IS AN INFINITE LOOP"
20 GOTO 10
```

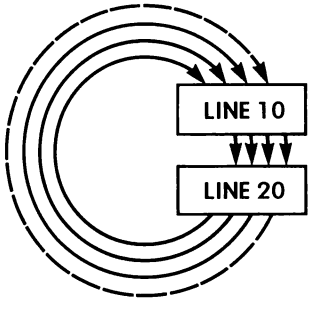


Figure 3–2: The Loop.

Figure 3–2 suggests what happens when you run this program. When you've had enough, press the STOP key. (To slow down the display, press the CTRL key.)

A more sensible kind of loop is one that repeats a procedure a finite number of times for a specific purpose. When the intended number of iterations has been reached, the program exits from the loop and the computer executes the next instruction in line.

The obvious question is, How does the computer know when the correct number of iterations has been reached. Two of the methods that BASIC uses to limit a loop are a combination of IF . . . THEN and GOTO statements, and a FOR . . . NEXT statement. Taking these methods in order, the IF . . . THEN method counts the number of iterations with a variable (say N), then it compares N to the number of desired iterations (say 50). When N>50, a GOTO statement takes the program out of its loop. The power of the loop is that it causes the computer to repeat designated steps any number of times, but you need to specify the steps only once.

As an example, here's a program that adds all the integers from 1 to 50:

```
5 REM A FIFTY-CYCLE LOOP
10 N=1:REM N=NUMBER - STARTS WITH 1
20 T=0:REM T=TOTAL - STARTS WITH ZERO
30 T=N+T
40 N=N+1
50 IF N>50 THEN 70
60 GOTO 30
70 PRINT "THE SUM OF 1-50 IS";T
```

Assigning Values to Variables

Up until now, you've given a value to a variable via the keyboard, using the INPUT statement. But the value of a variable can also be set by the program and changed during the course of the program. Suppose we have a variable N that counts the number of times the loop is executed. N is set for 1 at the beginning of the program. If the value is incremented by 1 each time a series of instructions is executed, then the programmer can construct a test to determine when the counter reaches the desired limit.

Remember, the variable names a memory location where the value is stored, so the value of a variable can change during execution.

In the FIFTY-CYCLE LOOP program, the counter begins with 1 (line 10) and the total (T) is set at zero (line 20). The value of both variables increases with each loop. When N exceeds 50, the loop stops and no more numbers are added to the total.

There are three steps to creating a loop: initializing the counter (line 10); incrementing the counter (line 40); and terminating the loop after the desired number of iterations (line 50). The FOR . . . NEXT statement condenses these steps into a two-part statement that optionally allows you to change the increment rate as well. This is a convenient way to program loops, and most programmers favor it over the IF . . . THEN/GOTO combination. With FOR . . . NEXT, you can also program up to nine loops within loops (or *nested* loops), but programs complex enough to require nested loops are beyond the scope of this book.

The FOR . . . NEXT loop requires two BASIC lines. If you use a FOR without a NEXT (or vice versa), you'll get an error message. The FOR part — FOR N=1 TO 1000 — initializes the variable at 1 and specifies one thousand iterations. The NEXT part

— NEXT N — increments the variable by 1, unless specified otherwise with a STEP command added to the FOR statement — FOR N=1 TO 1000 STEP 2. This line increments the variable N by 2 after each iteration. This would cause the program to count by two's, eventually executing 500 loops.

As you may remember, the COUNTING PROGRAM in Chapter 1 (see page 29) used a FOR . . . NEXT loop to instruct the computer to count from 1 to 1,000. For a more interesting example, let's have the program compute the squares and cubes of every whole number from 1 to 1,000.

```
5 REM *** 1000 SQUARES & CUBES
10 PRINT "NUMBER","SQUARE","CUBE"
20 FOR N=1 TO 1000
30 PRINT N,N*N,N*N*N
40 NEXT N
```

If you want to slow down the screen display, press the CTRL key. If you want to stop the program altogether, press STOP. If you want printed output from the program, add these two lines:

```
15 OPEN 1,4:CMD1
45 PRINT#1:CLOSE1
```

It's important to note that the file referred to in line 15 isn't OPENed *within* the loop. If you open a file within a loop, the computer attempts to open the file *again* on the second iteration of the loop. The result is a FILE OPEN error — you can't OPEN the same file twice.

Saving Space with Abbreviations

Although BASIC cannot handle lines of more than eighty characters, you can get more on a line by abbreviating commands and eliminating the spaces between commands and strings. The abbreviation for PRINT is ?. The line 10 PRINT "THIS IS A TEST" can be abbreviated to 10?"THIS IS A TEST". The computer ignores spaces between the line number and key word, and also between the key word and the *argument* (or value) that follows it.

Other key words can be abbreviated by pressing the first letter of the word plus a SHIFTed second letter. For example, instead of LOAD, type L SHIFT-O; instead of SAVE, type S SHIFT-A; instead of VERIFY, type V SHIFT-

E; and so on. (A list of abbreviations can be found in Appendix D of the C-64's *User's Guide*.) When two key words have the same first two letters (GOTO and GOSUB), type a SHIFTed third letter (GO SHIFT-T and GO SHIFT-S). When you LIST the program, the BASIC interpreter displays the key words in full, even though you typed them in abbreviated form.

Program Design

A series of specific, ordered steps leading to the solution of a problem is called an *algorithm*. Some programmers consider creating algorithms to be the heart of their work. Some beginners, however, hear the word algorithm and run the other way. Actually, algorithms amount to logical thinking and methodical planning. Whether or not you're attached to the word algorithm, it's helpful to make a list of the steps necessary for solving the problem your program is addressing.

You can also improve your program design by creating a visual representation of the flow of the program — a *flowchart*. For complex programs, the flowchart is an essential part of designing an efficient solution to the problem and identifying bugs. Programming hasn't made pencils and paper obsolete.

The shapes used by flowcharts have been standardized. Oval shapes refer to the start and end of the program; diamond shapes signify decisions or IF . . . THEN tests. Parallelograms are used for INPUT/OUTPUT statements (such as PRINT statements), and rectangles for most other statements. The advantage to standard symbols is that others can read your flowcharts with a minimum of explanation.

Let's conclude this introduction by designing and programming a guessing game (Figure 3–3) that uses each type of command discussed in this chapter. There are two unfamiliar BASIC statements that are explained in the box on pages 100–101. The program tells the computer to pick a random integer from 1 to 1,000 and gives the player fifteen tries to guess the number. The program uses the IF . . . THEN statement to compare the player's guess to the computer's number. It uses the FOR . . . NEXT loop to count the number of guesses.

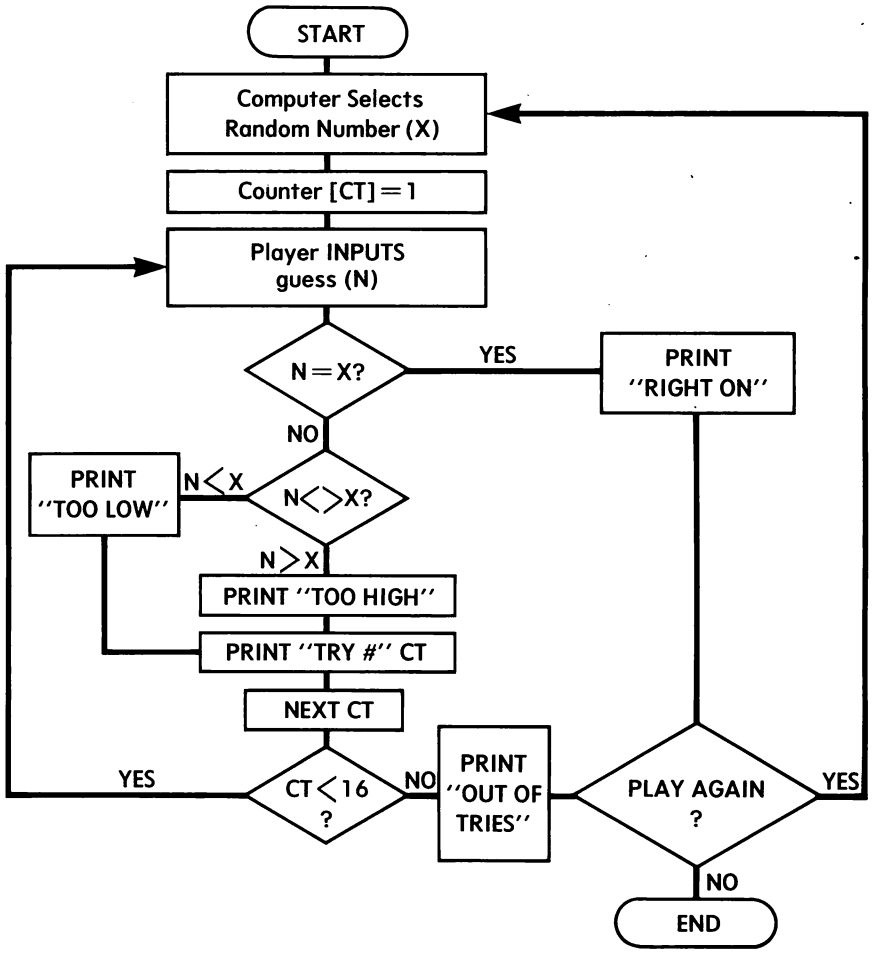


Figure 3–3: Guessing Game Flowchart.

```
10 REM GUESSING GAME
15 PRINT "♥":REM*SHIFT-CLR PRODUCES
THE HEART
20 X=INT(RND(1)*1000)+1:REM PICKS
RANDOM NUMBER
30 PRINT "I PICKED A NUMBER FROM 1-1000"
35 PRINT
40 PRINT "CAN YOU GUESS IT IN 15 TRIES?"
45 FOR CT=1 TO 15
50 INPUT "WHAT'S YOUR GUESS":N
60 IF N=X THEN PRINT "***RIGHT
ON!***":GOTO 110
```

```
70 IF N>X THEN PRINT "TOO HIGH. TRY AGAIN"
80 IF N<X THEN PRINT "TOO LOW. TRY AGAIN"
85 PRINT "THIS IS TRY NUMBER";CT
90 PRINT
95 NEXT CT
100 PRINT "YOU'RE OUT OF TRIES"
105 PRINT "MY NUMBER WAS";:PRINT X
110 PRINT "PRESS 1 AND RETURN TO PLAY
AGAIN"
120 PRINT
130 PRINT "ANY OTHER NUMBER AND RETURN
TO QUIT"
140 INPUT "ANOTHER ROUND . . ";N2
150 REM N2=DECISION TO START AGAIN
160 IF N2=1 THEN 15
170 PRINT "GOODBYE"
```

RND
INT

RanDom Numbers and the INTeger Function

To generate a random number from 1 to 1,000, the program uses an RND statement in conjunction with the INT function, which converts decimal numbers to integers. RND (1) selects a random number greater than zero and less than one. The following steps show how the statement in line 20 of GUESSING GAME was derived. The statements are in the immediate mode; you can go through the same steps as you read the explanation. Of course, each generated number will be different because the numbers are selected randomly.

You type: PRINT RND (1)
Computer: .838893164

Since the random number above is always between 0 and 1, multiply the number by 1,000 to make sure the desired number is between 1 and 1,000.

You type: PRINT RND(1)*1000
Computer: 931.229627

Use the INT function to drop the decimal part of the number.

You type: PRINT INT(RND(1)*1000)
Computer: 188

Since the RND(1) is never equal to 1, the largest possible integer will be 999. Even 999.999999 is converted to 999 by the INT function. To make it possible for the computer to select 1,000 randomly, you must add 1 to the result.

You type: PRINT INT(RND(1)*1000)+1
Computer: 973

Line 20 sets this expression equal to the variable x.

A little programming goes a long way. The game you just programmed was created with only a handful of BASIC's sixty different kinds of commands and statements. Check Appendix B for books devoted to the subject of BASIC programming. BASIC offers a broad horizon of opportunities for an aspiring programmer. If you learn to speak the C-64's language, you'll find it's a good listener and faithfully obedient.

Bits, Bytes, and Binary Numbers

The technical information in this box is useful for those who intend to pursue programming in earnest. It will also be helpful in working through the examples in the next chapter on programming graphics and sound. But you can very well write useful programs without giving a thought to bits, bytes, and binary numbers. The C-64 takes care of these matters automatically.

Each character you type on the keyboard is remembered by the computer as an eight-digit binary number. As noted earlier, binary numbers are composed of only two values — 0 (zero) and 1 (one). Each binary number is called a "bit" (short for *b*inary dig*it*), and each character you type is represented by a string of eight bits, or a *byte.*

A single binary digit has two possible values (2^1), and two binary digits have four (or 2^2) possible values. Three binary digits have 2^3, or eight, possible values. Let's extend this reasoning to the full eight bits (binary digits) in a byte:

Number of Bits	*Possible Values*
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256

Since an eight-digit binary number has 2^8 (or 256) possible values, the C-64 can only represent 256 different characters (letters, numbers, and special symbols) in a character set.

The C-64 represents whole numbers, or integers, in two bytes — two 8-bit words. That means there are 2^{16} (or 65,536) possible permutations of sixteen binary digits. Consequently, allowing for both positive and negative

numbers, the C-64 can represent any integer between −32,768 and +32,767 — a spread of 65,536 numbers (counting 0).

Larger numbers can be calculated, however, by using *floating point*, or decimal, numbers, which are represented in five-byte groups. The C-64 always calculates numbers as if they were floating point decimal numbers, but it converts them to integers (by dropping the digits after the decimal point and rounding *down*), if the programmer requests it. Positive and negative numbers between (and including) −999999999 and +999999999 can be represented in standard notation; larger numbers are expressed in *scientific notation* — as a decimal number multiplied by 10^n (where n = any number from −39 to +38). Decimal fractions smaller than .01 are also expressed in scientific notation. (See Your BASIC Calculator, pages 76–78, for an explanation of scientific notation.)

Let's examine how the computer stores a value in binary form in its memory. Each character is remembered as a series of eight bits, or one byte. (This is sometimes called the computer's *word-length*.) Below is a diagram of one byte, and RAM, as you know, has 65,536, or roughly 64K bytes altogether.

bit 7 bit 6 bit 5 bit 4 bit 3 bit 2 bit 1 bit 0

The line below the diagram shows that the bits are numbered from 0 (the first bit) to 7 (the last bit), starting from the right. The following list shows the decimal equivalent of placing a 1 in each position of the eight-bit binary number. If 1's are placed in more than one bit position, the decimal equivalent is determined by taking the sum of the decimal equivalents for each position with the value 1. (An example follows the list.)

00000001	=	1	(or 2^0)
00000010	=	2	(or 2^1)
00000100	=	4	(or 2^2)
00001000	=	8	(or 2^3)
00010000	=	16	(or 2^4)
00100000	=	32	(or 2^5)
01000000	=	64	(or 2^6)
10000000	=	128	(or 2^7)

How does the computer store the decimal number 44 in memory, using the model of a byte shown above?

0	0	1	0	1	1	0	0

2^5 (32) + 2^3 (8) + 2^2 (4) = 44

Programming graphics requires you to turn on (1) and off (0) specific bits with the POKE statement, and knowing the decimal equivalents enables you to accomplish this in BASIC.

Stop, Look, and Listen to Graphics and Sound*

*This chapter was contributed by Herb Moore, who has written extensively on the subject of graphics and sound.

The C-64 computer can create a wide variety of sounds and graphics images, making it an excellent tool for both entertainment and education. Adding graphics and sound to any type of program will make the program more interesting and enjoyable to run. This chapter shows you the kind of thing you can do with the computer itself, without having to purchase additional software. The examples of sound and graphics are programmed in BASIC.

Frankly, programming sound and graphics on the C-64 is not particularly easy, but your efforts will be repaid by the effects you can create. You may not understand every line of the sample programs at first, but using the preceding chapter as background and working through the examples slowly will give you a general idea of how to proceed. Several REM statements and some annotations have been included in the program listings to help clarify what each segment of the program does. You can acquire more expertise through practice, experimentation, and by consulting books devoted to programming.

Remember that accuracy is essential in typing the programs ahead. If you find yourself making numerous typing errors, refer to the section entitled Tips on Numbering Lines and Editing Programs in Chapter 3.

Let's Be GRAPHIC

There are sixteen colors that you can use on the C-64 screen, its border, or on a figure you have created on the screen. Text can

also be colored. One method for putting colored bars on the screen is described in Chapter 1. For a quick review of the colors available, try the following experiment:

1. Press CTRL, and while holding it down, type the number 1. The cursor should turn black.
2. Next, type your name and you'll see that the text appears in black.
3. Finally, press the CTRL key again, and while holding it down, type the numbers 2,3,4,5,6,7,8. Notice the color of the cursor changes each time you type a new number. When you type 7, it seems to disappear, but that's just because it turns the same color blue as the background screen.

But that's only half the colors available. To see the remaining eight colors, follow the same procedure. This time, use the COMMODORE key. Holding the COMMODORE key down, type the numbers 1,2,3,4,5,6,7,8. Notice that you get a different set of colors. (For a list of the colors available on the C-64, see Chapter 1, page 25.)

The area in which text is placed or figures appear is usually called the *background screen,* or *playfield.* There is also a border around this playfield or background screen. (Oddly enough, this area is usually called the border.) To see the border change colors, carefully type the following program. Be sure to press RETURN after each line, so that it will be entered into the computer's memory.

```
0 REM *** COLOR BORDER
5 REM *** COPYRIGHT HERB MOORE 1983
10 FOR EDGE=0 TO 15
20 PRINT CHR$(147)
30 POKE 53280,EDGE
35 MID=0
40 POKE 53281,MID
50 PRINT "THE BORDER COLOR IS";
60 PRINT EDGE
70 FOR PAUSE=1 TO 1000:NEXT PAUSE
80 NEXT EDGE
90 GOTO 10
```

Places each of the 16 colors in memory location 53280.

This loop makes the computer pause before changing colors on the screen.

After typing the program, type RUN <RETURN>. If all goes well, you should see the border of the screen change colors and text appear in the playfield telling you which value is being used to create the colors. Since the computer isn't using the CTRL key or the COMMODORE key, it enters values from 0 through 15 for color.

If you'd like to see the background screen (or playfield) change colors as well, press STOP, then LIST the program, and change line 35 to:

```
35 MID = INT(15*RND(1))
```

Also, remove line 50; just type 50 <RETURN>.

Now run the program again and this time you won't set the values for COLOR printed on the screen, but both the border and the playfield will change colors. *Note:* If the text is difficult to read, press STOP-RESTORE to bring back the normal color scheme. Then LIST and edit the program.

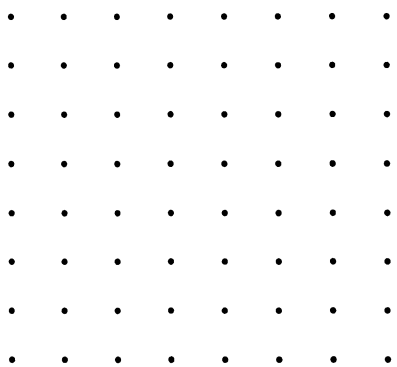
Screen Memory

The screen area where figures are displayed can be thought of as a grid consisting of forty columns (numbered 0 through 39) and twenty-five rows (numbered 0 through 24). This amounts to a total of 1,000 squares, or pixels, in which characters can be displayed. Each of these squares has a *memory location* associated with it. As the name would imply, this is a location in the computer's memory which contains information that tells the VIC-II graphics chip what to display on the screen. By changing the information in these locations, you can make changes in what is displayed on the screen. In BASIC, the POKE statement is used to place information in a specific memory location. Each square in the grid mentioned above can display one of 256 screen display characters in one of sixteen colors.

Pictures and "Pixels"

Each picture or image on your screen is composed of tiny illuminated dots. Each dot is known as a *pixel,* or "picture element." The C-64 has a high-resolution graphics mode that allows you to turn on or off any one of 64,000 possible dots on the screen. The more pixels, the sharper the picture.

The number of pixels in high-resolution mode is a function of the 1,000 screen memory locations, each of which is actually a grid of 8 by 8 bits. Thus, there are 64 pixels per character, or 64,000 on the total screen. "High-res" graphics is also called "bit-mapped" graphics, because it requires programming a *map* that accounts for every bit on the screen.



A single-character grid composed of 64 pixels. The screen contains 1,000 such character positions.

There are 1,000 memory locations (one for each square on the grid), and each can contain a number from 0 through 255. The number indicates which character is to be displayed at a given spot on the screen. Another 1,000 locations can contain values from 0 through 15 for the color of the character in a particular location.

Since a picture is worth a thousand words (or in this case a thousand bytes), type and run the following program to get a sense of the colors and characters available. The program should display all 256 characters on the screen. If you'd like to see the characters displayed on a different background, replace the 0 in line 40 with any number between 1 and 15. (Remember, you can press STOP-RESTORE to bring back the normal color scheme.)

```
0 REM **** SCREEN CHARACTERS
5 REM *** COPYRIGHT HERB MOORE 1983
15 REM *** CLEARS SCREEN
20 PRINT CHR$(147)
25 REM ** BORDER GREY
```

```
30 POKE 53280,12
35 REM ** SCREEN BLACK
40 POKE 53281,0
45 REM *** DISPLAY AND COLOR CHARACTERS
50 FOR CH=0 TO 255
60 POKE 1224+CH*2,CH
70 POKE 55496+CH*2,INT(15*RND(1)+1)
80 NEXT CH
90 GOTO 90
```

For the Programmer

When you turn on the C-64, the screen memory locations are 1024 through 2023 and the color memory locations are 55296 through 56295. This is shown in Figures 4–1 and 4–2 below:

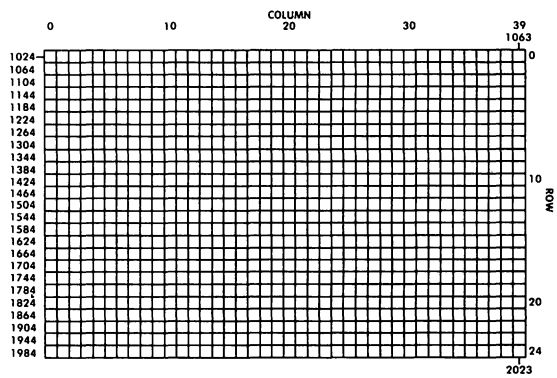


Figure 4–1: Screen Memory Map.

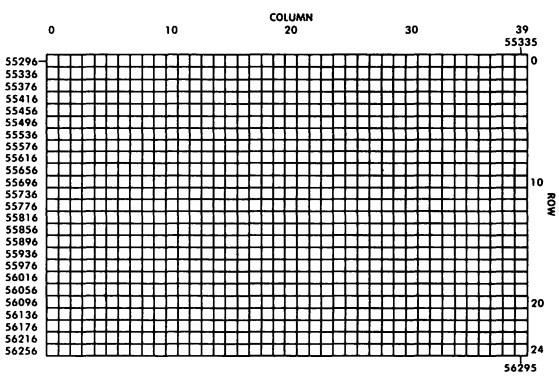


Figure 4–2: Color Memory Map.

The starting point for screen memory can be moved, but the locations for color memory are fixed. Screen memory is usually found in locations 1024 through 2023, in which case the following formula will place a character at a particular point on the 40-column screen:

POKE 1024 + X + 40 * Y,CH

(Where X is the value for column, Y is the value for row, and CH is the value for character.)

If the starting point of screen memory is moved to a new location, use that number in place of 1024 in the formula above.

The formula for coloring a particular pixel is:

POKE 55296 + X + 40 * Y,C

(Where X is the value for column, Y is the value for row, and C is the value for color.)

This will always be the same, since color memory is stationary.

Figuratively Speaking

Figures can be drawn on the screen using combinations of one or more of the screen display characters. Enter and run the following program for an example of how this feature is used. When you run the program, you will be prompted to enter values for a screen display character, and then the color. After you've done this, a triangle composed of the character and color you have chosen appears on the screen.

The screen characters and their corresponding values can be found in Appendix E of the Commodore *User's Guide* that came with your machine. The values for color, as you know, are from 0 through 15.

```
0 REM *** TRIANGLE
5 REM *** COPYRIGHT HERB MOORE 1983
15 REM **** BEGIN SCREEN & COLOR MEM
20 SM=1024:SC=55296
25 REM **** ENTER CHARACTER & COLOR ***
30 PRINT "PICK YOUR CHARACTER (0-255)"
35 INPUT CH
40 PRINT "PICK YOUR COLOR"
45 INPUT C
```

```
50 REM **** CLEAR SCREEN ******
55 PRINT CHR$(147)
60 REM *** COLOR SCREEN AND BORDER ***
65 POKE 53280,2
70 POKE 53281,0
100 REM *** DRAWS FIRST LINE
110 FOR X=8 TO 18
120 Y=26-X
130 GOSUB 510
140 NEXT X
200 REM *** DRAWS SECOND LINE
210 FOR X=18 TO 28
220 Y=X-10
230 GOSUB 510
240 NEXT X
300 REM *** DRAWS THIRD LINE
310 FOR X=8 TO 28
320 Y=19
330 GOSUB 510
340 NEXT X
490 GOTO 30
500 REM *** POKE CHARACTER & COLOR
510 POKE SM+X+40*Y,CH
520 POKE SC+X+40*Y,C
530 RETURN
```

(It is also possible to design your own set of up to 256 screen display characters. But that involves some rather tricky programming that we won't discuss here.)

Sprites

A unique feature of C-64 graphics is the ability to create on the screen objects called *sprites*. These are displayed in a special high-resolution mode, which divides the screen into 320 pixels across and 200 pixels down. The sprite itself consists of a grid of 24 dots across and 21 dots down. By turning each of these dots "on" or

"off" you create a figure that can then easily be moved around the screen. It is possible to place up to eight sprites on the screen at any one time, and to have others stored in memory. They can be colored. They exist independently of other objects on the screen, and they can be made to move in front or back of each other. You can detect if one sprite collides with another, and you can also expand the size of sprites. Yes, this is the stuff that video games are made of.

Creating and using sprites doesn't require a great deal of sophisticated knowledge once you understand the fundamentals. There is, however, a rather substantial amount of detail involved. So with the space available in this chapter, the emphasis will be on demonstrating how sprites work rather than on explaining how to program them.

A brief description of how sprites are created follows. Refer back to Bits, Bytes, and Binary Numbers in Chapter 3 if you need help understanding the discussion. It is not essential that you comprehend every detail of this description, so you can skim the next couple of pages, and then type the SPRITES IN ACTION program. The pages that follow the listing explain how the program works.

As noted above, each sprite is composed of a grid 24 squares across and 21 squares down. (See Figure 4–3, below.)

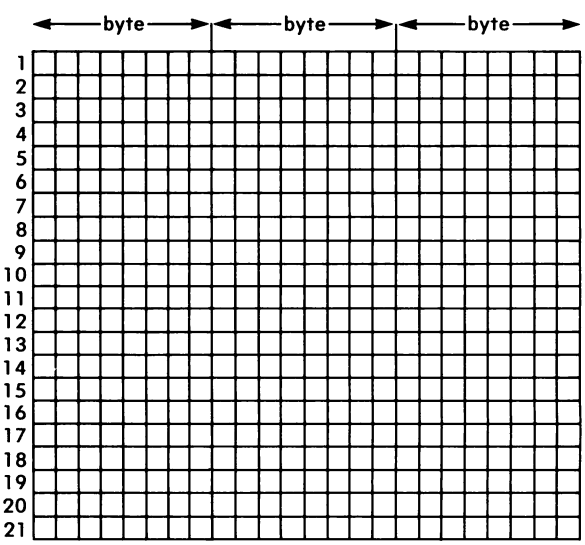


Figure 4–3: Sprite Grid.

Each line of the grid consists of three bytes, each byte being eight squares (or eight bits) long. Since the grid is 21 lines down, this amounts to 63 bytes (21 times 3) of information needed to create a sprite. In creating a sprite, you have to decide which bits are to be turned on or off in each byte.

For example, suppose you want to create on the screen the sprite shown in Figure 4–4 below:

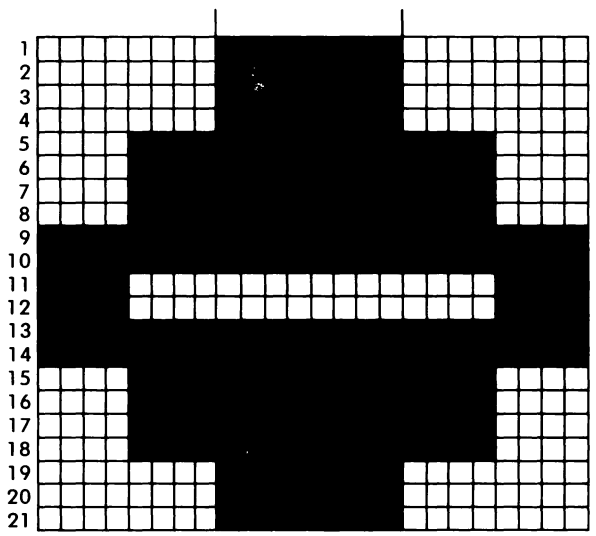
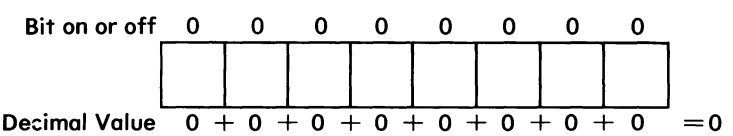


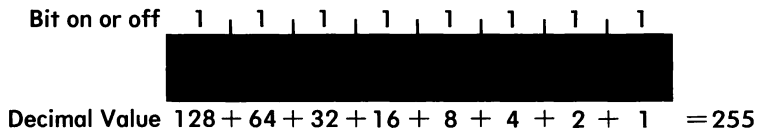
Figure 4–4: Spaceship.

The information for the first row is stored in three bytes of RAM:

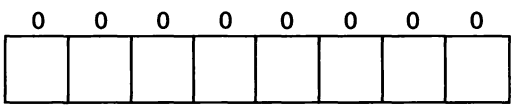
You can see that all of the bits in the first byte of this row are turned off, so the decimal value for that byte would be 0.



In the second byte of the first row, all of the bits are turned on, so its decimal value is 255.



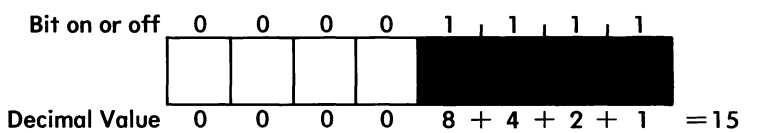
The value for the third byte of the first row is again 0 since all of the bits are turned off.



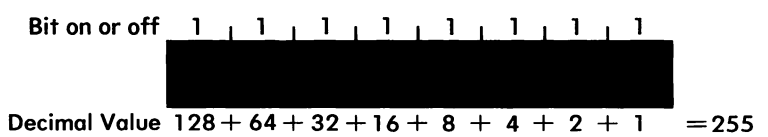
Rows 2, 3, and 4 of the figure repeat the same pattern as row 1 for each of the three bytes.

Row 5, however, is a little different. The pattern for the three bytes of row 5 looks like this.

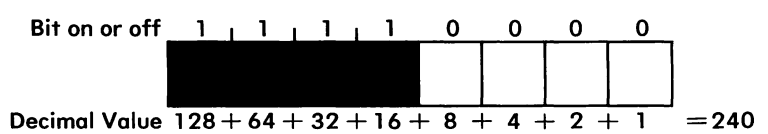
The first byte of row 5 has a decimal value of 15. It looks like this:



The second byte of row 5 would have a decimal value of 255 again, since all the bits are turned on:



The third byte of row 5 has a decimal value of 240. It looks like this:



The information for each of the 63 bytes for a given sprite must be calculated in this way. Once this is done, you can put

this information into a program that tells the computer to display the sprite on the screen as you have defined it. The sprite shown in Figure 4–4 is included in the program listed below. Notice that the values we calculated above are entered in the first few lines of the DATA statements beginning at line 210.

As mentioned earlier, an exciting feature of sprites is their ability to move around and detect when they've run into each other. This is one of the features that will be demonstrated in the experiments ahead.

The program listing for SPRITES IN ACTION is pretty long, because the computer needs many commands to color the sprites, move them around, and so on. Type very carefully when entering this program.

Note: You can save some time by leaving out the REM statements. REM statements, as you learned in Chapter 3, don't affect how the program runs. They simply clarify various parts of the program. However, if you intend to save this program, you should probably include the REM statements to help you comprehend the listing when you look at it later.

```
0 REM *** SPRITES IN ACTION
5 REM *** COPYRIGHT HERB MOORE 1983
15 REM *** CLEAR SCREEN
20 PRINT CHR$(147)
25 REM *** START OF DISPLAY CHIP
30 V=53248
35 REM *** ENABLE SPRITES
40 POKE V+21,14
45 REM **** MEM LO **** SPRITE 2
50 POKE 2041,13:POKE 2042,14
55 POKE 2043,15
60 PRINT "COLOR SPRITE 1"
65 INPUT C1:POKE V+40,C1
70 PRINT "COLOR OF SPRITE 2"
75 INPUT C2:POKE V+41,C2
80 PRINT "COLOR OF SPRITE 3"
85 INPUT C3:POKE V+42,C3
90 REM *** COLOR BORDER & SCREEN
95 POKE 53280,7:POKE 53281,0
```

```
100 REM **** READ-DATA FOR SPRITE 1
110 FOR N=0 TO 63
115 PIX1=255
120 POKE 832+N,PIX1
125 NEXT N
150 FOR X=20 TO 175
155 Y=145
160 POKE V+2,X
165 POKE V+3,Y
170 NEXT X
200 REM **** READ-DATA FOR SPRITE 2
210 FOR N=64 TO 128
215 READ PIX2
220 POKE 832+N,PIX2
225 NEXT N
260 DATA 0,255,0,0,255,0
261 DATA 0,255,0,0,255,0
262 DATA 15,255,240,15,255,240
263 DATA 15,255,240,15,255,240
264 DATA 255,255,255,255,255,255
265 DATA 240,0,15,240,0,15
266 DATA 255,255,255,255,255,255
267 DATA 15,255,240,15,255,240
268 DATA 15,255,240,15,255,240
269 DATA 0,255,0,0,255,0
270 DATA 0,255,0,0,255,0,0,255,0
275 RESTORE
280 FOR Y=20 TO 165
285 POKE V+4,X:POKE V+5,Y
290 NEXT Y
300 REM **** READ-DATA FOR SPRITE 3
310 FOR N=129 TO 193
315 PIX3=255
320 POKE 832+N,PIX3
325 NEXT N
350 FOR X=20 TO 190
355 Y=160
360 POKE V+6,X
```

```
365 POKE V+7,Y
370 NEXT X
590 FOR PAUSE=1 TO 1000:NEXT
595 GOTO 15
```

Type RUN <RETURN>. If you get an error message, check your typing and try again. Once you get the program working, don't erase it since you'll be adding two segments to it soon. If you have a Datassette recorder or disk drive, you may wish to save the program before going on.

It's time to watch these famous C-64 sprites in action. When you run the program, the following prompt appears on the screen:

```
COLOR SPRITE 1
```

Type a number from 1 to 15 and <RETURN>. Then another prompt appears:

```
COLOR SPRITE 2
```

Again, type a number from 1 to 15 and <RETURN>. And finally:

```
COLOR SPRITE 3
```

Guess what you do next? You got it! Type a number from 1 to 15 <RETURN>. Now, the action begins. Watch closely. The first sprite is a square that moves from the left side to the center of the screen. Next, a little spaceship drops from the top of the screen, moves in front of the square, and then becomes stationary. Finally, a third sprite, also square, moves from the left of the screen in front of the other two sprites, until it comes to rest in a permanent screen position. The color of the sprites depends upon the numbers you've entered in response to the prompts. Once the cycle is complete, you can change the color of the sprites by going through the series of three prompts again, and entering different numbers between 1 and 15 for the color of each sprite.

If you want the sprites to remain the same color but move again on the screen, just type

<RETURN> <RETURN> <RETURN>

when the prompts appear on the screen.

Notice that the background screen is black. You may remember that the number zero colors the screen or figures black. Try entering a zero for the color of one of the sprites and it will move "invisibly" on the screen. But if you look carefully, you'll see its shadow when the three sprites reach their stationary positions on the screen. The idea of moving invisible objects around on the screen presents some provocative possibilities, doesn't it? (Then again, maybe it doesn't.)

Before continuing, you may wish to spend a little time experimenting with this program to see how sprites look in different colors.

Expanding and Contracting Sprites

Whether or not sprites can expand their consciousness hasn't been established, but it is possible for them to expand their size. They can expand in length only, in width only, or in both length and width. They can also return to their original size. You can see a sample of these expansion and contraction capabilities by adding the following lines to the SPRITES IN ACTION program, which should now be in memory or saved on disk or cassette. Remember, all you need to do is type the new lines — followed by <RETURN> — and let the computer order the lines within the program. Type LIST <RETURN> to see the complete program in memory with the lines in order.

```
500 REM *** EXPAND SPRITES
510 POKE V+29,4
530 GOSUB 900
550 POKE V+23,2:POKE V+29,2
570 GOSUB 900
580 POKE V+23,0:POKE V+29,0
595 GOTO 15
900 REM *** SUBROUTINE TO PAUSE
910 FOR PAUSE =1 TO 1000:NEXT
920 RETURN
```

When you run the program, notice what happens to each of the sprites after the values for colors are entered. First, they

go through the previous sequence of moving from the edges of the screen to stationary positions in the center of the screen. Then, the spaceship sprite gets wider. Next, the square, which is sprite 1, gets both wider and longer, covering the other two figures.

The Sprites Collide

You've seen sprites exhibit a trait similar to humans — occasionally trying to upstage each other and block each other out of the picture. Sprites can also be taught to detect when they collide with each other or with the edge of the screen. Add the following lines to the SPRITES IN ACTION program, and you'll see sprites 1 and 2 turn white when they detect a collision. After a moment, they return to the colors you've selected, and the sequence goes on. You realize, no doubt, that collision-detection is the essence of programming video games that require the player to shoot down targets or to elude and evade pursuing objects.

```
295 IF PEEK(53278) AND 6=6 THEN 600
600 REM *** SPRITES WHITE ON COLLISION
610 POKE V+40,1:POKE V+41,1:POKE
V+42,1
620 GOSUB 900
640 REM *** CLEAR COLLISION FLAGS
645 DD=PEEK(53278)
646 POKE V+40,C1:POKE V+41,C2
650 REM *** COLOR SPRITES AGAIN
655 POKE V+40,C1
660 POKE V+41,C2
665 POKE V+42,C3
690 GOTO 300
```

To clearly see the effects of this addition to the program, enter values from 2 through 15 when prompted to do so for the color of each of the sprites. The value for the color white is 1, and that's the color they'll be turning, so the collision is less dramatic if you color them white from the start. Another interesting effect is created if you give one or more of the sprites a value of 0. You won't see it on the screen unless it passes in front of another

sprite, because it's the same color as the background screen. But it appears briefly when one and two sprites collide and they all turn white. Consider giving a value of 0 to some of the sprites in this program. You might come up with some tricks that haven't been mentioned here. Have fun experimenting.

Sounding Off

Over the past few years there has been a growing awareness of the process of sound synthesis. This was previously thought to have been the domain of a few mad scientists, musicians, and composers who used very elaborate equipment, but with the advent of computers like the C-64, it is possible for ordinary people to synthesize music and sound effects inexpensively and in their own homes. In this section, there are discussions of how sounds are shaped and some sample programs that let you enter different parameters for shaping sounds. As you did with the graphics programs earlier in this chapter, type carefully when entering the programs in this section. If you have a Datassette recorder or disk drive, you may want to save one or more of these listings.

Just as in the production of video, the C-64 uses a separate chip to create sounds. It is called the MOS 6581 Sound Interface Device, or simply SID. Using SID, it is possible to create a variety of types of sound for three independent voices with your C-64. The sounds produced by SID are normally sent to the speaker of your TV or monitor. For better reproduction, you can purchase an inexpensive gadget called the Sound Box made by Human Engineered Software (HES) of Brisbane, CA. It enables you to send the sound signals from your C-64 directly to your stereo system, tape recorder, or other similar devices. The signal will still be monaural, but if you have a good sound system, it can greatly improve the quality of the music and sound effects you create.

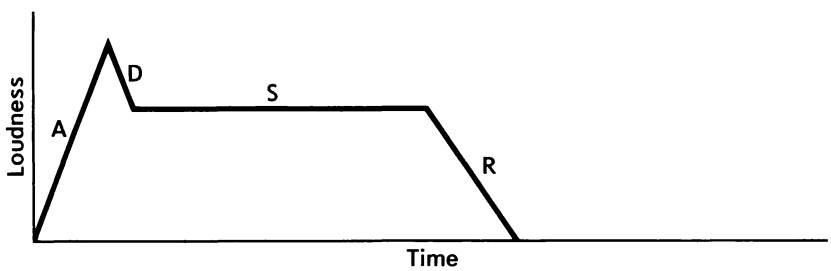
Shaping Sounds

A note with a given fundamental frequency, for example middle-C, can sound quite different when played on different instruments, such as a piano, flute, or violin. The quality that gives the

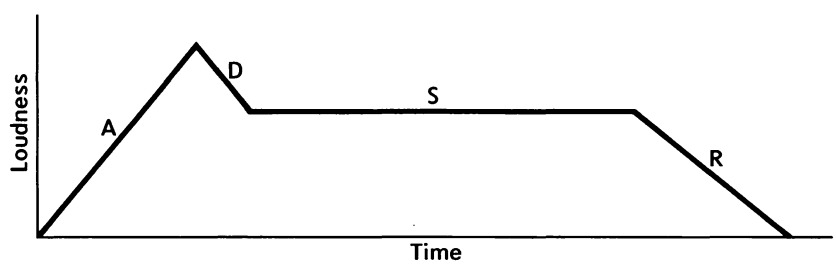
same note its unique characteristic is often called the *shape* of the note. Many factors contribute to the shape of a particular sound. Two of the most important are its *envelope* and its *waveform*. Since the SID chip is designed to allow for considerable control of these two parameters, it will be helpful for you to understand them.

Envelope, usually called the ADSR (for attack, decay, sustain, release) envelope, describes what happens to the loudness of a note through time. The *attack* of a note refers to the speed with which it reaches its maximum loudness from silence. A piano or a guitar has a fast attack, while a bowed violin or a flute has a slower attack. The *decay* of a note refers to the speed with which the note falls off to some level of loudness, which it maintains for some duration. The *sustain* of a note is the length of time it is held at a given volume. Instruments like flutes or violins generally give the player the opportunity to have greater control over the sustain of notes than does the piano or guitar. Finally, *release* refers to the time it takes for the note to fall off to silence after it has ceased to be played.

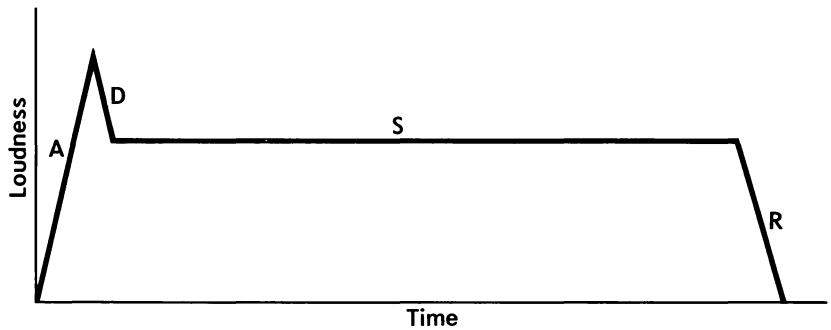
Here are some diagrams of a few different-shaped ADSR envelopes to help you visualize the idea.



This one represents a relatively fast attack and decay, a medium sustain, and a relatively fast release.



This one has a slower attack and decay than the one above; it also has a somewhat longer sustain and a slower release.



This one shows a note with a very quick attack. The decay is also very fast, but the amount of decay is very slight. That is, it only falls off from the peak loudness before it reaches a rather long sustain period. The release part of the diagram indicates that the note quickly clips off at the end.

The guitar is a good instrument to use to explain an ADSR envelope. Imagine a player who holds his or her finger down on a string at some fret (one of those little bars on the neck of the guitar) and plucks the string. The result will be a rapid attack to maximum volume followed by a gradual decay to a briefly sustained note. If the player holds his finger on the fret, there will be a very slow release. On the other hand, if the player quickly removes his finger from the string and "damps" it, there will be a very rapid release. So, the player can have some influence over the ADSR envelope depending on how hard the string is plucked and how quickly it is damped or released.

Similarly, a flautist and a violinist can have a limited influence upon the ADSR envelope characteristic of their respective instruments. As mentioned before, flutes and violins allow for greater control of sustain than acoustic guitars or pianos. The electric guitar enables the player to have much greater control of the ADSR envelope.

With computer-generated sounds, such as those produced by the C-64 SID chip, you are able to exert maximum control over the ADSR envelope. You'll get a chance to explore the parameters of ADSR envelopes using the NOTE SHAPE program on the opposite page, but first let's have a look at *waveforms*.

When you hear a note of a given frequency, like middle-C, you hear not only the fundamental frequency of that note but you also hear a number of other frequencies called *harmonics*, or *partials*, of this frequency. It is the relative strength or weakness of these harmonics that determines what is often called the timbre of an instrument. Thus, a sustained middle-C sounds quite different when played on a cello and a flute, regardless of the ADSR envelope. The pattern of harmonics for a given sound is usually called its waveform.

Electronic synthesizers create different timbres by controlling the waveform. This is done by making different combinations of harmonics louder or softer relative to each other. The C-64 SID chip is able to produce four different waveforms. These are called triangle, sawtooth, pulse, and noise.

It is not necessary that you fully understand every detail of how ADSR envelopes and waveforms work, but the above discussion should help you get some idea of how you are effecting the shape of the notes in the following program. (BASIC allows you to drop the closing quotation mark of a PRINT statement, as in lines 100, 120, 140, and 160.)

```
0 REM *** NOTE SHAPE
1 REM *** ATTACK-DECAY AND WAVEFORM
5 REM *** COPYRIGHT HERB MOORE 1983
10 PRINT CHR$(147)
100 PRINT "ATTACK AND DECAY OF NOTE
110 INPUT AD:PRINT
115 PRINT AD:PRINT
120 PRINT "SUSTAIN-RELEASE OF NOTE
125 INPUT SUS:PRINT
130 PRINT SUS:PRINT
140 PRINT "WAVEFORM
145 INPUT WAV:PRINT
150 PRINT WAV:PRINT
160 PRINT "NOTE DURATION
165 INPUT DUR:PRINT
170 PRINT DUR:PRINT
200 REM **** SETS VOLUME
210 POKE 54296,15
```

```
215 REM **** ATTACK AND DECAY
220 POKE 54277,AD
225 REM **** SUSTAIN - RELEASE
230 POKE 54278,SUS
400 REM **** HIGH-LOW FREQ. OF NOTE
410 FOR N=1 TO 13
420 READ HF
430 READ LF
440 REM **** HIGH-LOW FREQ. OF NOTE
450 POKE 54273, HF:POKE 54272,LF
460 POKE 54276,WAV
470 FOR PAUSE= 1 TO DUR:NEXT
480 POKE 54276,0
485 FOR PAUSE= 1 TO DUR:NEXT
490 NEXT N
495 POKE 54296,0
500 REM ****
510 FOR PAUSE =1 TO 1000:NEXT
610 DATA 21,154,17,37,25,177
620 DATA 72,169,57,172,38,126
630 DATA 115,88,137,43,8,147
640 DATA 12,216,10,205,17,37,14,107
690 RESTORE
710 GOTO 10
```

If you've entered the NOTE SHAPE program correctly, here's how it should work. First, you'll see the prompt:

```
ATTACK AND DECAY OF NOTE
?
```

When you see this, type 16 <RETURN>. Next, you'll see the prompt:

```
SUSTAIN-RELEASE OF NOTE
?
```

In response to this prompt, type 127 <RETURN>. This should bring the next prompt up on the screen:

```
WAVEFORM
?
```

For this one, type 17 <RETURN>. The final prompt you'll get is:

```
NOTE DURATION
?
```

Type 50 <RETURN> for this one.

This should result in a brief melody in which each note has a kind of echoing flute sound with a rather short duration. The value you entered for NOTE DURATION determines how long each note plays.

You can take advantage of a little quirk of this program and hear the melody again by pressing <RETURN> without typing a number each time one of the prompts appears on the screen. This will only work after you've been through the cycle once and entered values for each parameter. If you press <RETURN> without entering a value for a parameter, the previously entered value will be printed on the screen. You can also use this technique to change only one parameter and leave the others the same. For example, try the following little experiment.

First, run the program and enter the following values for each of the parameters of the note shape:

ATTACK AND DECAY OF NOTE:	65 <RETURN>
SUSTAIN-RELEASE OF NOTE:	0 <RETURN>
WAVEFORM:	17 <RETURN>
DURATION:	200 <RETURN>

Now when the prompts appear on the screen, instead of typing a number for each parameter just change the duration:

ATTACK AND DECAY OF NOTE:	<RETURN>
SUSTAIN-RELEASE OF NOTE:	<RETURN>
WAVEFORM:	<RETURN>
DURATION:	75 <RETURN>

Here's a chart with some suggested values for ADSR, SUSTAIN-RELEASE, WAVEFORM, and DURATION to create some different shapes for notes. Portions of this chart were adapted from the chart found on page 162 of the C-64 *User's Guide*.

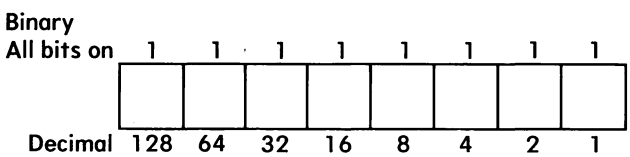
Attack-Decay	Sustain-Release	Waveform	Duration
96	0	17	200
9	0	33	200
102	0	17	200
96	0	33	200
17	0	17	50
33	128	33	25
33	136	33	300
142	136	17	8000
248	240	33	5000

Try some of these combinations in the NOTE SHAPE program, then make up some of your own.

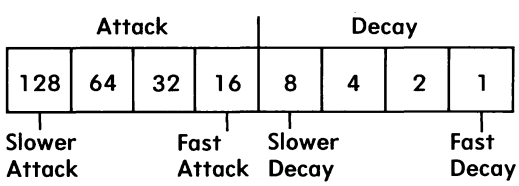
More on Bits and Bytes

To understand clearly how the C-64 SID chip handles ADSR envelopes, you'll need to have another look at bits and bytes.

Remember that the decimal values for one byte in which all bits are turned on are:



The C-64 uses one byte to set both the attack and decay of a single note. It also uses one byte to set both the sustain and release of a single note. Here's a diagram of how attack and decay are handled by one byte:



The number 65, which when you first ran the program you entered for attack and decay, was reached by adding: 64 (medium attack) + 1 (low decay) = 65.

It may not be apparent at first, but the higher the attack rate you give the note, the slower the attack. To see this run the program and enter these values:

ATTACK-DECAY:	128
SUSTAIN-RELEASE:	240
WAVEFORM:	17
NOTE DURATION:	200

Listen carefully to how the beginning of each note in the melody sounds. Remember, to hear it again, hit <RETURN> four times in a row without entering any values.

Next, when you get the prompt

```
ATTACK AND DECAY OF NOTE
?
```

type 16 <RETURN > <RETURN> <RETURN> <RETURN>.

Notice that the low value for attack gives the note more of a "pop" as it begins, while the higher value for attack gives the note a slower beginning.

Now let's play around with the decay of the notes in the same melody. Run the program and enter the following values:

ATTACK-DECAY:	65
SUSTAIN-RELEASE:	0
WAVEFORM:	17
DURATION:	200

Here we have a note with no sustain and a medium attack with fast decay: 64 (medium attack) + 1 (fast decay) = 65.

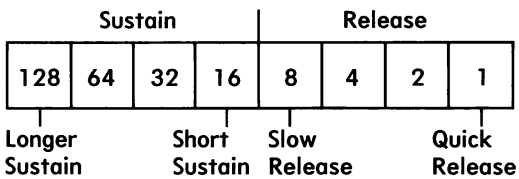
Remember that, in addition to the speed with which a note decays, another important factor is the sustain level to which it decays from maximum loudness. In this case, since there is no sustain, the note begins relatively slowly, but as soon as it reaches maximum loudness, it drops off quickly. Play the melody with this note shape a few more times by responding to each prompt with <RETURN>.

After observing how the notes sound with a fast decay, enter the value 79 when you see the prompt:

```
ATTACK AND DECAY OF NOTE
?
```

Now we have: 64 (medium attack) + 15 (slow decay) = 79. Notice how much more slowly the notes in the melody fall off with the slower decay.

Sustain and release work together on the same byte in much the same way as attack and decay work together on the same byte. Here's a diagram of the byte that controls sustain and release:



To hear a note with medium attack and decay and with a long sustain and quick release, enter the following values:

ATTACK-DECAY:	72
SUSTAIN-RELEASE:	15
WAVEFORM:	17
DURATION:	500

Next, you can see a relationship between decay and sustain. This note drops off quickly (decay) as soon as it reaches peak loudness, because there is no sustain. But it has a slow release, so the very end of the note is softer than the previous note.

These should give you a sense of how the C-64 can shape sounds. (For a more detailed exploration of this topic look for *Sound and Graphics for the Commodore 64* by Herb Moore, John Wiley & Sons, Inc., 1984.)

Mixed Sound and Graphics

In the beginning of the chapter, you were promised a sample of combined sound and graphics. The following program is a simple example that demonstrates how sound and graphics can be coordinated. When you run this program, you'll see the screen go blank and then hear a slowly increasing low note. This is followed by a sequence of sounds gradually increasing in pitch and accompanying the movement of an abstract figure that descends from the top to the bottom of the screen.

```
0 REM *** ABSTRACTION
1 REM **** SQ SPRITE WITH SOUND
5 REM *** COPYRIGHT HERB MOORE 1983
15 REM ** CLEAR SCREEN START VIDEO
20 PRINT  CHR$(147)
25 V=53248
30 REM *** ENABLE SPRITE
35 POKE V+21,4:REM ** ENABLE SPRITE 2
40 POKE 2042,13
100 REM *** FADE IN LOW NOTE
110 FOR L=1 TO 15
120 POKE 54296,L :REM * VOLUME ON
130 POKE 54277,136:REM * ATTACK-DECAY
140 POKE 54278,248:REM * SUS-RELEASE
150 POKE 54276,17:REM * WAVEFORM
160 POKE 54273,7     :REM * HIGH FREQ
170 POKE 54272,53    :REM * LOW FREQ
180 FOR PAUSE=1 TO 100:NEXT
190 NEXT L
200 REM *** CREATE AND DISPLAY SPRITE
210 FOR N=0 TO 62 STEP 2
215 POKE V+23,4:POKE V+29,4
220 P=16
230 POKE 832+N,P
240 NEXT N
300 REM *** LOCATES SPRITE
301 POKE 54296,15:REM * VOLUME ON
310 FOR Y=25 TO 250
320 X=170
325 PITCH=251-Y+INT(5*RND(1)*5)
330 POKE V+4,X:REM ** X COORDINATE
340 POKE V+5,Y:REM ** Y COORDINATE
350 REM *** CREATE SOUNDS - VOICE 1
370 POKE 54276,17:REM * WAVEFORM
375 POKE 54273,PITCH:REM * HIGH FREQ
380 POKE 54272,PITCH:REM * LOW FREQ
395 NEXT Y
```

```
400 REM *** TURN SOUND OFF
410 POKE 54276,0:POKE 54296,0
420 POKE 54296,0
450 FOR PAUSE = 1 TO 1000:NEXT
500 GOTO 100
```

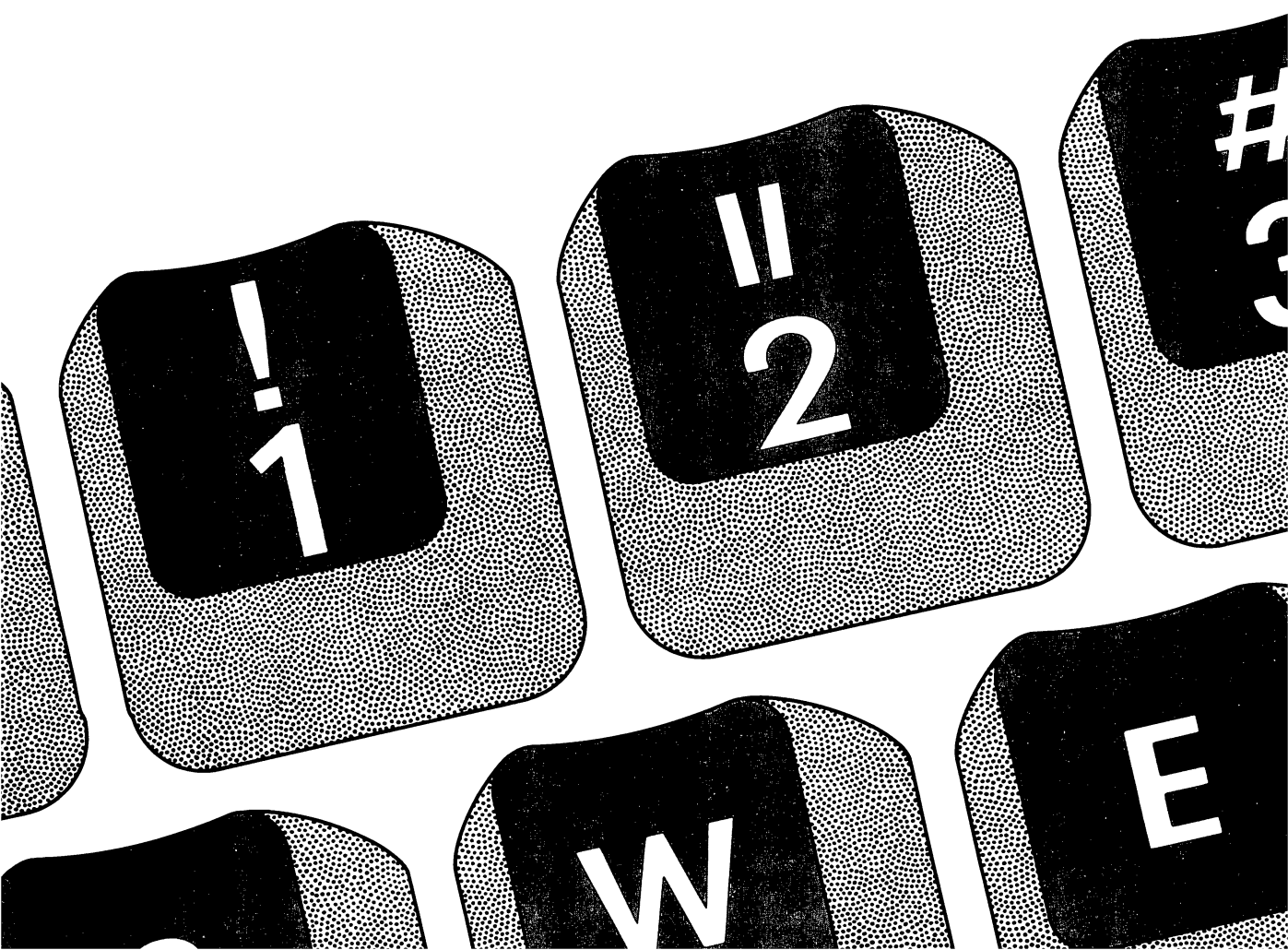
Here's a little experiment you can perform with this program running. When the figure gets to the center of the screen, press STOP. The figure stops moving and the pitch of the note becomes constant. Now type CONT <RETURN> and the program continues.

The programs in this chapter are intended to demonstrate a few of the fundamental concepts necessary for understanding how the C-64 creates sound and graphics. The next time you play a C-64 game, you'll know that the spaceship you see is a sprite, and you'll have some idea of how it is able to move around and respond to collisions. You'll also be better able to appreciate the subtleties (or lack of them) in the sound of its engines.

A new ROM cartridge for the C-64 called Simon's BASIC should be available by the time you read this book. This cartridge provides you with a much more "friendly" set of commands with which to explore the sound and graphics capabilities of the C-64. Commands such as REC, for RECTANGLE, CIRCLE, and ARC, as well as other graphics commands allow you to PLOT a point or DRAW a line more simply than you can with numerous POKEs. Likewise, you'll appreciate such commands as WAVE and ENVELOPE, which facilitate your efforts to design sounds.

2

COMPUTER APPLICATIONS AND HOW TO MAKE THEM WORK FOR YOU



Word Processing with EASY SCRIPT*

Conquering the Paper Tiger

Writing something that you expect others to read is usually a time-consuming and intimidating activity. Nobody minds jotting down a shopping list to take to the market or notes in an appointment calendar, but once you start composing sentences for the eyes of your clients, your college professor, your boss, or a distant sweetheart, the anxiety mounts. Writing for readers means being organized, clear, convincing, grammatical, and of course never having to say you're sorry for what you wrote. It's not easy, not even for professional writers.

The reason that writing is so time-consuming can be summed up in one word: revising. That word covers a multitude of miseries. One obvious reason to revise is to look for (and fix) simple spelling and grammatical mistakes. But there are also times when you're reading over your final draft, and suddenly realize that there was a better way of saying it after all.

What happens when you decide that the paragraph on page three really belongs on page one? How can you write business- or sales-oriented letters to three dozen clients, sending each the same basic letter but tailoring the letters to the special needs of each client? These changes require *revisions*, and revising means retyping. That's what makes the job so tedious.

Now you know the problem. Word processing is the solution. It won't turn you into a best-selling novelist, but by re-

*EASY SCRIPT is available through Commodore dealers.

ducing the tedious and tiring task of retyping, word processing gives you more time to improve your work — whatever the type of writing involved.

How Word Processing Works

In principle, word processing is nothing new. Any machine that transfers text to paper is "processing words" in the broadest sense of the phrase. By this definition, even your typewriter is a primitive word processor. But when word processing is mentioned today, it refers to computers that allow you to write and revise your text on a display screen (such as your TV set) before you ever commit it to paper. Word processing began with the realization that computers can manipulate letters, words, and sentences as efficiently as they can handle numbers.

In a limited way, the "magic tablets" designed for children behave like a display-based word processor. These drawing boards are covered with a see-through, plastic sheet. When you lift up the plastic, the lines underneath instantly disappear, and the child can draw something else. If you've never used a word processor before, you'll feel a child's sense of awe at the magic of word processing.

Word processing turns your screen into an electronic writing pad that can rearrange your text in any form you request — instantly. You can store what you've written on diskettes and work on it again later. When you're ready, you can print it automatically, and at speeds far greater than any typist can achieve.

Let's look behind the magic of word processing. Figure 5–1 on page 132 is a graphic overview of how your computer functions as a word processor. Word processing programs usually perform four basic functions:

- entering and revising text;
- formatting the text for printing;
- saving and retrieving the text from tape or disk; and
- printing the text.

In the next few pages, we will see how these basic elements fit together.

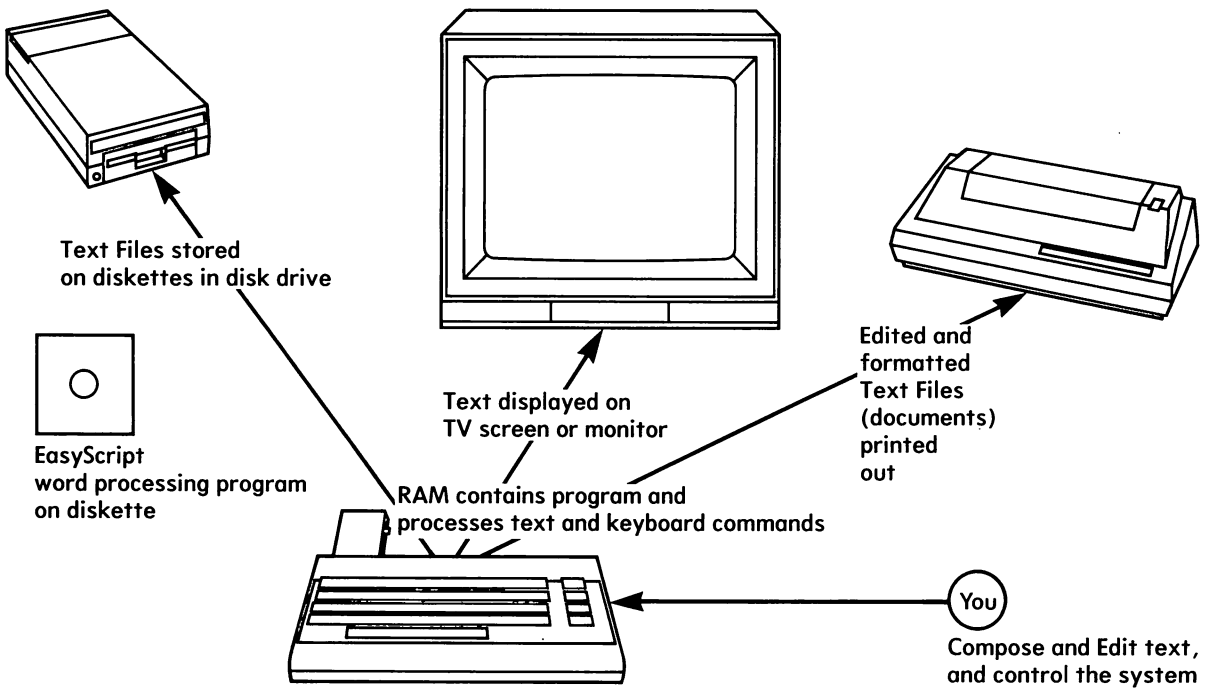


Figure 5–1: A Word Processing System.

Getting It Down on "Paper"

After loading the software — the program that turns the C-64 into a word processor — you can use the computer's keyboard as you would a typewriter to type the text. Although you're actually entering "data" into RAM memory that's subsequently displayed on your TV screen, the computer responds so fast that you'll feel as if you're typing directly onto the screen.

Most people type faster on a word processor than on an electric typewriter because there's no carriage return to wait for at the end of each line. As you type, the text wraps around to the next line automatically. You press the computer's RETURN only at the end of paragraphs, to skip a line, or to confirm a command.

After typing in the text, you can use the editing commands to revise it. The word processor rearranges the text on your electronic writing pad. You can delete individual characters, punc-

tuation marks, and paragraphs, words, sentences, or whole blocks of text anywhere in your document.

Making It Look Good

There are also word processing commands that tell the computer how you want the text to be printed. This is called *formatting* because it sets the page formats. Formatting includes specifying the margins, type size, spacing between lines, when to justify the right margin, and many more options.

You can revise both the text itself and the page formats on the screen by pressing a few keys. The drudgery of erasing, cutting and pasting sentences in new locations, and whiting out unwanted characters with liquid erasers will soon be as unthinkable as making copies of your important letters by hand, rather than with a photocopy machine.

Saving Your Work

Although you can edit the text on your TV screen — the text in memory — without even turning on your cassette recorder or drive, there are two compelling reasons to use an external memory device. First, as was explained in the beginning of Chapter 2, the computer's memory is erased when it's turned off. Whatever you've spent those long hours grinding out and polishing up is going to disappear as soon as you throw the switch, *unless* you save it.

Second, if you're writing a very long document (a thesis, novel, company report, and the like), you won't be able to fit the entire document into the computer's memory at one time. EASY SCRIPT limits you to 764 lines of text (at forty characters per line) in memory at one time. Printed and double-spaced, that amounts to about thirty typed pages. When you've reached the limit, you have to *save* that text in an external memory device before writing more.

Finally, there's the writer's famous "desk drawer," where unfinished works await the final touches that will turn them into modern classics — or at least make them publishable. Even for nonwriters, returning to a piece of writing for a second pass usu-

ally improves the final result. In word processing, the disk drive or cassette recorder becomes the desk drawer.

EASY SCRIPT is available on both cassette and diskette. Disk drives are far superior to tape because of their speed and random access. One EASY SCRIPT command reads text from a stored file and writes it into a separate file in memory. Moving text from one file to another is often clumsy and sometimes impossible using tape. This introduction to EASY SCRIPT is written for a disk-based system, but most of the practice examples apply equally to tape. The main difference is that tape memory is slower. If you're using the Datassette recorder, you can practice the examples ahead and consult the EASY SCRIPT *User Guide* for instruction on how to save onto tape. (The *User Guide* occasionally mentions a cartridge, because EASY SCRIPT was once distributed on a cartridge and the manual hasn't been updated. Disregard these references.)

Printing Hard Copy

Text that is transferred from the computer's memory to the printer and onto paper is called *hard copy*. Creating hard copy is usually the last step in word processing. Why commit your work to paper until you're completely satisfied with it? It's much harder to change the printed document than the "soft" electron-generated images on your TV screen.

In word processing, choosing the right printer is crucial. The categories of printers are impact, nonimpact, dot-matrix, and letter-quality. Impact style printers are similar to typewriters — a mechanical element strikes the paper and presses an ink (or carbon) image on it. In word processing, these are more commonly used than laser and ink-jet — nonimpact — printers, which require specially treated paper. But the choice between dot matrix and letter-quality printers should be based on your specific needs. (See Table 5–1.)

Getting Started with EASY SCRIPT

EASY SCRIPT is an excellent word processor, especially when compared with comparably priced software. It has the capabilities

Dot-Matrix	Letter-Quality
Characters are formed by a pattern of dots striking the ribbon against the paper; the dots are arranged in a grid. The dot matrix should have a density of 7 × 9 to create correspondence-quality characters.	Characters are formed by the "petals" of a wheel. Each petal impresses the ribbon with a fully formed or "whole" character.
Print quality is not usually considered acceptable for formal correspondence and publishable manuscripts (although that attitude is becoming less entrenched).	Print quality is comparable to the best typewriters'. Different typefaces are obtainable by changing the print wheel.
Can accommodate any character set and graphics (although not usually *color* graphics).	Character sets are limited to available print wheels. No graphics capability.
Prints at 50 to 150 characters per second (cps): 100 cps = about 3 double-spaced pages per minute.	Prints at 20 to 40 cps: 40 cps = 1 double-spaced page per minute.
Dot-matrix printers with a 7 × 9 matrix and speeds of 100 cps are available for about $300.	Letter-quality printers with speeds of 20 cps are available for about $900. Greater speed raises the price considerably.
Note: If you plan to run your printer more than six hours per week, make sure to get a *business-quality* printer. Lightweight, inexpensive printers may not stand up to continuous use.	

Table 5–1: Comparing Dot-Matrix and Letter-Quality Printers.

of systems costing several times more. It's also easy to use. If you read on, you'll be comfortably entering and editing text in less than half an hour. Formatting the text is a bit more demanding, partly because the training manual provides only the most basic examples for practice. The section of this chapter entitled Writing in the Real World provides several examples of page formats and editing tricks that you can apply to a variety of situations.

EASY SCRIPT is a copy-protected program, so don't try to make a duplicate of the EASY SCRIPT diskette. If the diskette gets damaged, contact Commodore at the address on the back of the package containing the product, and you will be sent a replacement for a nominal charge.

To load EASY SCRIPT, make sure your computer is turned *off*, and then follow these steps. (If you're using the Datassette recorder, adapt the procedure to a tape-based system.)

1. Power up the TV or 1701 monitor, disk drive, printer, and computer — turning on the computer *last*.
2. Insert the EASY SCRIPT diskette in the drive.
3. *Type* LOAD "0:*",8,1 <RETURN>

The screen changes color as the program loads. Loading takes up to forty-five seconds, which may feel like a long time when you're eager to start writing. The first screen display (Figure 5–2) asks you how you want the text displayed and what kind of memory device and printer you're using:

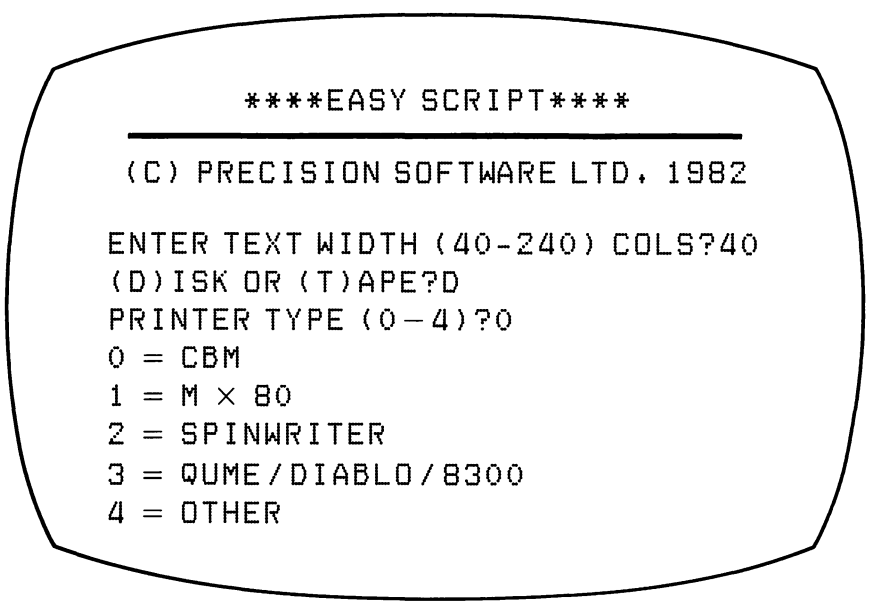


Figure 5–2: Initial EASY SCRIPT Screen. The preselected ("default") answers are 40 columns, (D)isk storage, and type 0 printer. Press RETURN to select these answers. Otherwise, type the response that applies to you, and then press RETURN.

You may see some flickering points of light, known as sparkle, dancing across your screen. The sparkle will disappear after these questions are answered and the program is "initialized."

"Precision Software, Ltd." identifies the company that created EASY SCRIPT, which Commodore now distributes under its own name.

In response to ENTER TEXT WIDTH, type 40 <RETURN>. This means that as you type, the text will stretch across the screen (40 characters) but no farther. TEXT WIDTH on the screen does *not* affect how much text per line you'll have when you *print* your text. The print format is determined later on.

In response to the (D)ISK OR (T)APE prompt, type the first letter of the memory device you're using and press RETURN.

For PRINTER TYPE choose the appropriate number. If your printer is in the OTHER category, you must answer one more question regarding the type of interface your printer uses. The information should be easy to obtain by consulting the printer's manual.

After initializing EASY SCRIPT, the electronic writing pad appears on the screen (Figure 5–3), and you're ready to write. The screen is in edit mode, the mode used for entering and revising text. The words EDIT MODE appear at the top of the screen in the status line. The status line also shows the position of the cursor. L stands for line, C for column.

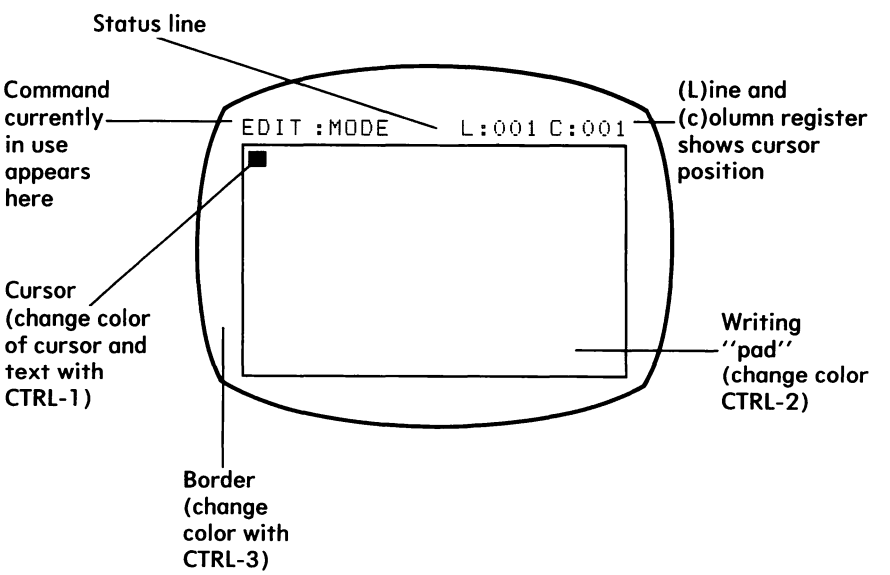


Figure 5–3: Edit Mode Screen.

CTRL — 3
CTRL — 2
CTRL — 1

You can alter the color scheme of an EASY SCRIPT screen to improve its legibility. Type the words "Does this color combination make text easy to read?" Then type CTRL-3 to vary the screen border. Each time you press CTRL-3, another color appears. Then type CTRL-2 to change the color of the writing pad. Then type CTRL-1 to alter the color of the text. Black letters on a white background or vice versa is likely to be the easiest on the eyes. (There are *two* styles of black text in the color cycle, so examine both before you make a selection.) You can change the colors at any time during an EASY SCRIPT writing session without affecting the text.

> When you're ready to write, remove the EASY SCRIPT diskette and insert a formatted (data) diskette for saving your work.

Sit Right Down and Word Process Yourself a Letter

Before typing in a practice example, erase the sentence you just typed to test the legibility of the color scheme. Use the DEL key to erase, just as you do when correcting a BASIC statement. When the cursor is in the home position, the counter on the status line reads L:001 C:001.

The first line of EASY SCRIPT text must state the name of the document — its file name. Next, you should tell EASY SCRIPT how to print the text — the margins you want, single spacing or double spacing, and so on. Before typing the file name or the formatting instructions, however, you must press F3.

F3

When you press F3, an asterisk (in reversed video) appears on the screen at the cursor position. Begin the document by choosing its file name, which can be up to sixteen characters. The rules established in naming files (Chapter 2, pages 43–44) apply to EASY SCRIPT files as well. For the practice example ahead:

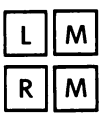
You type: F3nb"LETTER.BOB" <RETURN>

When you save the text later on, it will be listed as LETTER.BOB in the disk directory. The letters *nb* signify what the EASY SCRIPT manual calls a *comment line*. A comment line affects

neither the text nor the printed document. "nb" allows you to name the file and to write reminders to yourself that are saved along with the text. Writing comment lines is like jotting down notes to yourself in the margin of a typed page — except that the notes never show up on the printed page. Although the manual doesn't explain the letters "nb," they probably stand for *nota bene* (Latin for "note well"), which writers use to indicate an important aside or comment.

Now establish the margins for the printed page. If you're using normal letter-sized paper:

You type: F3lm10:rm70 <RETURN>



This formatting command sets the left margin (lm) at 10 and the right margin (rm) at 70, leaving about one inch of white space on either side of the page. There are many more formatting options, but those can wait until later.

Now type the text of the letter in Figure 5–4. The keyboard can be used as if it were a typewriter, bearing in mind one major difference: the RETURN key. Use RETURN only when you come to the end of a paragraph, when you want to insert a blank line, or when you don't want any more text to appear on a line. Otherwise, the text wraps around to the next line automatically. The final result should resemble Figure 5–4, except that <RETURN> is symbolized on the screen by a < in reversed video.

```
*nb"LETTER.BOB" <RETURN>
*lm10:rm70  <RETURN>
Dear Bob    <RETURN>
<RETURN>
I'm writing to you with my new Commodore
64 computer. I know you'll be surprised
to get a letter from me (since we're nex
t-door neighbors), but writing in the au
tomated age is so much fun that I couldn
't wait to tell somebody about it. I don
't know why you popped into mind. <RETURN>
<RETURN>
There's no telling how much I'll be able
```

```
to accomplish once I learn to use the ot
her software this machine runs. <RETURN>
<RETURN>
By the way, as long as I've got your att
ention, how about that $10 you owe me? <RETURN>
<RETURN>
Your neighbor, <RETURN>
```

Figure 5–4: Sample EASY SCRIPT Letter.

Learning to Revise

Before correcting typing mistakes or revising text, you should practice moving the cursor around the screen. You can move the cursor and use the HOME key just as you do when the computer boots up in BASIC.

INST DEL

The INST/DEL key also works the same way it does in BASIC. DEL erases the character immediately to the left, and SHIFT-INST inserts a space at the cursor position, allowing you to type a character.

EASY SCRIPT enters text in the overwrite mode, so you need to press SHIFT-INST to insert text. You can, however, type in the insert mode continuously, which means that when you enter text at the cursor position, the text to the right of the cursor is pushed forward. To use the insert mode:

F1

I

You type: F1 (the F1 function key)
Computer: MODE (flashes)
You type: i (for Insert)
Computer: Insert ON

(Notice that you don't have to press the RETURN key.) For practice, delete the word *how* in line 18 with the DEL key, and then insert the words *let me remind you*. To turn off the insert mode, press *F1* and *i* again. Some word processors preselect the insert mode, while others, like EASY SCRIPT, start off in the overwrite

mode. But EASY SCRIPT gives you a choice, so experiment with both modes until you decide which is more comfortable.

Using the INST / DEL key (or insert mode), correct any typing errors in the text of your letter.

Previewing, Printing, and Saving

Preview

EASY SCRIPT displays exactly forty characters (or spaces) per line, even if the line breaks in the middle of a word. Until you get used to it, reading text in this form can be disconcerting. EASY SCRIPT, however, provides a command that lets you preview the text as it will appear on the printed page.

To preview the text, select the output mode, and then select Video:

F1
O
V

You type: F1 (*the function key*)
Computer: MODE (flashes)
You type: o (for Output)
Computer: Output
You type: v (for Video)

The computer shows you how the text will look on the printed page, given your formatting instructions.

Note that the format commands and other special symbols have disappeared from the screen. The screen becomes a *window* through which you can view the final document before printing it. Since the screen can display a maximum of forty characters per line — while your page will have a line length of sixty characters, from a left margin of 10 to a right margin of 70, you can't see the whole page at once. You have to scroll the window. Examine Figure 5–5 on page 142 and practice the window movement commands shown in the margins.

Hint: If you want to view the entire page at once with the preview option, change the left and right margins to the screen margins 1 and 40. Line 2 of the edit mode screen would then read: [*]1m1:rm40[<] (Remember, [*] symbolizes F3; [<] symbolizes <RETURN>. You can now see all the text in preview mode with-

out having to scroll. Don't forget to change the margins back again before printing.

To return to the edit mode, press the STOP key. The status line reads "Aborted"; it's the preview that's been aborted. Hit any key, and the status line changes back to its normal edit mode.

The print preview command examines one page at a time. To view the next page, press C (for Continue). Page breaks are symbolized by a broken line: (----). A solid line indicates the end of the text: (——). Press C or STOP to return to edit mode.

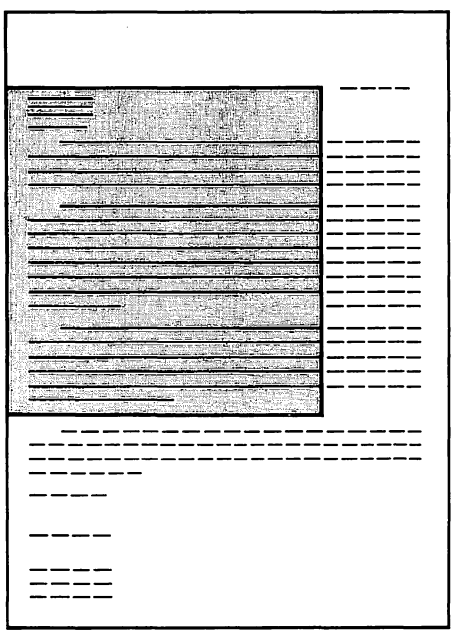


Figure 5–5: Print Preview Window. Use the information in the margins to practice moving the preview window.*

*Practicing window movement commands with a short file in memory is not feasible. Type in at least two pages of text, or hold down RETURN key until line counter reads L:070. Add dummy text along the way by holding down any key.

Printing the Final Version

After making the necessary corrections to the sample letter to Bob, print the letter. Its format, or visual design, is determined by the commands you typed in after pressing the F3 key. After printing the letter in this simplified format, you can experiment with other formats, print the results, and compare them.

Printing text is accomplished in three steps, just like previewing text on the screen. Select the output mode, but then press "p" for Print instead of "v" for Video.

You type:	F1	
	o	(for Output)
	p	(for Print)

The printer should respond immediately. If it doesn't, press STOP-RESTORE, reinitialize EASY SCRIPT by answering the four start-up questions, and try the print procedure again. (Remember, press STOP-RESTORE firmly and, if necessary, several times. STOP-RESTORE won't erase your text.) If the printer still doesn't work, you have a problem with the printer itself or the interface cable. Appendix C of the *User's Guide* may help you solve the problem. If not, call your dealer or a member of a nearby user's group.

Because EASY SCRIPT assumes you're using individual sheets of paper, it stops printing after the first page of text — unless you instruct it to print all the pages of text continuously. When the printing stops at the end of the page, insert another piece of paper and press C (for Continue printing). EASY SCRIPT can print all the pages in your file continuously, but you have to use continuous form paper, such as rolled or fan-fold paper. To print continuously:

You type:	F1	
	o	
	c	(for Continuous printing)
	p	

Saving the Text

Although the letter's been printed out, let's suppose you wish to save it onto disk for future revision. There are three variations of the procedure for saving a file:

F1
F

1. *You type:* Fl
   Computer: MODE
   You type: f (for File)
   Computer: File Name:
   You type: LETTER.BOB <RETURN>

   (In the last step, you need to type RETURN to let EASY SCRIPT know you've finished typing in the file name.) The disk drive begins to whirr and click until the file is saved. The word COMPLETE appears on the status line in normal circumstances. If an error message appears on the status line, turn to page D–3 of the EASY SCRIPT manual to identify the problem.
2. There's a convenient shortcut to the procedure above. Begin by bringing the cursor to the home position. Then press F1 and F, as before. Instead of typing in the file name when the status line requests it, simply type F2 <RETURN>. (Remember that F2 is made by typing SHIFT-F1.) The file name on line 1 instantly appears on the status line. Press RETURN and the file will be saved. Don't practice this shortcut yet. Read variation 3 below, and then use the shortcut method for saving LETTER.BOB. The manual tells you to press *0*: (the drive number) before pressing F2. If you do, then you must press F2 *twice* to get the file name in the status line. Identify the drive number only if you're using two drives.

F2=SHIFT–F1

3. If you've already saved your LETTER.BOB file once, and then decide to save a new version of it, REPLACE FILE? appears on the screen. Like the save-and-replace command in DOS (Chapter 2, pages 60–61), saving a new version of a file erases the earlier version. For practice, make a slight revision to the letter on the screen, and then save it (using the shortcut method — pressing F2 instead of typing in the file name). Type Y to go ahead with the save-and-replace. (To abandon the save-and-replace, type N.)

Displaying the Directory and the Housekeeping Mode

F4=SHIFT–F3

Your formatted diskette has a directory of files that should now include the LETTER.BOB file you saved. To see the file name in

the directory, select EASY SCRIPT's housekeeping mode with the F4 key (made by pressing SHIFT-F3). The *housekeeping* mode enables you to view, rename, and erase the files on your diskette and to determine how many free blocks remain for new files. To view the directory:

You type: F4 (press SHIFT-F3)
Computer: `Disk mode` (text disappears temporarily)
You type: $ <RETURN>
Computer: (displays your directory)
You type: <RETURN> (to continue) *or* STOP (to exit)

After using EASY SCRIPT for a few months, your directory of files may exceed the twenty-four lines on the screen. Press RETURN to see more files displayed on the next screen. When you've examined the directory, press the STOP key. EASY SCRIPT returns you to the edit mode immediately.

Notice that consulting the directory doesn't erase the text in memory. You'll appreciate that fact when you use a sophisticated EASY SCRIPT command that writes text from a file on diskette into a file on the screen. Should you forget the exact name of the stored file, you can simply display the directory and then return to your text.

The EASY SCRIPT *User Guide* explains on pages 2–22, 23, and 24 how to delete (or erase), rename, and copy files. Since these disk operations are fairly simple — and are unnecessary for the practice session in this chapter — they won't be explained here.

Dynamic Deletions

After entering text, deleting text is the writer's most typical activity. There's very little writing that can't be improved by revising it. Ernest Hemingway, whose craft as a novelist is legendary, once claimed that he threw out one of every three pages. He would have loved word processing, because you can "throw out" pages all day and never waste a sheet of paper.

Now that you've stored the sample letter on disk (or tape), you can practice some powerful erase commands without fear of

losing your work. Even if every character is erased, the letter can be loaded back into the computer in seconds.

The DEL key erases one character at a time, which takes too long when you're eliminating a paragraph or more. Pressing F1 makes some dynamic erasing commands available.

Try each of the techniques below. Don't be afraid to chop up your text; LETTER.BOB is safely stored and you'll be loading it again shortly.

To erase to the end of a sentence, position the cursor where you want the deletion to begin. Then:

F1
E

You type:	F1	
	e	(for Erase)
	s	(for Sentence)

The text from the cursor to the first period disappears. (EASY SCRIPT doesn't interpret a question mark or exclamation point as a sentence terminator, so you can only erase a sentence with one command when the sentence ends with a period.)

To erase to the end of a paragraph, position the cursor where you want the deletion to begin. Then:

You type:	F1	
	e	
	p	(for Paragraph)

The text from the cursor to the first RETURN symbol disappears.

To erase the text to the end of the file, position the cursor where you want the deletion to begin. Then:

You type:	F1	
	e	
	r	(for Remainder of file)

To erase all the text and begin again:

You type:	F1	
	e	
	a	(for All)

All the text disappears, and the word ERASE remains on the status line. To return to the edit mode, either begin typing or press F1 and then begin typing.

> ***Caution:*** Once you've erased text, there's no way to get it back.

Loading Your File

If you've used the erase "all" command above, you're looking at a blank writing pad. Although no text remains in memory, you can retrieve the letter by LOADing the LETTER.BOB file.

You type: F1
Computer: MODE (flashes)
You type: 1 (for Load)
Computer: Load:
You type: LETTER.BOB <RETURN>

In the last step, you press the RETURN key to let EASY SCRIPT know when the file name is complete. (*Note:* When typing LETTER.BOB, use F5 to lock the capital letters on and F5 again to turn them off. Using the SHIFT LOCK key disables the RETURN key.) The loading procedure above is also used after booting up EASY SCRIPT when you want to work further on a stored file.

Panning

F1
SPACE BAR
STOP
F1
CRSR

When the document you're working on is several pages long, you may want to pan rapidly up, down, or across the text. *Panning* means moving the text continuously without holding down a key. Use the panning mode by pressing the F1 function key. The word mode on the status line begins to flash, awaiting your next command. Now use one of the cursor keys to pan in the desired direction.

To halt panning temporarily, press the SPACE BAR. Press the SPACE BAR again to continue.

To stop panning, press STOP. To pan in another direction, or to return to the previous screen position, you must *re*select F1 and press the CRSR key for the desired direction. You can practice panning with the LETTER.BOB file in memory, but since the text

is so short, you'll pan past the text almost instantly. Notice that EASY SCRIPT continues panning, even through the blank lines of the file — until you press the STOP key.

Playing with Blocks

EASY SCRIPT facilitates typing and revising text by letting you move and duplicate *blocks*, or segments of text. A block could be ten words, ten lines, or ten paragraphs. Once defined, the block can be moved to or duplicated in another location without retyping.

Suppose you decide that the last paragraph of your letter to Bob ("By the way," etc.) should be the first paragraph. Set the *range* of the block by positioning the cursor where you want the defined block to begin — over the B in line 18. Then:

You type: F1
r (for range)
Computer: `Set range`

Now move the cursor to the end of line 19 (line 19, column 40). The text you've ranged the cursor over appears in reversed video.

You type: <RETURN>
Computer: `EDIT MODE` (reversed video disappears)

The ranged text is stored in a *buffer* memory, an area of memory set aside temporarily for storing your block. You may return to the normal edit mode and transfer the block to its new location when you're ready. It's preferable, though, to move the block right away, before you forget what's in it. To move the block to a new location, move the cursor to that location. (Move the cursor up to line 4, column 1 of the LETTER.BOB screen.)

You type: Fl
x (for exchange position)

The last paragraph becomes the first paragraph.

A block of text can also be duplicated elsewhere in the document — moving it to a new location without removing it from the old. You might use this option in writing a contract or some other formal document in which clauses or conditions are often repeated.

To duplicate a block, set the range as you did in the previous exercise, move the cursor to the new location, and then:

F1
A

You type: F1
a (for again)

Practice this on your own with the LETTER.BOB document. The ranged text appears in both the new and old locations.

Searching (and Replacing)

EASY SCRIPT can search for a specific word or phrase in your text and, if you wish, replace it with another word or phrase. There are several uses for this option, from correcting spelling mistakes to locating a word or phrase in a long document. Some useful ways to implement this command will be presented later in the section called Writing in the Real World.

A Five-Minute Introduction to Formatting

The real power of word processing resides in its ability to format and reformat automatically. This eliminates the tedious, tiresome task of retyping your manuscript after each improvement. Control over the visual organization and overall appearance of your work is literally at your fingertips.

But defining page formats is not always simple. In later examples, you'll learn how (and why) to change formats in the middle of a page, and how to do some fairly complex formatting. At this point, we'll work with the LETTER.BOB file, demonstrating how simple formatting commands can alter the look of the final work.

You should already have printed the letter to Bob using the format appearing on line 2 of the screen. Let's change three aspects of the letter's appearance: its position on the page, the line spacing, and the right margin.

F3 V P

1. **vp**—A personal letter looks better if it's centered vertically on the page. The vertical offset command tells the printer how many lines to skip before printing the text. (You can also skip lines by pressing RETURN in

column 1 of the line, but that method uses more memory.) Move the cursor to line 3, column 1 and press F1 followed by SHIFT-INST. This inserts a blank line for a new formatting command. Then:

You type: F3vp8 <RETURN>

This command tells the printer to skip eight lines before beginning the letter. (The F3 key is displayed as an asterisk.)

F3 S P

2. **sp**—The original letter was single-spaced because no line spacing command was given. EASY SCRIPT assumes you want single spacing unless you specify otherwise. (See Appendix E of the EASY SCRIPT *User Guide* for a list of the other default — preset — formatting values.) But some people prefer double-spaced text (inserting one blank line between lines of text); and if you're planning to submit your work for editing, you might want triple-spaced text (two blank lines inserted).

   To change the printed version of LETTER.BOB to double spacing, insert another blank line at line 3 (F1 SHIFT-INST). Then:

   You type: F3sp1:

   This instructs EASY SCRIPT to double-space the printing. The line ends with a colon because we're going to add another command to the line. But first, preview the printed text with the video output option (F1-o-v) to see the effect of sp1.

J U

3. **ju**—The original letter was printed with a *ragged* (or uneven) right margin, the kind of margin that a typewriter produces. EASY SCRIPT enables you to *justify* the right margin — to space the words so that each line (except those ending with RETURN) is brought out to the last column of the right margin. Most books, including this one, have a justified right margin.

   To justify the margin, finish the line in step 2 above so that it reads: F3sp1:ju1 <RETURN>. Now preview the justified text, and then print the letter in its new form.

Summary of EASY SCRIPT Commands

Cursor Movement

CRSR keys operate as they do in BASIC.

HOME	cursor returns to line 1, column 1 of screen.
CTRL-W	moves cursor to first letter of next word.
CTRL ←	moves cursor to last letter of previous word.
SHIFT-RETURN	moves cursor to start of next line (but does not insert a RETURN symbol).
←	moves cursor to last letter of previous line.
F7	moves cursor to next (horizontal) TAB.
F8	moves cursor to next (vertical) TAB.

GOTO Commands

F1 g nn (line number) <RETURN>	cursor goes to specific line number.
F1 g e <RETURN>	cursor goes to end of text.
F1 g 999 <RETURN>	cursor goes to end of file.

Formatting Commands

Commands should be preceded by F3 and terminated with <RETURN>. String together commands with colons: for example, *lm10:rm70:sp1 <RETURN>. Formatting commands can be issued at any point in the text, but they normally begin with the F3 (*) in column 1.

lm	set left margin.
rm	set right margin.
hl	set left margin for header and footer.
hr	set right margin for header and footer.
pl	page length (66 is preset for letter size; 72 for legal size).
tl	number of lines to print (including blank lines for double spacing; preset for 60 on letter-size paper).
sp	spaces between lines (preset for 0 — single spacing; use 1 for double spacing; 2 for triple spacing).
ju	justification of right margin (preset for "ragged"; use 1 for justified right).
cnl;. . .cn0 <RETURN>	center . . . and then turn off centering command (note that a semicolon separates *cnl* from the text).
ra	right align text (leaves left margin ragged).
fp0	forced page break (for example, at the end of a chapter).
In (followed by a number)	number of blank lines inserted before printing begins (in other words, indicates extra space at the top of a page).

Enhanced Printing

These commands are printer-dependent. Experiment with your printer to determine their effect. The effects can*not* be determined by using the Output Video command.

F1[...F1]	underlined or enlarged
F1(...F1)	boldface or reversed image
F1;...F1:	alternative boldface
F1'	superscript (affects one character)
F1,	subscript (affects one character)
F1&%	shadowed text
F1!"	red ink (if available on printer) or condensed type (16 characters per inch)

Deleting

DEL erases character to the left.

F1
e
s erases from cursor to end of sentence (first period).

F1
e
p erases from cursor to end of paragraph.

F1
e
r erases remainder of file (from cursor position).

F1
e
a erases all text in the file.

Inserting Text

SHIFT-INST inserts a blank space.

F1
SHIFT-INST inserts a blank line (line should not be left blank; use RETURN or "In" command to insert blank lines in the text).

F1
i type in Insert Mode.

Previewing and Printing Text

F1
o
v displays formatted text on screen (press C to see next page).

F1
o
p begins printing (press SHIFT-C after first pause to print continuously; to print *multiple copies*, press x before p and specify number of copies).

Capital Letters

F5	turns CAPS on (other keys behave normally).
F5	turns CAPS off (other keys behave normally).
SHIFT LOCK	turns all keys to upper-case mode (disables DEL and RETURN key).
Flu . . . Flu	changes intervening text to upper or lower case.

Hyphens and Spacing

-	hard hyphen
F1-	soft hyphen (breaks word at soft hyphen only if formatting requires it).
SHIFT-SPACE BAR	"linked" space (ensures that words linked by a SHIFTed space will remain on the same line during formatting).

Search (and Replace)

F1
s
xx <RETURN>
<RETURN> (as response to REPLACE?)
F1
h
m search memory for string xx (without replacing);
or
l search linked files for string xx (without replacing).

F1
s
xx <RETURN>
yy
F1
h
m search memory for string xx and replace with yy;
or
l search linked files for string xx and replace with yy.

Setting TAB

Position cursor at desired TAB location, then:

F1
t
h for horizontal TAB setting; *or*
v for vertical TAB setting.

Continue typing to return to edit mode.

CLEAR TABS: First position cursor at TAB, then:

F1
c
h for horizontal TAB; *or*
v for vertical TAB; *or*
z for "zero" — to CLEAR ALL TABS.

To SAVE a file with TABS, add + to file name: LETTER.BOB+.

Scrolling Text in Edit Mode

F1
SPACE BAR displays next 23 lines of text.
F1
SHIFT-SPACE BAR displays previous 23 lines of text.

Scrolling Text in Video Output Mode

Commodore (C=) Key moves window down.
CRSR (horizontal) moves window to right.
SHIFT-CRSR (horizontal) moves window left.

F5	moves window 20 columns right.
F7	moves window 40 columns right.
RETURN	returns to column 1.
SPACE BAR	stop scrolling.
SPACE BAR (again)	continue scrolling.
C	see next page (continue past broken line).

Panning in Edit Mode

F1 and CRSR key moves text in direction selected by normal operation of CRSR keys.

SHIFT	speeds up panning.
SPACE BAR	stops panning temporarily.
SPACE BAR (again)	continues panning.
STOP	stops panning. (Reselect F1 to pan in a new direction; begin typing to return to edit mode.)

Display Directory

F4	
$ <RETURN>	(use drive number after $, if more than one drive).
CTRL	slows down automatic scrolling of directory.
STOP	return to edit mode and text in memory.

Writing in the Real World

The preceding practice session shows that EASY SCRIPT is a good antidote for writer's cramp. You can delete words or paragraphs, move blocks of text, and cut and paste with abandon, knowing that your word processor will reformat the results automatically.

But for most of us, real writing involves more than grinding out text, polishing it, and formatting it into paragraphs. That's because, in a general sense, *writing* is as mythical as "the average

family," the "typical consumer," and "normal circumstances." In the real world, writing means specific types of documents: research and term papers, business correspondence, magazine articles, film scripts, novels, and love letters. Each type presents its own challenges and requirements.

In this section we'll practice using EASY SCRIPT to improve a variety of specific writing tasks. Even if the examples ahead aren't tailored precisely to your needs, you can adapt most of the techniques to whatever you're planning to write. Some techniques are simple tricks that make your document more eye-catching; others are detailed procedures that require some study and practice.

Reproduce the examples ahead on your screen and print them to make sure you've mastered the technique. Then apply them — or variations on them — to your real-world writing projects. Have your EASY SCRIPT *User Guide* nearby for reference.

> Remember that the reversed video asterisk (*) that prefaces format commands is made by pressing the F3 key. Remember, too, that if you plan to save these exercises on disk or tape, you must give the saved file a name by typing *nb"FILENAME.XXX" <RETURN> on line 1 of the file.

Getting Centered

F3 C N

In term papers, reports, or chapter titles, centered text can look impressive, especially if you use large type. EASY SCRIPT can both center your text and, if you have a dot matrix printer, produce oversized characters. (With letter-quality printers, the typeface is stamped on the print wheel, so you can't change type styles.) Suppose you're submitting a report to your class or a Commodore users' group on the virtues of word processing. You might begin as shown in Figure 5–6 below.

```
*nb"Report.wp"<
*lm10:rm10:spl<
*vp5<
*cn1;■Word Processing■*cn0<
```

```
*cnl;How to Spend Your Time Writing -- I
nstead of Typing<
*ln3<

     A famous writer was once asked to r
eview the work of an aspiring but less t
han promising novelist. His assessment w
as, "That's not writing — that's typing
." If you write frequently, typing easil
y. . .
```

Figure 5–6: Formatting a Centered Title.

The text has been formatted with a left margin of 10, a right margin of 70, and double spacing. Line 3 tells the printer to start printing five lines from the top of the page. In lines 4 and 5, notice the semicolon between the cnl, which turns on the centering command, and the text. Use a semicolon whenever text follows a formatting command. The small squares before the "W" and after the "g" in line 4 are made by pressing F1-[and F1-] respectively. If this command doesn't cause your printer to produce large characters, try some of the alternatives on pages 8–9 of the *User Guide*.

Line 5, which contains the subtitle, is also centered, but without elongated characters. Line 6 tells the printer to skip three lines before printing again. Using this command requires less memory than hitting RETURN three times, which means that you'll have more room to write in the file.

The printed result should look like this:

```
                Word Processing
How to Spend Your Time Writing -- Instead of Typing

  A famous writer was once asked to review the work
of an aspiring but less than promising . . .
```

If you'd like to save this screen for future reference, do so now. Otherwise, erase the screen by pressing F1, e, and a.

Do-It-Yourself Letterheads

Centering text can also provide you with a letterhead that you can save on diskette and load into memory before you start writing a letter. Create a sample letterhead by typing the lines in Figure 5–7.

```
*nb"Letterhead"<
*vb5<
*cnl;BlowRite Balloon Co.,<
2543 Rubber Tree Avenue<
New York, New York 10001<
<
"We're proud to be full of hot air!"*cn0<
*ln3<
```

Figure 5–7: Creating a Letterhead.

Notice that the centering command is turned on only once for lines 3, 4, 5, and 7. Each line is centered until the centering command is turned off at the end of line 7. Line 6 inserts a blank line between the address and the company's motto. The letterhead and text are single-spaced if no other spacing is specified.

> *Note:* An incorrect format produces an error message in the status line when you attempt to use the video output command. If you get an error message, consult page D-2 in the *User Guide* and check your typing for inaccuracies.

The letterhead comes out like this:

```
BlowRite Balloon Company
2543 Rubber Tree Avenue
New York, New York 10001
"We're proud to be full of hot air!"
```

Type in your own letterhead and preview it by selecting video output (F1-o-v). Then save the file. The next time you write

a letter, begin by loading Letterhead. Write your letter and print it. If you want to save your correspondence, change line 1 to a new file name — for example, LETTER.IRS — and save the letter. The Letterhead file remains on disk or tape for the next time.

Getting Ahead with Page Numbers

F3 H D

Another way to improve long documents is by putting running heads and page numbers on each page. There are running heads at the top of the pages in this book, so you can thumb through it and locate discussions of specific topics. In word processing jargon, running heads are called *headers*. (EASY SCRIPT can also produce *footers* — "running heads" that appear at the bottom of the page — if you prefer that style.)

Figure 5–8 is a sample running head for a hypothetical term paper on the subject of surrealism. The header has three parts to it. The report title begins at the left margin, the page number (#) is centered, and the subject title ends flush with the right margin. (Naturally, you must limit the length of the header to words that fit within the three segments of the page.)

```
*nb"Surrealist Art"<
*lm10:rm70:spl<
*hd3:Surrealist Art,#,Beginnings<
*vp6<
*cnl;Surrealist Art *cn0<
*ln3<
   Surrealist art juxtaposes realism a
nd the totally imaginary. The painters i
n this tradition love to surprise and ev
en to shock the viewer. . . .<
```

Figure 5–8: Formatting a Header.

Preview the format before printing. The final version looks like this:

```
Surrealist Art     1          Beginnings

                Surrealist Art

     Surrealist art juxtaposes realism
and the totally imaginary. The painters
in this tradition love to surprise and
even to shock the viewer. . . .
```

In line 3, the 3 tells the printer to print the heading three lines above the first line of text, which in this example is the title. The commas in columns 20 and 22 divide the printed page in thirds. Every header command requires two commas to divide the page in thirds, but it's not necessary to print in each third. To have a simple heading and page number on the right-hand side of the page, the line would look like this:

```
*hd3:,,Beginnings — #<
```

The result:

```
                                  Beginnings — 1
```

Formatting Footnotes

The easiest way to cite your sources formally is by listing them in footnotes at the end of the document you've written, or at the end of each chapter. But you can also place them at the bottom of the page, if you leave room. Type a solid line with the hyphen, or minus sign, key from the left margin across ten columns. Begin your footnote immediately below the line.

No matter where you list the footnotes, the formatting is the same. Footnoting requires you to use the superscript and underlining commands. As a sample, let's footnote the following quotation from the paperback edition of an entertaining and provocative book about the impact of microcomputers on society:

```
By the late 1980s, if data compression
techniques continue on their present
```

```
curve, it will be possible to store
very large books, perhaps even sets of
books, on a microchip and a whole library
in a space about the size of one of
today's paperbacks.[1]
```

To cite the source, type Figure 5–9.

```
[U]1Christopher Evans, [L]The Micro
Millennium[J] (New York: Washington
Square Press, 1981), p. 115.<
```

Figure 5–9: Formatting a Footnote.

The reversed U is made by pressing F1 and the apostrophe. (Your printer may require a different key following F1. See the *User Guide*, page 8–9.) The L-shaped block before and after the book title is made by pressing F1-and-(and then F1-and-) respectively. (Again, your printer may differ.)
The result:

[1]Christopher Evans, The Micro Millennium, (New York: Washington Square Press, 1981), p. 115.

Justifying Your Rights

As stated earlier, right justification spaces your text so that it's flush with the right margin, just as it is with the left. Most books are typeset with a justified right margin. The alternative is called ragged right, which is the way your typewriter prints text. EASY SCRIPT justifies text by inserting *soft* spaces between words to bring every line (except lines ending with RETURN) out to the right margin.

EASY SCRIPT is preset for a ragged right margin (sp0), but you can change the setting *globally* (for the entire document) or anywhere in the text. You might want to right-justify a block of text that you're citing within a larger report. Figure 5–10 illustrates a sample of this. (*Note:* In this example, only the format

command appears exactly as it does on the screen. The text below appears already formatted.)

```
        The amount of learning that occurs
during the first few months of life is
difficult for the adult to imagine, as
Selma H. Fraiberg emphasizes in her
classic work, The Magic Years:
*sp0:ju1<
     A project of such magnitude in
academic research would require
extensive laboratory equipment and
personnel; to be fair about it, it has
taken just that to reconstruct the
experiments of the infant.
*sp1:ju0<
```

Figure 5–10: Right Justification.

After the quoted material, the format command returns to double spacing and a ragged right margin.

Whether or not to justify your right margins is partly a matter of taste. Except for typeset work and special uses, such as the long quotation above, most documents will look more natural with a ragged right margin.

Search — and You Shall Find

F1 @ S M

When working with long documents, use the search and replace option when you discover a spelling error or when you want to find a specific part of the text. Suppose you've written a ten-page document on European architecture and suddenly notice — to your chagrin — that your *rococo* has one too many *c's*. Bring the cursor to the top of the file with F1, g, 1, and <RETURN>. Then,

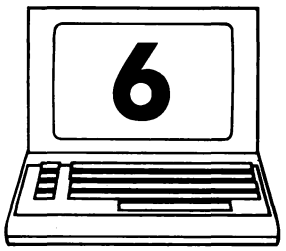
You type: F1
s
Computer: SEARCH:

You type: roccoco <RETURN>
Computer: REPLACE:
You type: rococo <RETURN>
Computer: EDIT MODE
You type: F1
@
m

The spelling corrections are made instantly throughout the text and the cursor remains at the bottom of the file. (The @ means that you want to both search and replace. The m means the "text in memory." You'd use l to search linked files.)

You can also use this function to find a section of the text to improve upon. Suppose you want to add some information to a ten-page composition about the Gaudi buildings in Barcelona. Don't scroll through the text on screen looking for that section. Search for the word *Gaudi* or the word *Barcelona*. You'll be there in no time. In this case, press <RETURN> in response to the REPLACE: prompt. Then follow that with F1, h (for hunt), and m.

Mastering a few of the techniques above should start you thinking about specialized formats and EASY SCRIPT commands that you can adapt to your actual writing needs. If you discover a useful procedure, be sure to document it on paper or on disk. Keep a list of these tools with your EASY SCRIPT *User Guide*, and you'll continue to get the most out of your word processor.



Managing Your Money with Easy CALC RESULT*

Except for filing complex tax returns, drawing up a will, or other such singular necessities, most of us are stuck with handling our own financial bookkeeping — trying to make ends meet, and showing a bit of profit along the way. The paperwork required can be tedious, especially if you want to consider the consequences of possible future income and expenses.

The software discussed in this chapter automates your financial record-keeping. Easy CALC RESULT lets you display your financial situation in a layout that makes it easily understood; it performs all the calculations for you, including taking subtotals of any specified items. It can automatically calculate percents, averages, net present value, and other useful functions. Equally important, the software enables you to make financial projections based upon hypothetical financial scenarios. You can, for example, see the long-term results of an investment at a variety of possible interest rates; or determine your monthly expenses given a rise in certain costs and a decrease in others. Using the computer to make projections based on hypothetical values is sometimes called *modeling*.

Of course, the C-64 isn't a crystal ball. You'll still need an analyst — a clairvoyant one! — to tell you where the stock market's

*Easy CALC RESULT may also be sold as EASYCALC 64, a Commodore software product. Under either title, the software should be available where Commodore computers are sold.

going, and you may need a CPA to help you decipher the IRS's most recent regulations. But now that you're using the C-64, you don't have to spend hours a month with paper, pencil, and a scratch pad to keep track of how your money's being spent and whether it's being invested wisely.

To see how Easy CALC RESULT automates your financial record-keeping and planning, let's construct a typical family budget. You'll see how Easy CALC RESULT assists you in organizing your finances and in making crucial financial decisions.

The Family Budget

The most fundamental bookkeeping procedure keeps track of income and expenses. Accomplishing this is fairly simple, though time-consuming. Make a list of your basic monthly costs, leaving room for unanticipated expenses; then make a list of your various sources of income. Total the income and expenses separately, and subtract the total expense from the total income. That tells you whether you're keeping your head above water.

Easy CALC RESULT helps you by performing the necessary calculations automatically and by saving the data on tape or diskette. Next month, you'll load the computerized budget, fill in the current month's figures, and let Easy CALC RESULT compute the totals.

Sometimes the scenario becomes more complicated, and then the advantages of an automated bookkeeping system emerge more dramatically. Suppose you're contemplating buying a home instead of renting one, or perhaps moving from one location to another, where your expenses will be different. Easy CALC RESULT allows you to substitute hypothetical numbers for your actual expenses and, instantaneously, shows you what the result will be *if* the hypothetical case becomes true.

The software not only calculates your present financial situation, it can also project the state of your budget in the future, given any set of circumstances you'd care to imagine. In short, it can answer the question, *What if* my income and expenses change in specific ways?

The Spreadsheet Model

Easy CALC RESULT uses an electronic *spreadsheet* (or *worksheet*) to depict your financial situation and to answer your what-if questions. Grasping the spreadsheet model is the key to understanding this and similar types of financial management software.

The spreadsheet is a grid, or matrix, formed by the intersection of vertical columns and horizontal rows. The columns are labeled by letters (A–Z,AA–AZ,BA–BK) and the rows by numbers (1–254). The intersection of a column and a row is called a *cell*. Each cell can contain a *value* (a number, expression, or formula) or a *label* (a word or number to be printed *literally*). With 63 columns and 254 rows, the Easy CALC RESULT spreadsheet makes available a total of 16,002 cells. The C-64's memory, however, can accommodate information in only 1,023 cells, but that is still more than is normally needed for home and small-business use. Figure 6–1 presents a facsimile of a full-sized spreadsheet.

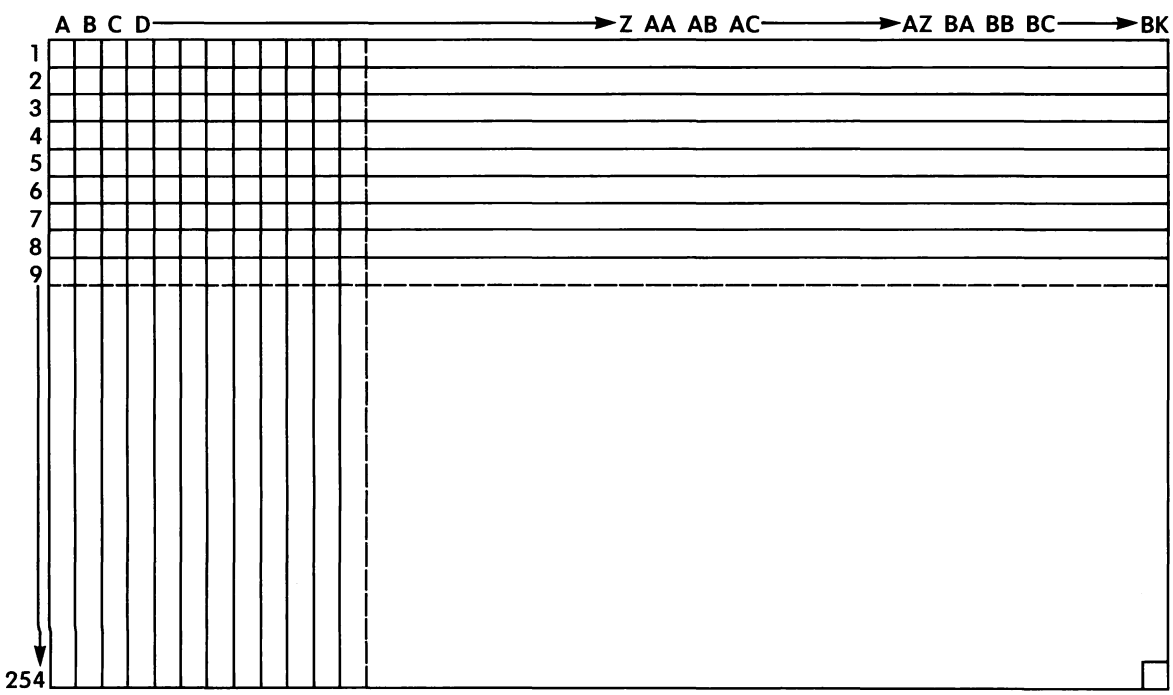


Figure 6–1: The Full-Sized Easy CALC RESULT Spreadsheet.

Loading Easy CALC RESULT

The Easy CALC RESULT program is stored in a cartridge that must be inserted in the C-64's cartridge slot *before* turning on the computer. (Never insert or remove a cartridge while the computer is turned on.) Follow this start-up procedure:

1. Make sure the computer is *off*, and insert the cartridge carefully into the cartridge slot.
2. Turn on the TV or monitor.
3. Turn on the computer (with the Datassette recorder connected, if you're using the Datassette).
4. Turn on the disk drive (if you're not using the Datassette recorder). Make sure there is no diskette in the drive when you turn it on.
5. Turn on the printer.

The start-up procedure for Easy CALC RESULT violates the normal Commodore procedure, which requires the computer to be turned on last. With Easy CALC RESULT the drive doesn't work properly if it's turned on before the computer.

Figure 6–2 shows you the initial screen display. It also illustrates how the screen functions as a window revealing a limited section of the available spreadsheet. (The window concept is also discussed in Chapter 5 in the context of EASY SCRIPT.)

The Easy CALC RESULT Story

Financial management software for microcomputers began with a program called VisiCalc. In 1977, an M.B.A. student at the Harvard Business School and a friend studying at M.I.T. formed a company called Software Arts in Cambridge, Massachusetts. Software Arts was formed to create a program designed to solve the countless what-if projections required by business managers and accountants. The result was VisiCalc (for *visi*ble *calc*ulator). Another company, now known as VisiCorp, was formed to sell the program to the fledgling home computer market. VisiCalc set the standard for financial management software. Small businesses that did not have the capital to purchase large computers suddenly

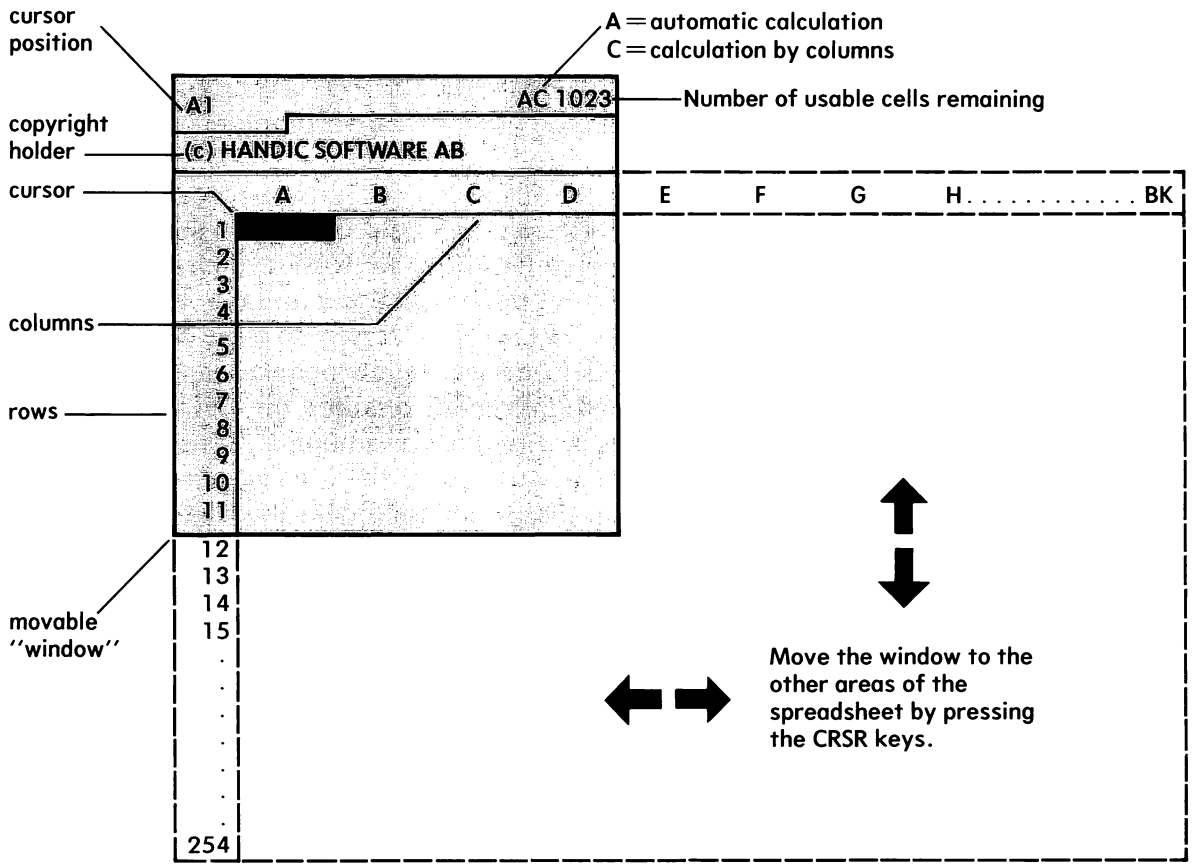


Figure 6–2: Initial Easy CALC RESULT Screen.

had the capacity to draw what-if scenarios on their screens, calculating the financial impact of various price structures, loans, employment policies, and other hypothetical investments and expenses. With this type of software, rather than hire an accountant who might need hours to compute the results of each scenario, the business manager can feed the dollar-and-cents amounts and the applicable formulas into a *model* that calculates and *re*calculates the results of each possibility in seconds. The model allows you to project the results of hypothetical situations.

Easy CALC RESULT also relies on the spreadsheet model, and it works very much like VisiCalc. But don't look upon Easy

CALC RESULT as ersatz VisiCalc, a poor imitation of the real thing. Your version of Easy CALC RESULT is more powerful in many ways than the original VisiCalc, and, besides, virtually every financial management program for microcomputers derives from VisiCalc.

Easy CALC RESULT was created by a Swedish software company, Handic Software Ab. The program is a simplified version of the same company's CALC RESULT, which includes features usually needed by professional accountants and business managers. Easy CALC RESULT is simpler to learn and designed for keeping track of personal and family finances, and for making financial analyses and projections required by small businesses.

The program is very popular among C-64 users, and it is widely available where software for the C-64 is sold and through mail-order houses. In 1983, Commodore Business Machines, Inc., had reportedly negotiated for the rights to distribute Easy CALC RESULT as a Commodore product called EASYCALC 64, but as of this writing, the software has not been released under Commodore's name. If you do come across Commodore software called EASYCALC 64, ask for a demonstration of the product and compare the commands to those described ahead. You will probably find that the software is the same as that discussed in this chapter.

Moving Around the Spreadsheet

When you load Easy CALC RESULT, the upper left corner of the spreadsheet is visible on your screen, as shown in Figure 6–2.

The cursor, represented by a solid black bar, is initially located at cell A1. The cursor indicates where the labels, values, or formulas that you type are going to be placed. Before entering any information on the spreadsheet, practice moving the cursor to different locations. Getting familiar with cursor movement gives you confidence that you know where you are on the total sheet, which is quite vast compared to the 4 columns and 21 rows visible on the screen.

First, move the cursor right by pressing the ←CRSR→ key. The cursor advances one cell each time you press the key. If you hold down the key, the effect is like scrolling the sheet past a window. Scroll until you reach the last column (BK). To return

to the A1 position, you can press SHIFT-←CRSR→. To move to cell A1 more quickly, press HOME twice. The first HOME places the cursor in the upper left corner of the screen. The second HOME takes the cursor back to A1.

Now, from A1, move the cursor down to row 145 by pressing ↓CRSR↑. You can move the cursor back to A1 by pressing SHIFT-↓CRSR↑ or by pressing HOME twice.

Labeling Your Family Budget

SPACE BAR

Let's construct a spreadsheet to keep track of a hypothetical family budget. This budget (depicted below in Figure 6–3) is a reasonably simple one designed to serve as a learning experience.

	A	B	C	D	E	F	G	H	.	.	.
1	BUDGET	JAN	FEB	MAR							
2											
3	INCOME:										
4	SALARY	2607.31	2607.31	2607.31							
5											
6	INTEREST										
7	STOCKS	304.58	267.87	356.89							
8	SAVINGS	718.98	769.73	356.89							
9	CARDS	-24.5	34	0.98							
10	MISC.	0	0								
11											
12	TOTAL:	3606.37	3678.91	3322.07							
13	-----										
14	EXPENSES										
15	MORTGAGE	867.73	867.73	867.73							
16											
17	UTIL:										
18	GAS	64.09	57.87	49.45							
19	ELEC	78.61	59.08	34.07							
20	OIL	123.65	0	0							

```
21WATER              0    23.74         0
22PHONE          46.62    37.89     56.87
23
24MEDICAL           65       35         0
25AUTO
26 GAS/OIL       39.76    45.87        67
27 REPAIRS           0   275.95     25.32
28 INSUR         34.74    34.74     34.74
29MISC.              50       95        34
30
31TOTAL:       1370.2  1532.87   1169.18
32-----
33NET PROF   2236.17  2146.04   2152.89
34
35
36G
37
```

Figure 6–3: Sample Household Budget.

You can, however, substitute labels and values that apply to your individual circumstances. Although it may take a bit longer to finish the practice example, you'll come out with a perfectly usable spreadsheet.

With the cursor at location A1, press the SPACE BAR. Notice that the word *Label* appears in the upper left of the screen (on the second status line). The SPACE BAR tells Easy CALC RESULT that you're about to enter a label, or title, for a column or row. Since A1 is the head of both a column and a row, it's usually reserved for giving a title to the spreadsheet itself. Let's title this practice spreadsheet BUDGET:

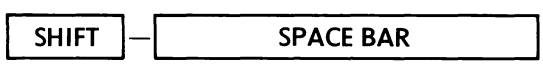
You type: BUDGET <RETURN>

The word BUDGET appears in white letters inside the cursor. If you make a typing error, use the DELete key to erase the letters. If you begin entering letters without first hitting the SPACE BAR, the screen will flash a red error message when you press RETURN.

The error message occurs because if you don't hit the SPACE BAR, Easy CALC RESULT expects you to supply a *value*. Since the word BUDGET is a *label*, not a value, the red error screen appears.

In the *value* mode, each letter you type appears in upper case. In the *label* mode, letters can be typed in either upper or lower case. Labels can consist of either letters or numbers. Values, on the other hand, must be either numbers or recognized formulas, composed of variables, arithmetic operators, and key words.

You can convert a label to a value and vice versa by moving the cursor to the cell in question and pressing SHIFT-SPACE BAR. To enter a label in upper-case letters, you can press the SHIFT LOCK key. But then you must release SHIFT LOCK before pressing RETURN or DEL, because SHIFT LOCK disables RETURN and DEL. An easier way to enter labels in upper-case letters is to omit pressing the SPACE BAR, type in the label, which appears in capital letters, then press SHIFT-SPACE BAR <RETURN> to convert the value to a label.



Some printers interpret SHIFTed letter keys as *graphics characters*, instead of printing them as capital letters. If this occurs, type your labels in *lowercase*. They will be printed as capital letters. This means you cannot use upper lower case — Typewriter Mode — with this program and your printer, because the printer reads characters in the upper-case/graphics mode only.

Move the cursor down past row 2 to A3. Press the SPACE BAR, and type the label INCOME <RETURN>. In cell A4, enter the label SALARY. Now skip cell A5 and label A6 INTEREST:. Below INTEREST:, in cells A7 and A8, enter the labels STOCKS and SAVINGS. Just before entering each of these labels, however, press the SPACE BAR twice (instead of once), so that these subcategories will be indented. (See Figure 6–3, shown on pages 171–172.)

Let's suppose card-playing accounts for some of your monthly income (or expenses, if you lose). Type CARDS in cell A9. Label row A10 OTHER. Now skip A11, and in A12 type TOTAL: <RETURN>.

In A13, press the SPACE BAR and then type five hyphens (-----). This uses the label mode graphically — to divide INCOME from EXPENSES.

Now label the rows that designate specific expenses, as they appear in Figure 6–3. Since you can only see 21 rows at a time on your screen, use the ↑CRSR↓ key to reach the cells below A21.

Varying the Column Width and "Blanking" Cells

Notice that as you type, the label appears simultaneously in the cell and in the status line. The cell allows only eight characters to appear, but Easy CALC RESULT remembers up to eighteen characters in a labeled cell. All the characters appear in the status line, but only the first eight appear in the cell — *unless you expand the column width.*

The width of the columns can be set for from five to eighteen characters. The main restriction is that all the columns must be the same width (except for a titles column to be discussed ahead).

Easy CALC RESULT Commands

F7

The F7 key is used to enter the Command Mode. Except for entering labels, values, and formulas, you must press F7 before doing anything else with this program. F7 calls the main menu on the status line. (See Figure 6–6 for a command overview.)

To change the column from the initial eight-character width, type:

F7 G C

F7 (the function key)
g (for Global)
c (for Column)

and then type in a number from five to eighteen followed by <RETURN>. Experiment with different length labels to see how the feature works. Remember, the column width affects only what you *see* in the cell; the full label (up to eighteen characters) is stored in memory.

Once a label or value has been entered with <RETURN>, it can't be altered with the DEL key. The information in a cell is thus *protected*, unless you decide to erase it entirely and enter a new label or value. To erase a label, type:

F7 B

F7 (the function key)
B (for Blank)

This returns a labeled cell to its blank condition.

After labeling the rows, press the HOME key twice to bring the cursor back to A1. Now label the columns, entering "JAN," "FEB," and "MAR" in cells B1, C1, and D1, respectively. To simplify the example, only the first quarter-year is labeled on the sample spreadsheet; but if you're creating a spreadsheet for future use, label columns E through M with the remaining months.

Right-Aligning Labels

When labeling, the text is aligned against the left margin. The values, however, are going to be right-aligned, because that's how they're normally displayed on a financial worksheet. To line up the label JAN with the values that come under it (as in Figure 6–3), you must right-align the labels.

If you wish to right-align the labels now, simply enter the label, press <RETURN>, and then press the following keys (while the cursor remains over the label to be aligned):

F7 F R

F7 (the function key)
F (Format)
R (Right-align)

Here's a quick review of the rules for labeling rows and columns:

1. Press the SPACE BAR before typing the label.
2. For capital letters, enter the label as a value, press RETURN, and then press SHIFT-SPACE BAR.
3. Indent subheadings by inserting spaces before typing the label.
4. To right-align a label (usually used in labeling columns), enter the label, press RETURN, and then press F7 F R.

> Instead of pressing RETURN to enter a label (or value), you can press one of the CRSR keys. This moves the cursor to the next cell and enters data in the preceding cell with just one key stroke.

Entering Values and Formulas

Enter values (numeric constants) by moving the cursor to the appropriate cell, typing the value, and pressing RETURN (or one of the CRSR keys). The program assumes that each number you enter is positive, unless it's preceded by the minus sign (−). If a value is too large for the width of the column — which is expandable to a maximum of eighteen characters — the value is converted to scientific notation (see Chapter 3, pages 76–78.)

Enter the values in the JAN column (column B), but stop when you reach B12. Cell B12 requires a formula.

The real power of an electronic spreadsheet is found in its calculation and recalculation functions. A formula indicates the desired mathematical operation and the cells involved. In this case, you want Easy CALC RESULT to add the sources of your income (B4 through B10), and then display the total in cell B12. There are two ways to accomplish this.

1. You can use the plus sign (+), entering the following formula in B12: B4+B7+B8+B9+B10 <RETURN>. Although the complete formula can't be displayed within an eight-character cell, the whole formula appears on the status line and can be entered in memory. When you enter the formula, Easy CALC RESULT performs the calculation immediately and displays the total.

To observe the *re*calculation function at work, change one of the values in the cells B4, B7, B8, B9, or B10, and notice that the computed value in B12 changes accordingly. Essentially, that's how spreadsheets answer what-if questions — by recalculating the same formula, using new or hypothetical values.

2. The formula can also be expressed in a shorthand code:

SUM(B4:B10) <RETURN>

The colon (:) means "calculate the sum of B4 through B10 inclusively." Use a comma to separate sets of consecutive rows. For example, the formula

SUM(D2:D10,F3:F8)

totals the values in D2 through D10 and F3 through F8.

The symbols for arithmetic operations are the same as in BASIC:

↑ exponentiation
* multiplication
/ division
+ addition
− subtraction

Easy CALC RESULT also understands the relational operators <, >, =, <=, >=, and <>. It also obeys the logical commands NOT, AND, OR, and IF . . . THEN . . . ELSE. For a discussion of the logic commands, which are beyond the scope of this book, consult section 2.4.9 of the Easy CALC RESULT manual. The manual presupposes knowledge of the arithmetic and relational operators. (See Chapter 3 of this book for a discussion of these subjects.)

N/A

When a formula cannot be calculated because the values for its arguments are missing, an N/A appears in red in the formula cell. To illustrate this using cell B11 (which is blank), enter the formula B2 + B4. Since there are no values in the cells to be added, the result will be an N/A in B11. N/A stands for "Not Available." Before continuing, be sure to "blank" B11 by leaving the cursor at cell B11 and typing the F7 key and B.

Now fill in the values under EXPENSES. When you reach B31, enter the formula for computing January's total expenses:

SUM(B15:B30) <RETURN>

Complete column B by entering a formula to compute the month's net profit in B33:

B12-B31 <RETURN>

As you type, the formula appears in the status line, and when you press RETURN (or one of the CRSR keys), Easy CALC RESULT performs the calculation, displaying the result in B33.

In the next section, you will see how to complete columns C and D of the spreadsheet by *replicating* values and formulas. The replication function can dramatically reduce the amount of time required to fill in the cells of a spreadsheet.

Replicating Values and Formulas

Some figures in your family budget are going to be repeated monthly. Your salary is one example of this (unless you get a raise or suffer a pay cut). When entering the same value in several cells, the Replicate command saves you time and reduces the possibility of typing errors.

To enter your salary (B4) in the salary row of the remaining months (February and March):

F7 E R

You type: F7 (the command line)
e (Edit)
r (Replicate)

Computer:
```
SOURCE RANGE TARGET RANGE
FROM TO FROM TO
```

SOURCE refers to the cell(s) containing the values to be replicated. TARGET refers to the cell(s) where the replicated values are to be placed.

There are two ways to replicate B4 in C4 and D4:

1. Move the cursor to B4 and press RETURN *twice* (because the source of the replicated material comes from only one cell). Then move the cursor to C4 <RETURN> and D4 <RETURN>. Watch the status line each time you press RETURN. As you'll notice, the SOURCE and TARGET cells are identified by the cursor position.
2. Another method involves typing in the SOURCE and TARGET ranges, pressing RETURN as each cell location is typed. That is, B4 <RETURN>, B4 <RETURN>, C4 <RETURN>, D4 <RETURN>. This method is quicker if you know the TARGET range. Suppose, however, that you're filling in your (expected) salary for the next two years. It takes more time to figure out which column will contain December of the second year than it

does to hit the cursor key twenty-three times (followed by RETURN).

Before filling in the remaining INCOME values, move the cursor down to row 15 and replicate your mortgage payments in columns C and D. For practice, use both methods of replication. If you want additional practice, replicate your January mortgage payment through the end of December (columns C through M). (If you have a variable rate mortgage, however, the replication may not reflect your actual payments.)

Now fill in the remaining INCOME values for February. Stop when you reach C12, because C12 requires a formula totaling the sources of income. You created this formula for B12: SUM(B4:B10), and now you want to use the same formula — *except that it totals the values in column C.* The same holds true for column D.

The goal is to replicate the formula in B12, and to make that formula *relative* to the values in the TARGET columns. Follow the same procedure as above, except for the SOURCE and TARGET ranges:

You type: B12 <RETURN>
B12 <RETURN>
C12 <RETURN>
D12 <RETURN>
Computer: ABS. OR REL B4

Easy CALC RESULT asks whether you want to apply the formula to the same values computed in B12 (ABSOLUTE) or to the values related to each column containing the formula (RELATIVE). Since your monthly total includes only the value from the month in question:

You type: R <RETURN>
Computer: ABS OR REL B10
You type: R <RETURN>

For each one of the values mentioned in the formula — in this case B4 and B10 — you will be given an option to select ABSOLUTE or RELATIVE.

Now fill in the remainder of the values under EXPENSES and replicate the formulas in B31 and B33 in columns C and D.

Order of Recalculation, Averaging, and the $-Mode

The calculations in the BUDGET spreadsheet calculate values by columns. When specifying the cells in a formula, the calculation can cross the boundaries of rows and columns. For example, to add January's NET PROFIT to February's TOTAL INCOME, enter the formula B33+C12. However, to calculate a set of values inclusively with the colon (:), you must occasionally change the calculation system to work by rows.

Suppose you wish to compute the average amount of money spent on gas and oil for the family car for January through March. Use the MEAN function to compute the average of a set of values — in this case B26, C26, and D26. To complete this experiment, label A36 "AVE. GAS/OIL" and then move the cursor to B36. You want to compute the formula MEAN(B26:D26), but now the computation is by row instead of by column. That's what the letter C (in AC) means in the upper right corner of the screen. To change to calculation by rows:

F7 O

You type: F7 (command line)
o (Order of recalculation)
Computer: (The C in the status line changes to R)
You type: MEAN(B26:D26) <RETURN>

The average expense ($50.8767) is computed *to four decimal places.* Unless it is told otherwise, Easy CALC RESULT's computations are precise up to twelve decimal places. To round off numbers to two decimal places, position the cursor in B36, and:

F7 F $

You type: F7 (the command line)
F (format options)
$ (round off to two decimal places)

The average expense is rounded off to the Dollar-and-cents Mode — to two decimal places.

Automatic Easy CALC RESULT Computations

Beyond taking a sum with SUM and an average with MEAN, Easy CALC RESULT performs other calculations automatically. Use these key words in a formula to perform the corresponding calculation:

MAX	select the maximum value in the specified cells.
MIN	select the minimum value in the specified cells.
STDDEV	calculate the standard deviation in the specified cells.

Using parentheses, the cells can be specified individually (separated by commas), or as a series (connected by the colon), or both. For example:

MAX(B3:B12,D4:D9,F14)

The Net Present Value can be calculated by using NPV within a formula that specifies the percent and the cells. The cells can include an area, or block of cells, as in the following example:

NPV(.08,B2:F10)

There are other math functions available. They are listed in section 3.12.2 of the Easy CALC RESULT manual, but there are no examples given. It is assumed that if you have a use for advanced functions, you'll know how to implement them.

Creating a Titles Column and Varying Its Width

As an experiment, move the cursor to column E, which would be the month of April in a full year's spreadsheet. Filling in the values and formulas for this month is complicated by the fact that the row labels (column A) are no longer visible.

There's an easy solution to this problem: create a special titles column that moves along with the cursor, always remaining on the left of the screen. In effect, you are going to create a spare column A, which is placed permanently (until you remove it) next to the row numbers.

To tell Easy CALC RESULT to set a titles column, first move the cursor to the home position (cell A1) — assuming that's where your labels are located. (The fix-titles command creates a spare titles column of *whichever column the cursor is in.*) When the cursor is in the labels column (normally, column A),

F7 E T

You type: F7 (the command line)
e (Edit)
t (fix Titles)

The screen responds with another column A, as you see in Figure 6–4. Now move the cursor to column E or beyond. The titles column remains permanently in view.

```
    A          A          B          C
 1BUDGET    BUDGET          JAN       FEB
 2
 3INCOME:   INCOME:
 4SALARY    SALARY      2607.31   2607.31
 5
 6INTERESTINTEREST
 7 STOCKS    STOCKS      304.58    267.87
 8 SAVINGS   SAVINGS     718.98    769.73
 9CARDS     CARDS        -24.5        34
10MISC.     MISC.            0         0
11
12TOTAL:    TOTAL:      3606.37   3678.91
13-----     -----
14EXPENSESEXPENSES
15MORTGAGEMORTGAGE       867.73    867.73
16
17UTIL:     UTIL:
18  GAS       GAS        64.09     57.87
19 ELEC      ELEC        78.61     59.08
20  OIL       OIL       123.65         0
21WATER     WATER            0
```

Figure 6–4: Sample Titles Column.

Removing the titles is simple. With the cursor anywhere on the screen, press F7 and L. The titles column disappears. (You should delete the titles column before printing the first four columns of the spreadsheet with the F6 command. If you don't, you'll have duplicate labels.)

The titles column also allows for a greater column width to accommodate your labels. As you've seen, the column width can be varied from five to eighteen characters *globally* — all columns will take on the same width. Often, however, it helps to have a wider titles column to leave room for text. Easy CALC RESULT gives you this option in the following procedure:

F7 G C

1. *Remove* the titles column (if necessary).
2. Expand the column width globally by pressing F7, G, C, and the desired width of the titles column (say 15).
3. Type in labels up to fifteen characters.
4. Now reduce the column width to normal (whatever is sufficient for viewing the values; say, eight characters).
5. The permanent titles column is left unchanged, while the other columns are reduced to eight characters.

Saving and Loading

Saving the spreadsheet is an essential part of financial management. In the next section, you'll learn to print parts or all of the spreadsheet, but in most cases you'll also want a copy stored on tape or disk. A saved copy enables you to modify the values easily, to consult your financial situation, and to ask what-if questions without having to reenter the formulas and labels.

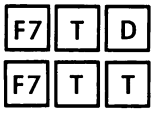
The Easy CALC RESULT manual does a poor job of explaining how to save spreadsheets, and the software itself omits some crucial screen prompts as well. The SAVE function works dependably, but it's important to know what is taking place behind the scenes if you want to avoid trouble. *Read this section carefully before saving a valuable spreadsheet.*

Easy CALC RESULT saves the whole spreadsheet, even if you only fill in a single cell. Thus, the procedure can take some time — nine minutes to save to tape and about one minute to save to disk. You can save a total of nine spreadsheets on a single diskette; each spreadsheet takes up sixty-six blocks of disk space. The number of spreadsheets you can save on tape depends upon the length of the tape. With diskettes, it's a good idea to reserve a diskette solely for Easy CALC RESULT files. Then you can predict how much room you'll have for saving another spreadsheet.

To save a spreadsheet, you must give it a name. The file name is *not* the name you typed in cell A1, although you can use the same name if you like. The point is that you must type in a file name at a specific time during the save procedure. But there's no screen prompt to tell you to type in the name. Without ques-

tion, that is a flaw in the software, but so long as you're aware of the problem, the flaw won't affect the program's operation.

To save the spreadsheet currently on the screen:



You type: F7 (the command line)
t (for Transport)
t (if you're using Tape)
d (if you're using a Disk Drive)
Computer: LOAD , SAVE
You type: s (for Save)

Now, *watch out*! When you press *s*, a blue square (the status line's cursor) on the second status line moves to the right one position. That's the only indication you'll get that you must now type the file name of this spreadsheet.

The file name can have from one to twelve characters or spaces. Do *not* use any of the following special characters:

' " : ; , # $ * ?

Warning

It's a mistake to think that *your* spreadsheet title in cell A1 is automatically the file name. In this case, the natural mistake would be to press RETURN after pressing *s* — without supplying a separate file name. If you make that mistake, the spreadsheet will be saved with no file name, and you won't be able to load it later on.

A second problem with the save function is that you can't save-and-replace. When a saved file is loaded and modified, you must select a different file name to save the revised version. If you try to save it by typing the same file name again, the drive stops working and the red error light flashes. This is extremely inconvenient.

The remedy is to leave two spare character places in the file name for a number. You can then save your spreadsheets as BUDGET1, BUDGET2, BUDGET3, and so on. Using the normal SCRATCH procedures (see Chapter 2), you can later erase the outdated versions. Easy CALC RESULT provides no means for deleting spreadsheets or for viewing a directory of the saved files. Use the BASIC command (LOAD "$" — or LOAD "$",8 with a

disk drive) to see the file directory. It pays to keep your diskette labels current, so you don't have to return to BASIC to recall the file name.

> **Warning**
>
> If you forget the file name of your spreadsheet, you must shut down the system, remove the Easy CALC RESULT cartridge, and boot up the system in BASIC. After reading the disk directory, shut the system down again and restart Easy CALC RESULT. Remember, *never remove or insert the program cartridge while the computer is on.* Ignoring this rule will in all probability ruin the cartridge.

Make a backup copy of a spreadsheet by loading the spreadsheet, inserting another tape or diskette, and saving the spreadsheet to the backup medium.

To load a spreadsheet, follow the same steps as in the save procedure, except press *l* (for load) instead of *s*. As in the save procedure, Easy CALC RESULT doesn't tell you when to type the file name. Following the procedure outlined above, type the file name after you press *l*.

The What-If? Mode

Calculating the outcome of possible scenarios is an essential step in financial planning and decision-making. Without knowing the dollars-and-cents results of various financial moves, there's no way to evaluate and compare them.

The spreadsheet's power to answer what-if questions comes from its ability to recalculate formulas with new values. Using the spreadsheet as a financial advisor can be as easy as substituting hypothetical values for the current values in some of the cells.

For example, suppose the utility company announces a 10 percent increase in domestic gas supplies and in electricity costs for the coming year. Simultaneously, one of your high-tech stocks splits and promises to earn another $45 per month in interest. What if these predictions come to pass? Based on the first-quarter spreadsheet called BUDGET, you could project the effects of these changes by:

1. substituting the new values in cells B7-D7,
2. substituting the new values in cells B18-D18 and B19-D19, and
3. checking the NET PROFIT columns B33-D33. (Formulas are recalculated automatically.)

If the changes you envision are more pervasive and complicated, you'll want a second version of the spreadsheet for experimenting. Otherwise, the current values will have to be reentered to preserve this year's BUDGET.

Make a copy of your spreadsheet with the copy command.

F7 E C

You type: F7 (the command line)
e (for Edit)
c (for Copy)

Computer: SOURCE RANGE TARGET RANGE
FROM TO FROM TO

The copy command lets you take the entire spreadsheet, or a block of information from it, and duplicate it in an area of equal size. (Copy is different from replicate, which works only within specific columns or rows.) To define the block to be copied, indicate the SOURCE RANGE as the top left cell and the bottom right cell.

You type: A1 <RETURN>
D33 <RETURN>

Then select the top left cell of the TARGET RANGE:

You type: A40 <RETURN>

A duplicate spreadsheet now appears in the area defined by A40 and D72. When using the Copy command, make sure that there are no other cells in use in the TARGET RANGE, because the copied block will replace them. Figure 6–5 shows how the copy command works in the example described above.

Now change the label in A40 to read BUDGET#2. With a duplicate of the original spreadsheet on the screen, you can experiment with possible values without having to worry about reentering the original information. Move the cursor to the TOTAL income for January (B51), and notice the formula in the status line: SUM(B43:B49). The formulas remain relative to their new

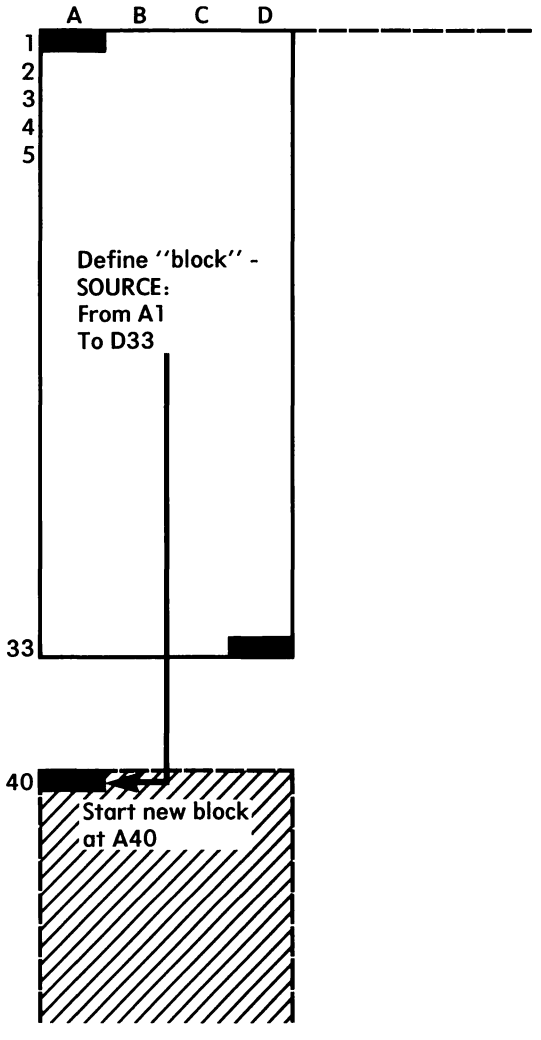


Figure 6–5: The Copy Command.

location on the spreadsheet, so they will total the new or hypothetical values.

With a little practice and imagination, you can use the same technique to answer questions about investment opportunities, the potential yield of an I.R.A. or a Keogh plan at various annual contributions and rates of interest, or any plan expressible by numerical values ordered in columns and rows.

Taking Command of Your Spreadsheet with the F7 Key

As was just demonstrated, you can do more with a spreadsheet than enter information in the cells. You can also erase information, control the appearance of the columns, and implement more powerful features, such as creating a bar graph to compare selected values, copying a block of data from one area of the worksheet to another, printing your results in one of three formats, and so on.

F7

The key that opens the door to these options is the F7 function key. You may have noticed when you pressed F7 to adjust the column width and to blank (erase the contents of) a cell that a cryptic string of letters appeared in the status line.

```
          SYSTEM: B E F G L O T R -
F3=GOTO    F6=PRINT    CLR=CLEAR
```

The letters in the first line represent the commands available to you in the Easy CALC RESULT program. (See Figure 6–6: Easy CALC RESULT Command Tree.) B stands for "blank," which you've just used. G stands for "global," meaning a change that affects *all* columns or rows. That command enabled you to change the width of the columns. (F3, F6, and CLR are discussed in the next section.)

Two of the commands lead to sublists of commands: E is for the edit commands. When the SYSTEM commands are on the screen, press E. The top status line changes to:

```
EDIT: C D G I M P R T
```

Now return to the SYSTEM commands by pressing F7 twice. Press F to see the format commands:

```
FORMAT: C G M I $ L R
```

You'll see many of the commands in action ahead, as you build your first spreadsheet. After reading about the others in the Easy CALC RESULT manual, practice them on your own. Although it's impossible to demonstrate every command in this chapter, Figure 6–6 describes the commands briefly and serves as a useful quick-reference tool for seeing at a glance what you can do with each command.

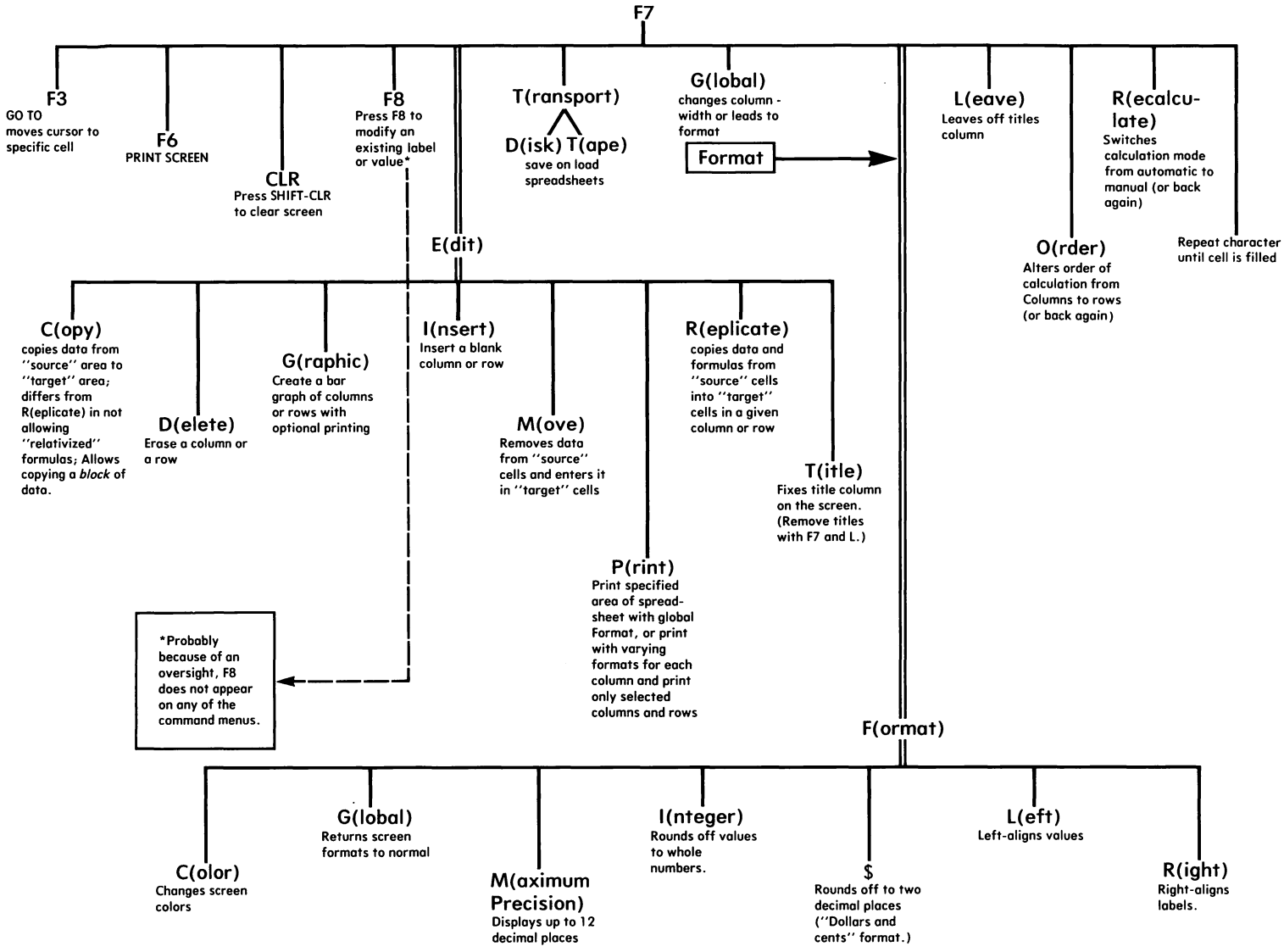


Figure 6–6: Easy CALC RESULT Command Tree.

F3, F6, and CLR

F3 F6 CLR

The commands on the second status lines (F3, F6, and CLR) appear on all three command menus. These are universally useful and easy to grasp. The F3 command lets you move the cursor directly to a selected cell:

You type: F7 (the command line)
F3 (the GOTO command)
Computer: GO TO (appears in status line)
You type: XXnn <RETURN> (XXnn = any cell location)

Using the F3 command, move the cursor to the following cells: A16, F3, AF126, BK214, and A1.

Although the CRSR keys suffice for moving to nearby cells, when working with large spreadsheets the F3 command can save time and key strokes.

F6 = SHIFT — F5

The F6 key instructs the printer to print whatever is on the screen (excluding the status lines). Using this command gives you an instant hard copy record of the current visible spreadsheet. When your printer is connected and turned on, try this feature at any time. Even if the spreadsheet is empty, F6 prints the column letters and row numbers.

Printing the full spreadsheet is more involved, because it must be printed in segments and because there are different formats available. The printing options are discussed in the last section of this chapter.

The CLR (actually SHIFT-CLR) command clears the spreadsheet of all labels, values, and formulas. In short, it starts you from scratch.

You type: F7 (the command line)
SHIFT-CLR
Computer: CLEAR
ARE YOU SURE? (Y/N)
You type: Y <RETURN> (for yes)
N (for no)

Typing Y resets the program to the opening screen, erasing everything in memory. Typing N tells Easy CALC RESULT to abandon the command.

Although F7 initiates every command, you can also abandon any command in progress by pressing F7. The F7 key toggles the program back and forth between the command line and the value-and-label entry mode.

Getting Graphic

Easy CALC RESULT is exceptional in providing the means to represent any section of the spreadsheet as a bar graph drawn to whatever scale you select. Let's compare the stock activity for the first three months in the sample BUDGET.

Move the cursor to the STOCKS row (A7). Then:

F7 E G

You type: F7 (the command line)
e (for Edit)
g (for Graphics)
Computer: GRAPHIC
COL. OR ROW
You type: R <RETURN>
Computer: LOWER LIMIT UPPER LIMIT

You can now set the scale against which the values are measured. Since the interest from stocks is between $200 and $400, use these values as the lower and upper limits.

You type: 200 <RETURN> 400 <RETURN>

Although there is no prompt on the screen, the computer now waits for you to type a NAME for the graph. Up to 32 characters are allowed. (If you don't want to add a name to the graph, press RETURN.) But let's call this graph MONTHLY STOCK YIELD. Type these words and press RETURN. A screen resembling Figure 6–7 on page 192 should appear.

To print the MONTHLY STOCK YIELD graph, type F6 (that is, SHIFT-F5). Make sure your printer has been connected — but do *not* connect it now, while the computer is running — and turned on. You can safely turn on the printer while Easy CALC RESULT is running. Notice that the cursor appears below the graph in the left-most column. You may now type a label under each bar (followed by <RETURN>) and then a descriptive subtitle on the bottom line (followed by <RETURN>). (See Figure

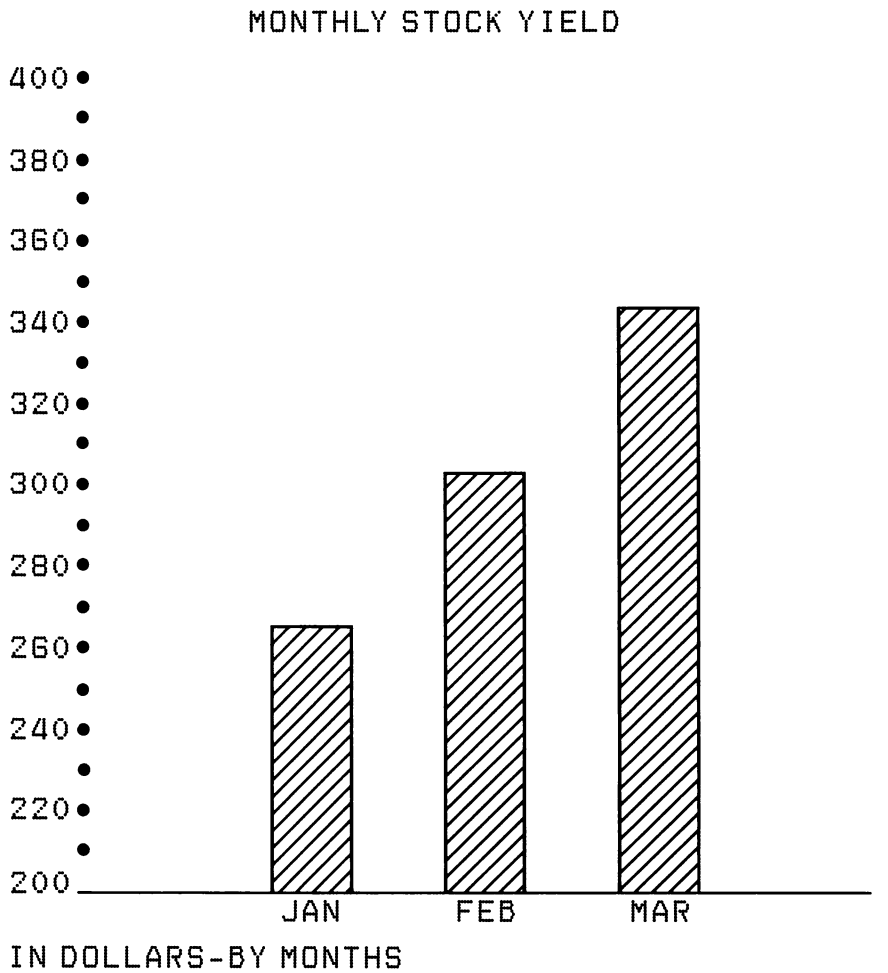


Figure 6–7: Easy CALC RESULT Bar Graphs. Labels on bottom two lines were created prior to printing.

6–7.) To print the graph without these additional labels, simply press RETURN twice.

(If you're not using a Commodore printer, you may find that the bars of the graph are composed of unusual graphics characters or not entirely filled in. There is no way to address this problem in general, because each nonstandard printer will need different modifications.)

Putting It Down in Black and White

Printed records of your spreadsheets can come in handy for easy reference at tax time, or, perhaps, for sending copies of what-if scenarios to a business partner or investment counselor.

There are three printing options. The simplest is the F6 command, which prints everything visible on the screen, except the contents of the status lines. Before using F6, position the window so that it reveals the block of information you want printed. Then press F7 and F6. (When printing graphics, the F6 command is used *without* F7.)

The second option is called *direct* printing. Just as in the copy command, you can define a block of the screen by selecting the top left cell and the bottom right cell of the block.

F7 E P

You type: F7 (the command line)
e (for Edit)
p (for Print)
Computer: `Format or Direct`
You type: d (for Direct)
Computer: `PRINT TO`

Indicate the cells that define the block by moving the cursor to each cell and pressing <RETURN>, or by typing each cell location (followed by<RETURN>). The printing follows immediately.

Some Reminders

Be sure not to define a block of the spreadsheet that's too large to be accommodated by your printer. Of course, you can fit more columns across a sheet of paper if you reduce the column width prior to selecting the print option.

Remember, you can cancel any command in progress by pressing the F7 key. Aborting a command returns you to the label-entry-value-entry mode.

The third printing option allows you complete control over which columns are printed and the individual column widths. This is called a *formatted* printout. Follow the same command sequence that you used for *direct* printing, until the computer displays FORMAT OR DIRECT.

You type: f (for formatted printout)
Computer: FIRST LAST
LINE LINE

The word *line* is misleading. This prompt is requesting the *rows* that you want printed. If you want rows 1 through 6 printed:

You type: 1 <RETURN>
6 <RETURN>
Computer: COLUMN
WIDTH

Here's another misleading prompt. Although it appears to be requesting the column width, it is really requesting the column letter first (followed by <RETURN>) and then the width-in-characters of that column. Thus, you can select which columns of information are to be printed and vary the widths of the individual columns. For the present example, your answer to the prompt might look like this:

A	B	C	D
12	8	8	8

But you could have used the *direct* print option in this case. In a hypothetical case, your request might be more selective:

A	C	E	AK	AL
15	3	10	6	6

Compared to what is available in other spreadsheets, the *formatted* printing option is extremely flexible, and it typifies the versatility of Easy CALC RESULT.

Keeping Your Records Straight with THE MANAGER*

As the previous two chapters demonstrated, the C-64 can automate your writing and bookkeeping, offering increased speed and capabilities in these areas. But the advantages of "electronic intelligence" are equally apparent in managing purely informational files, such as an address book, a collection of recipes, a catalogue of your book or record collection, a membership list, and a customer or client list. If you have data of this sort around the house or office, you may soon decide to retire your little black book and three-by-five index cards along with your portable typewriter and slide rule.

A database management system (or DBMS) enables you to store these informational files on one or two diskettes, and to search your files and reorganize them with speed and flexibility. Here are a few problems that are easily handled by a database manager, such as THE MANAGER.

1. In your Rolodex file, you have about 150 cards arranged alphabetically by last name. The cards list friends, relatives, a few business associates, and service people who have worked on your house, but the file includes dozens of obsolete address cards that you've been meaning to update or remove.

*THE MANAGER is available through Commodore dealers.

At 8:00 A.M., just as you're rushing off to work, a water pipe under the kitchen sink springs a leak. Fortunately, you filled out a card for the plumber who came to the house last year, the one who charged a reasonable rate and actually fixed the shower. Unfortunately, you don't remember his last name. You could have filed it under P for plumber, but a quick check of the Rolodex shows you didn't bother — you were so sure you'd remember his name. It was a long, unusual name. *Let's see, what was it, now* . . . You know you have another plumber on file, too, but you don't remember his name either. How about the Yellow Pages? Two pages of plumbers! And he may work out of a different city . . .

If that's how your day began, you'd be awfully late for work.

Let's see how THE MANAGER would handle the same situation. Boot up the program and select the REPORT GENERATE option (three minutes). Insert your data disk with your ADDRESS BOOK file on it, and request a REPORT to the screen with a SORT on the *field* containing the word *plumber*. Another minute, and you have both plumbers' "cards" in front of you.

Whether you "filed" the cards under *plumber* doesn't matter to a database manager. It can find information under any category.

Without dramatizing the point again, let's examine a few more situations that call for a computerized information file.

2. You're having a big Thanksgiving dinner for the whole family, which includes sending invitations to at least two dozen addresses. Not wanting to forget anyone, you use THE MANAGER to list all the entries containing the key word *relative*. With a printer and a sheet of stick-on labels, you can have the addresses of all relatives on file printed automatically.
3. You have on file a list of two hundred contributors to a charity. The entries include the amount of their donations. When you want to invite those who contributed $50 or more to a fund-raising dinner, you can SEARCH,

SORT, and LIST to the printer (alphabetically, if necessary) every $50-or-more donor.

4. You have over five hundred albums in your collection of rock music. The albums are shelved alphabetically by performing group. To celebrate the end of school, you decide to have a "Long, Tall Sally" party, playing all the recordings of that song in your collection. Since your database file includes the featured musicians and song titles on each record, you can easily search *across your collection* for all the albums with a "Long, Tall Sally" track.

There are surely enough examples here to reveal the advantages of a database management program. Now let's see how they work.

Databases come in a variety of sizes, structural shapes, and degrees of complexity. You'll often hear terms like hierarchical, network, and relational used to describe different kinds of databases. Some of these require a great deal more memory than microcomputers have, and they can be difficult to use. The relational database, for example, is the kind the Internal Revenue Service uses to compare the income you report on your 1040 Form with the amount of payments recorded under your social security number in totally separate files — including the W-2 and 1099 forms. Cross-file referencing is virtually unavailable in the microcomputer field, and certainly not at an affordable price; it's also virtually unnecessary.

Simply stated, THE MANAGER is a *file management system* that allows you to create files that can accommodate up to two thousand entries (depending upon the size and complexity of the entries), and to enter new information in the file, and revise or delete existing entries. You can also search those files for the data you need, sequenced in the way you request, and obtain *reports* on the screen or printed on paper. THE MANAGER can total numerical information contained on the entries, and it can *interface* with EASY SCRIPT (see Chapter 5), so you can compose documents containing information in the files.

The key to using a file management system is designing the entry. THE MANAGER requires you to use a *fixed format* for the entries. But once the data is filed, the format becomes flexible

— you can recall the data in any order. This is the principal difference between computerized and manual filing systems. Both formats are fixed when the data is entered, but with a manual system, *access* to the data is limited to the way in which it was entered. Figure 7–1 illustrates one way to visualize the difference.

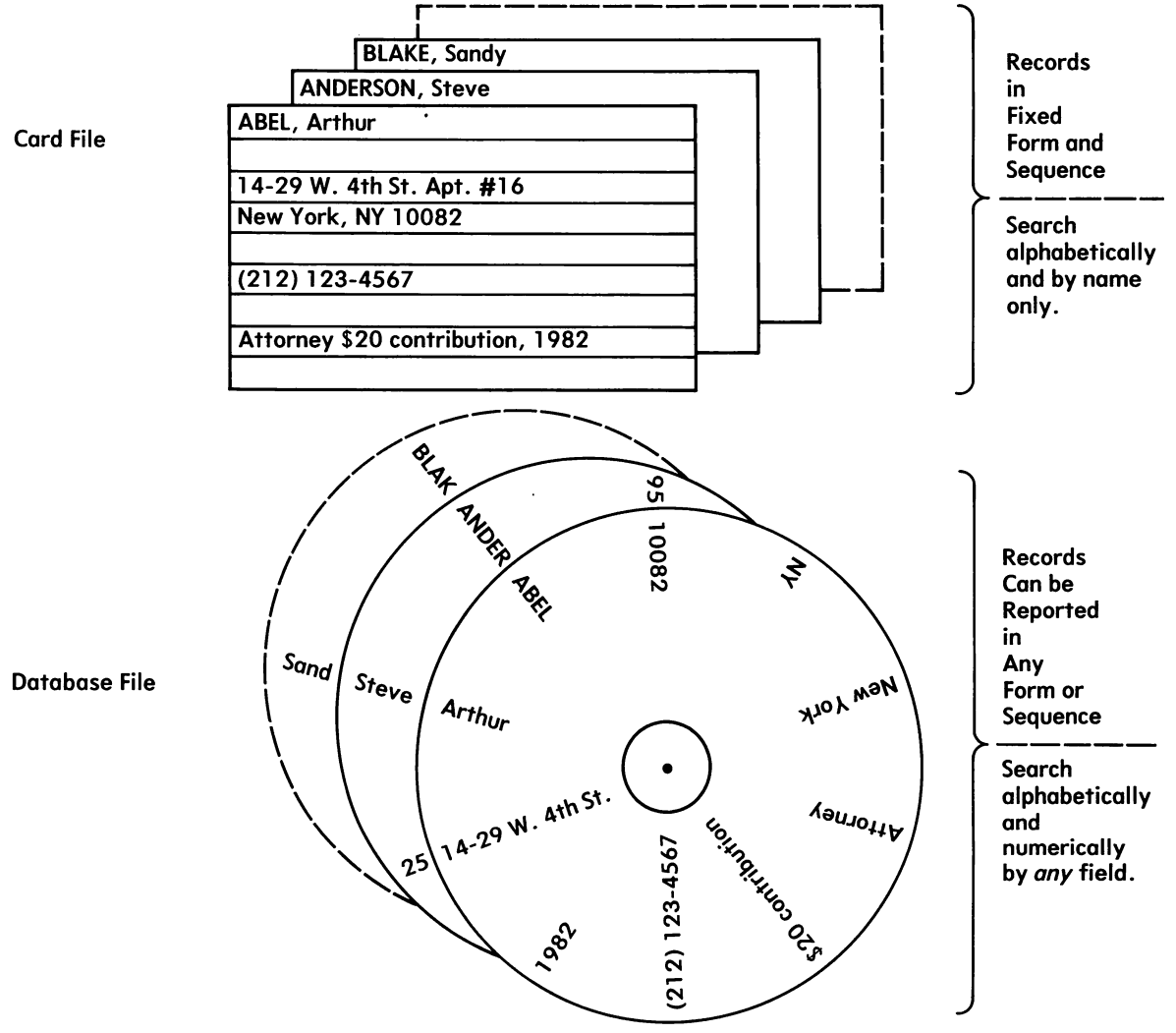


Figure 7–1: Comparing the Card File to the Database File. The circular shape of the database file entries doesn't mean that the information is actually stored on disk in a circular shape. Rather, the round shape is meant to suggest that there is equal access to the various categories of data.

Let's examine the components of a database management system.

Data: The most fundamental entity in a database system is the data itself. Data is composed of alphabetical or numerical characters, such as "Smith, Tom" or "(212) 123-4567."

Field: The field is a designated area of fixed length, where data of a certain type appears. For example,

LAST NAME:- - - - - - - - - - - - - - - - - - - -

is a field of twenty characters designed to contain an individual's last name. The name can be shorter than the maximum field length, but not longer.

Record: A record is a set of related data fields. A record is the technical name for what we've been calling an entry. The information on one three-by-five index card is usually comparable to a data record. Here are two simple records for examples:

NAME: Jones, R. T.
ADDRESS: 14 B STREET
CITY: NEWBURGH
STATE/ZIP: PA 19379
TEL: (215) 199-8765

AUTHOR: CAPRON, H. L., and WILLIAMS, BRIAN K.
TITLE: COMPUTERS AND DATA PROCESSING
PUBLISHER: BENJAMIN/CUMMINGS, MENLO PARK, CA
DATE: 1982
COMMENTS: COMPREHENSIVE REVIEW OF DP FOR BUSINESS

(Data) File: A file is a set of records sharing an identical structure. A bibliography consisting of twenty-five records designed like the sample above would be a data file. So long as the file structure is the same, the content (data) in the fields can be anything at all. You could have records devoted to books on sports, humor, art, or any other subject in the same file. (If you did, it would be advisable to add another field called SUBJECT: - - - - - - - - -. Then you could list all the books in the file on a given subject.) It doesn't matter to the database manager what you write in the fields — it can be total nonsense; the essential requirement

is that the fields appear in the same order and are of fixed length. That's what makes it a manageable data file.

Database: The database is a group of files that are related to your data file. There must be a file, for example, that tells the computer how your fields are ordered and how long they are. Another file may contain an index of names in the LAST NAME field arranged alphabetically. These support files are created by the database manager program automatically.

Database Management System: The system usually includes several programs that tell the computer how to manage the database. These programs enable you to enter new records, edit and update records, delete obsolete records, generate reports on the screen or printed reports, sort records alphabetically, and so on. In this sense, THE MANAGER is a database management system; once you create your data file, the collection of programs on THE MANAGER diskette is capable of performing the kinds of tasks outlined in the preceding pages.

Figure 7–2 provides a quick review of database terms.

Summary

There are two principal virtues of a file manager: First, you can easily add, delete, and modify records without the kind of messy results and overstuffed file boxes that are typical of manual filing.

Second, although you enter the information in one sequence or format, you can display or print it in a different order or format. That gives you instant access to the data that's most relevant at the moment, and in whatever form it's most useful to you.

In the case of THE MANAGER, a third advantage is that the software is compatible with the EASY SCRIPT word processor. Thus, you can insert selected data from your files without having to retype the entries.

Loading THE MANAGER

THE MANAGER is available only on diskette, and it is booted up like any other (machine language) program. Here are the steps:

1. Turn on the TV or monitor, disk drive, and printer.
2. Turn on the computer.

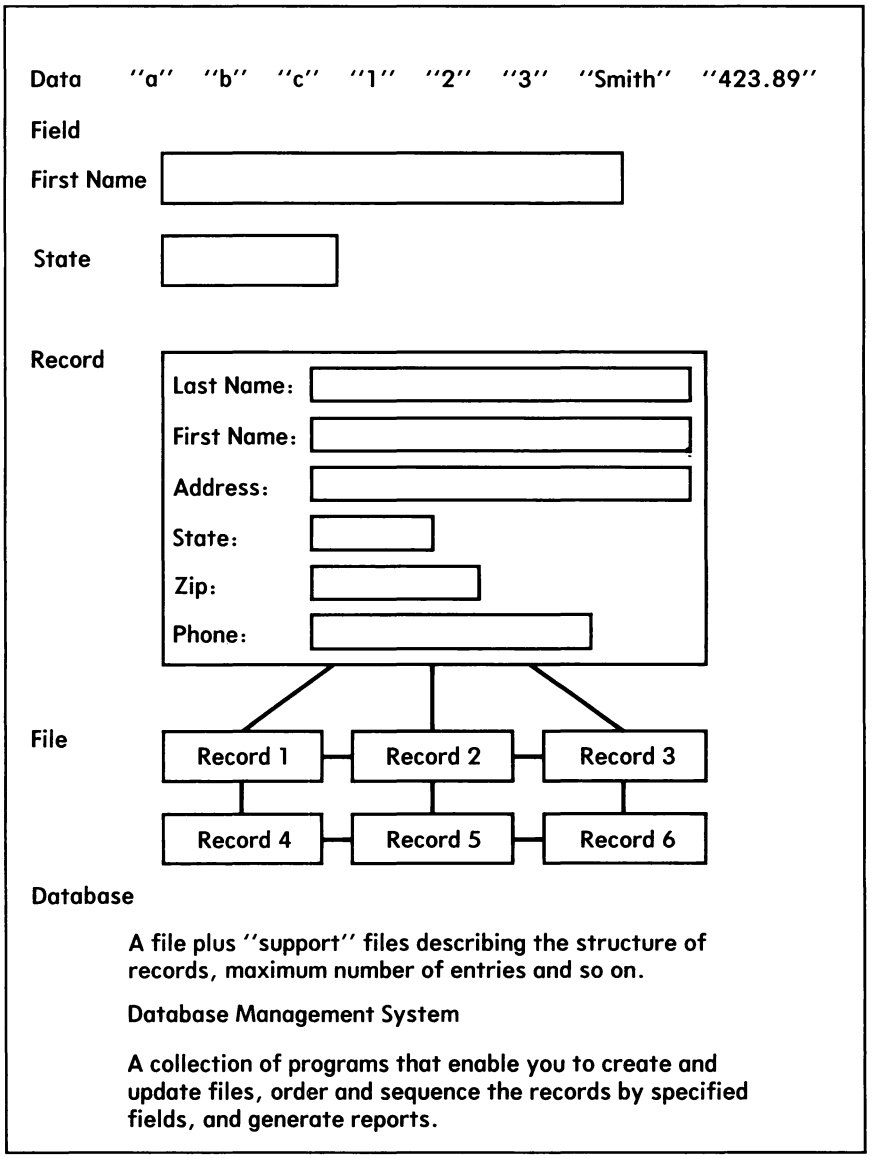


Figure 7–2: An Overview of Database Terminology.

3. Insert THE MANAGER diskette in the drive and latch the drive door.
4. Type LOAD "*",8,1 <RETURN>. Loading time: 2 minutes.

Keeping Up to Date

When you see THE MANAGER credit screen, notice the version number below the title. Version 1.06 is the most recent. About 5,000 copies of Version 1.04 were released earlier. There's no difference in how these two versions work, except that 1.06 runs slightly faster. (Commodore's policy is that an outdated version of a product can be exchanged for the recent version by sending the old version to Commodore along with a check for $5 to cover postage and handling. You may wish to call or write Commodore Customer Service, however, to confirm that the policy applies to THE MANAGER. In any case, keep a copy of your receipt of purchase.)

More crucial, the user's manual packaged with THE MANAGER is inadequate, omitting at least three essential commands. Although this chapter supplies the missing information, there could be further, undetected problems. Although Commodore is aware of the manual's inadequacy, there is no replacement as yet. If the first page of the manual contains the line "Copyright 1983, Commodore Electronics, Ltd." and the booklet contains exactly fifty-one numbered pages, you have the manual in question. Write Customer Service to determine if there is an alternative manual. If there is not, work through this chapter before trying to use the manual. Your patience will be rewarded; the software itself is powerful, and, after proper instruction, not difficult to use.

After the loading process is completed, you'll see the main menu on the screen (Figure 7–3 below):

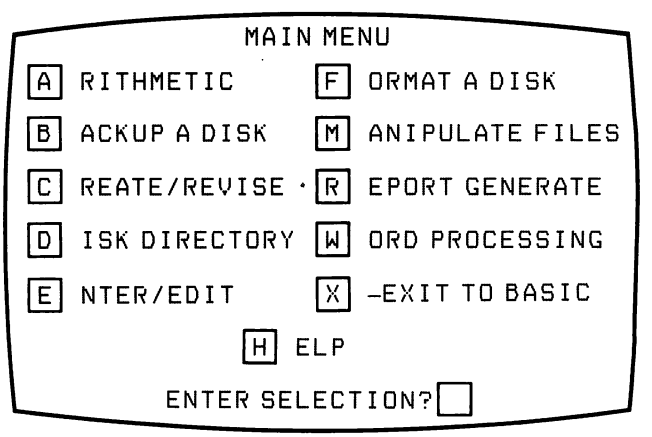


Figure 7–3: Main Menu.

F1

F2

Two Keys to THE MANAGER's Menus

When you want to return to the main menu, press F1. Pressing F1 is always followed by the question ARE YOU SURE?, because returning to the main menu means that anything you're working on will be abandoned.

Pressing F2 (that is, SHIFT-F1) returns you to the beginning of the option you're currently using. To *un*do your work and start again, press F2.

HELP (But Not Much)

The HELP option on the main menu presents an overview of the other menu options. It's an interesting way to get acquainted with THE MANAGER, but it does not include instructional material. Using the F7 key to advance each screen — F8 to see the previous screen — read through the HELP screens to get familiar with the options on the main menu.

As noted above, a database manager contains numerous files and programs that enable you to create and manage files. To look briefly at the files on your program diskette, press the directory option (D) <RETURN>. The next prompt asks you to choose:

```
[E] VERYTHING
[M] ANAGER FILES
[R] EPORT FILES

ENTER SELECTION? [E]
```

Pressing RETURN gives you the preselected choice. Select EVERYTHING. The first seven files listed in the directory are programs that make THE MANAGER work. The rest are related to ready-made "templates" included on the disk as samples. Because the manual is difficult to follow, it's actually easier to create your own files than to inspect the sample templates. The samples, moreover, are not likely to apply to your personal needs. You're better off learning to create your own files.

MANAGER FILES refers to the data files you create. REPORT FILES refers to report *formats* (the form in which you want data retrieved) that you save on disk for later use. These terms will be clarified as we create these types of files further on in the chapter.

Formatting a Data Disk

To save files created with THE MANAGER, you must use a formatted disk. You can either format a blank disk with the Wedge (see Chapter 2) or with THE MANAGER itself. If you decide to format the data disk with THE MANAGER, leave THE MANAGER program disk in the drive and press F <RETURN>. *After* the formatting program is loaded into the C-64's memory, the procedure continues as follows:

THE MANAGER: HAVE YOU PLACED THE DISKETTE YOU WISH TO FORMAT IN THE DRIVE (Y/N)?

Remove the program disk and insert a blank diskette.

Then:

You type: Y <RETURN>
THE MANAGER: ENTER DISKETTE TITLE?

You must now choose the name of the data disk. The limit is sixteen characters.

You type: DATA DISK <RETURN>
THE MANAGER: ENTER DISKETTE ID?

You must use two alphanumeric characters. (See Chapter 2 for a full discussion of the disk name and ID.)

You type: 15 <RETURN>
THE MANAGER: THIS WILL ERASE ANY INFORMATION YOU MAY HAVE ON THIS DISKETTE!!!
DO YOU WISH TO CONTINUE (Y/N)?

(*Warning:* Remember that formatting erases all the information on a disk. If you're not using a blank diskette, make sure the diskette to be formatted does not contain valuable files.) If a blank disk is in the drive, press Y <RETURN>.

THE MANAGER: FORMATTING DISKETTE

The formatting should take about a minute. At completion, press any key to return to the main menu.

Nonresident Programs

As you may recall (if you've read Chapter 5), booting up EASY SCRIPT loads the entire program into memory. You never have to reinsert the "system," or program, disk in the drive. Similarly, Easy CALC RESULT (Chapter 6) is effectively "resident" in the computer at all times, because the software is stored in a cartridge. But THE MANAGER includes several *sub*programs that are not loaded into memory when booted up. These are called *nonresident* parts of the system. Thus, it is sometimes necessary to reinsert THE MANAGER while working with your data files. When you need the program disk in the drive, THE MANAGER tells you:

```
PLACE SYSTEM DISK IN DRIVE — PRESS SPACE
```

or, when you need the data disk, THE MANAGER tells you:

```
PLACE DATA DISK IN DRIVE — PRESS SPACE
```

(SPACE refers to the SPACE BAR.)

Creating a File

C

Creating a file requires one of the subprograms stored on THE MANAGER diskette. Thus, remove the data disk from the drive and insert THE MANAGER. Then press C <RETURN>.

THE MANAGER: CREATE A NEW FILE/REVISE AN OLD ONE?
You type: C <RETURN>
THE MANAGER: ENTER FILENAME?
You type: ADDRESS BOOK <RETURN>

File names can be a maximum of thirteen characters or spaces. In this chapter we will create an address book file for practice, but you can apply the techniques to any file, or to an address file with different fields from those used here.

THE MANAGER: CREATE USING AN EXISTING SCREEN (Y/N)?
You type: N <RETURN>

If you already had designed a record for another file, THE MANAGER could borrow that format for the new file. Since this is your first file, you must design the record now.

THE MANAGER:
```
SELECT BORDER COLOR?
SELECT BACKGROUND COLOR?
SELECT CURSOR COLOR?
```

THE MANAGER can save your records in any color combination. These three prompts appear one at a time. If you want to change the screen colors, use the CTRL or COMMODORE key along with the color keys (see Chapter 1). It's easier, however, to simply press RETURN in response to each prompt, because the preselected screen colors are as legible as any other combination. But suit yourself. After completing the color requests, you'll see a blank screen with the following message at the bottom:

```
SCREEN 1 OF 1          L1   C1
```

The prompt means that this is the first screen of your record. You can have up to twenty screens per record. Using multiple screens is like having several three-by-five index cards associated with one name in your file box. This is generally applicable to business uses. One screen (card) would typically include an address, telephone number, and other vital statistics; the second screen (card) might contain the customer's payment history, purchasing volume, and so on. Let's try to get everything in our practice ADDRESS BOOK on *one* screen so it can be viewed more easily.

Designing a Record

Designing the record is probably the most important step in the creation of an information file. Beyond getting the necessary data in the smallest amount of space, you need to anticipate the fields for which you'll want to SEARCH and SORT in the future. You can, of course, REVISE the design of your records with the CREATE/REVISE option. But planning can eliminate a lot of tedious restructuring later.

Here's one possible record design that promotes flexibility and covers a variety of situations. Feel free to make whatever changes suit your needs.

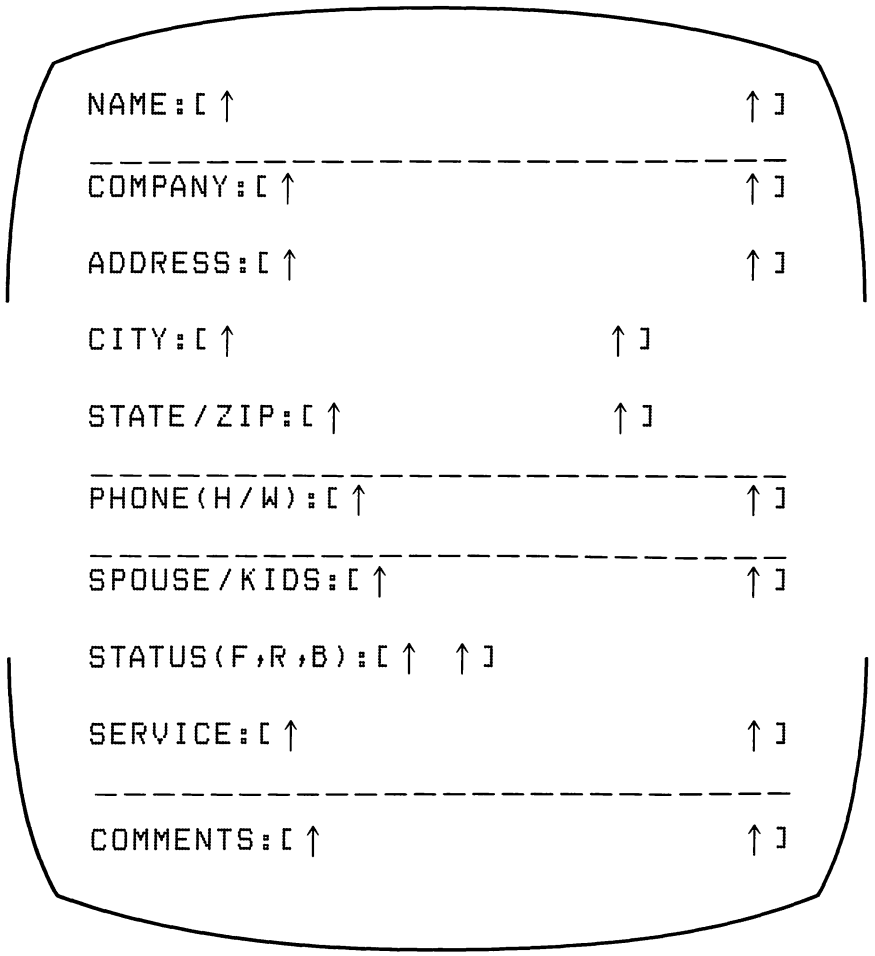


Figure 7–4: Record Format for an ADDRESS BOOK.

Step 1

Entering the fields as shown in Figure 7–4 can be accomplished using the keyboard normally. The brackets show you where the fields begin and end; the up arrow (↑) tells THE MANAGER that a field is being opened or closed. The arrows disappear when the actual records are filled with data. Blank lines are left between the fields to make them easier to read. Leave room in each field for the *longest* data that will appear there. The data you type in

later cannot exceed the predetermined space allotted for the field. The field, however, cannot exceed the forty-character line. [Status(F,R,B) means Friends, Relatives, or Business Associate.]

The dotted lines created with the hyphen key (-) help to isolate the name and phone number from the other fields. Design considerations such as these are strictly a matter of taste *and don't* affect the operation of the program. When you finish entering each field format, press RETURN. (Pressing RETURN, however, simply advances the cursor; it does not save the format. Saving the format is discussed in Step 2.)

Certain keys can facilitate improving upon the record design, once you have begun labeling and formatting the fields. To edit the screen design, place the cursor on the appropriate line, and press one of the following:

F3	deletes the line.
F4	inserts a blank line.
F5	memorizes a line.
F6	displays the memorized line.
F7	moves to next screen (of the same record).
F8	moves to previous screen (of the same record).

These keys affect the entire screen:

CTRL-I	inserts a screen (into the record).
CTRL-D	deletes the screen in view.
SHIFT-CLR	(same as CTRL-D).
CTRL-L	repeats previous screen design.
CTRL-C	change screen colors.

Step 2

When you're satisfied with the record's design, press the left arrow key (←). A prompt asks you ARE YOU SURE (Y/N)? If so, press Y <RETURN>. THE MANAGER asks you, ANOTHER SCREEN (Y/N)? Unless you want to add a second screen to this record, press N <RETURN>. Next, THE MANAGER reminds you to insert the data disk in the drive. Your record format is then saved onto disk.

(*Note:* If you forgot to press one of the up arrow keys (↑) to open or close a field, the message UNCLOSED FIELD appears

at the bottom left of the screen, and the cursor jumps to the problem field. Once the arrows and brackets are present in each field, the record format is saved onto disk.)

The potential size of your file is partly a function of the number of fields in the record and their lengths. Before the record format is stored, THE MANAGER gives you the total length reserved (in characters) of each record and the number of records allowable. If you've followed the sample format above, the message reads as follows:

REC LEN = 238, POSS NO OF REC= 397 OK?

Type Y <RETURN>. THE MANAGER then gives you an opportunity to specify how many records you'll actually need. If you want to save disk space and increase processing speed, type the lowest possible maximum number of records. If you think 100 address records is sufficient for your needs, type 100 <RETURN>. (Even if you underestimate, the consequences are not serious. The MANIPULATE FILES option allows you to extend the size of a file at any time.) THE MANAGER then asks:

DO YOU WISH TO ALTER FIELD TYPES?

If you have second thoughts about the fields in the record, type Y <RETURN>, and you'll be able to change them. Otherwise, press N <RETURN> and THE MANAGER creates your file one record at a time. A counter appears in the lower right of the screen. After a few minutes, THE MANAGER stores the screen format and returns you to the main menu.

Filling Up Your Address Book

Once the record format has been designed, the data can be entered into the file one record at a time. To fill out the address cards, use the keyboard like a typewriter to enter the information, and then *save* the record by pressing the left arrow key (←). First, however, you must select the ENTER/EDIT option from the main menu of the program disk. (The part of the program that lets you CREATE/REVISE a file isn't the same as the one that lets you ENTER/EDIT the data.) With the main menu on the screen:

E

You type: E <RETURN>
THE MANAGER: ENTER FILENAME?

(Remove the program disk and insert the data disk. Before you can enter data, THE MANAGER must load the ADDRESS BOOK record format from the data disk.)

You type: ADDRESS BOOK <RETURN>

(If you forget to switch diskettes, THE MANAGER tells you FILE NOT FOUND — DIRECTORY (Y/N)? This prompt anticipates the day when you'll have several data disks and may need to search the directories for your file. At this point, simply press N <RETURN>, remove the program disk, and insert the data disk.)

THE MANAGER now loads the record format for your ADDRESS BOOK file and displays it on the screen. The bottom of the screen looks like this:

```
SCR 1 REC 0        A,E,F,G,I,P,S,↑,=
```

The left side indicates the screen number (there's only one for this file) and the record number. When you save your first record, it will be record number 1. The right side lists a submenu of options. (See Figure 7–9 for a quick look at what each command accomplishes.) Begin entering data by pressing E(nter).

Print Hints

Pressing P (with the ENTER/EDIT submenu on the screen) prints the screen. Generally, printing is accomplished with the REPORT GENERATE option, which sequences the records and eliminates the field names. You may, however, wish to print a copy of the record format (REC 0) to remind yourself of the *order* of the fields. In formatting a report to the screen or printer, it is essential to know the order of the fields.

Begin entering data by pressing E(nter). The cursor moves to the first field. Type the practice data below into the fields, or type in the actual names, addresses, and other data from your card file or (manual) address book. To advance the cursor after typing in a field, either press the CRSR key or RETURN. Remember, when the data has been entered, you can save the record by pressing (←).

Use the INST/DEL key and CRSR keys for editing. Once several records have been entered and saved, you can experiment with the other commands in the submenu.

Below are three records designed to facilitate practicing some of the other MANAGER commands. These "dummy" records can easily be deleted when you're through with them. (You can, instead, use the actual data from your address book.)

> THE MANAGER stores records in the order that they are entered. It's unnecessary to alphabetize the entries because THE MANAGER allows you to do that with the SORT and INDEX commands.

Enter the dummy records in the order below:

```
NAME: [SMITH, JOHN & MARY                          ]
COMPANY:[                                          ]
ADDRESS:[135 S STREET                              ]
CITY: [NEW YORK                              ]
STATE/ZIP:[NY 10035                          ]
PHONE (H/W):[(212) 222-0987; 445-6000              ]
SPOUSE/KIDS;[ SON: KEVIN, 4 YRS. OLD               ]
STATUS;[F ]
SERVICE:[                                          ]
COMMENTS;[JOHN'S BIRTHDAY: JULY 17.                ]
```

Saves Record

After entering the record, press [←]. Wait briefly as THE MANAGER saves the record onto the data disk. Notice that a new menu appears on the lower right of the screen:

```
A ,C ,D ,E ,G ,P ,S , + , − , =
```

Press [+] to move you forward one record. (See Figure 7–9 on page 226 for an overview of these commands.) You should now see the blank REC 2 on the screen. The record has an ampersand (&) in the first field. Press E(nter) again; the (&) disappears and the cursor moves to the first field.

Fill in the fields for RECORDS 2 and 3, saving both by pressing (←).

```
NAME: [BAKER, BARBARA                              ]
COMPANY:[BAKER-HARRIS, CO.                         ]
```

```
ADDRESS:[12-75 8TH AVE                    ]
CITY: [NEW YORK                  ]
STATE/ZIP:[NY 10016              ]
PHONE (H/W):[WORK: (212) 816-5313         ]
SPOUSE/KIDS;[                             ]
STATUS;[B ]
SERVICE:[TYPESETTING & GRAPHICS           ]
COMMENTS;[PRINTED XMAS CARDS, 1983        ]

NAME: [ABEL, ALAN & CLAIRE                ]
COMPANY:[                                 ]
ADDRESS:[456 ACORN AVE, APT. #21C         ]
CITY: [LOS ANGELES               ]
STATE/ZIP:[CA 90022              ]
PHONE (H/W):[(213) 429-0078               ]
SPOUSE/KIDS;[JANET AND BOBBY              ]
STATUS;[R ]
SERVICE:[                                 ]
COMMENTS;[SECOND COUSINS                  ]
```

Although the manual neglects to mention this point, SHIFT-CLR returns you to the submenu at the bottom of the record-entry screen. If you wish to abort or restart any of the command options ahead, press SHIFT-CLR. Remember, too, that F2 returns you to the start of the option (in this case ENTER/EDIT), and F1 returns you to the MAIN MENU.

GETting Records

G

This option on the submenu lets you display a record on the screen by its number. Records are numbered consecutively in the order that they're entered. If you suspect a record you want is near the beginning of the file, or if you want to scan the first few entries, then use the GET (g) command to get started:

You type: g
THE MANAGER: RECORD NUMBER?

Type the number of the record you want to see and press RETURN. Once the record is displayed, you can "thumb through"

the other records in chronological order with the plus (+) and minus (−) commands.

SEARCHing Your File

S

The SEARCH (s) option allows you to look for specific data in a given field of your records. After you type s, the word SEARCH appears at the bottom of the screen and the cursor moves to the first field. Use the CRSR keys to move the cursor to the desired field, and then type the data you're looking for. Type accurately; THE MANAGER searches for the data exactly as you type it and tries to match it with the data on a record.

If you're using the sample records shown above, search for the typesetter in your ADDRESS BOOK file.

1. Press s.
2. Move the cursor to the SERVICE field.
3. Type TYPESETTER. (*You're not through yet.*)

At this point, you must tell THE MANAGER whether you've typed the data in the exact position that it was originally entered (an F3 search), whether the data could be anywhere within the field (an F4 search), or whether you're requesting a *Boolean search* (F5) — requiring a comparison of the data using relational operators =, <, or >. Since Boolean searches are fairly complex and not often required, they are not discussed here.

F3 Search: Position Dependent

After typing the data, press F3 to indicate that the data is in the exact field position you've indicated. Suppose you're searching the NAME field for someone whose last name is CLARK; yet you have another record for "ROBERTS, CLARK." THE MANAGER won't know which CLARK (first name or last) you want, unless you indicate the field position.

F4 Search: Position Independent

Suppose you're searching for a typesetter, but you're not sure if you wrote out the word completely; you might also have entered "INEXPENSIVE TYPESETTER," or "GOOD TYPESETTER" in the

SERVICE field. Thus, after typing the data, press F4 to indicate that you're not sure exactly where in the field the data is positioned. In fact, you can simply type "TYPESE" F4 and achieve the same thing.


Start Search

4. Type F3. (F4 also works.)
5. Press [←] to initiate the search.

Multiple Field Searches

Searching for data in two or more fields is a useful feature in scanning large files. Suppose you had four typesetters in your file, but only one located in your hometown of Fort Collins, Colorado. You could then request a search of two fields: for "typesetter" and "Fort Collins." Remember to press F3 (or F4 if necessary) after *each* entry. Press [←] to initiate the search.

Indexing

Indexing allows you to create an alphabetical list of the data in any field. Using that index, you can search the file rapidly at any time. The index (which contains only one field) can be searched up to one hundred times faster than the file. In a typical address book, you will probably want to index by last name. In a business file, you may want to index by company name, or by the city in which the business is located. You can only create one index per file, so choose the most useful category for making a quick search. Although the index itself contains only one field, when the data you're looking for is located, the whole record is displayed. You can then "thumb through" the file records alphabetically (forward or backward) with the [+] and [−] commands.

The index option is on the submenu of the first ENTER/EDIT screen (REC 0). Thus, you must first return to the beginning of the ENTER/EDIT option by pressing F2.

You can create an index when REC 0 is on the screen — immediately after you select the ENTER/EDIT option. To create the index, you must supply the *field number* of the field to be indexed. The fields are numbered consecutively. Although it seems unnecessary, there is a command that shows you the field num-

bers. (To see the number of each field displayed, press SHIFT- ↑ . Then press SHIFT-CLR to return the screen to normal.)

To create an index:

You type: SHIFT-I
THE MANAGER: FIELD NUMBER? 1
You type: <RETURN>

(Field number 1 is preselected; to select a different field, type in the field number and press RETURN.) THE MANAGER now saves on disk an alphabetical index by last name.

With the index saved on disk, practice searching the index with the i command on the submenu. After pressing i, the cursor is positioned in the index field (field number 1). Type the name you want to search for and press RETURN. The record filed under the name you type should appear shortly. You can then "thumb through" the records alphabetically. + moves you ahead and − moves you back. (When you reach the last record, + has no effect; when you reach the first record, − has no effect.)

A Short Guide to Indexing

- Use the SHIFT-I command to create one index per file on the most frequently searched field, such as the NAME field.
- Use the i command (INDEX SEARCH) to search the file for data in the indexed field, such as "SMITH."
- Then move through the file alphabetically with the + and − commands.
- The index is alphabetized on the first twelve letters *only*. The names SHAKESPEARE, WILLIAM, and SHAKESPEARE, WILBUR would not necessarily appear in alphabetical order.
- The index is automatically updated when a new record is entered to the file. But the index is *not* updated automatically when a record is deleted. (See the next section.) When records are deleted from a file, you must recreate the index to reflect those deletions.

Changing and Deleting

C, D

When a record in your ADDRESS BOOK becomes dated because an individual's address has been changed, you can update the record by pressing option C on the submenu. The cursor moves

to the first field, and the record can be revised using the CRSR and INST/DEL keys. Type the updated information and then save the revised record by pressing [←].

When a record becomes obsolete because, for example, a business contact has left your area, you can delete the record by pressing option D on the submenu. When you press D, *THE* MANAGER asks you at the bottom of the screen ARE YOU SURE? To continue with the deletion, press Y <RETURN>. If the record contained an indexed field, the next message tells you:

```
INDEX FIELD ALTERED, ARE YOU SURE?
```

As noted above, the index is *not* updated when a record is deleted. If the deletion is completed, you will then have a name in your index for which no record exists. This causes a gap in your reports, if your report is based on the index. Ideally, you should re-create the index after deleting records from the file.

Now You See It, Now You *Still* See It

Deleting a record places an ampersand (&) in the first field of the record. *It does not delete the text in the record.* This is a very confusing way to indicate that a record has been deleted. Normally, you'd expect to see the text disappear. But old records never die (get eliminated entirely); they just develop ampersands in the first field.

To get used to this idea, try deleting record number 3 in your file. Then return to the beginning of the option by pressing F2. Now get record 3 with the g option. The deleted record number 3 remains, but there is an ampersand in the first character position to show you the record "isn't really there." That is, the record will not be listed in SORTs or REPORTs.

One virtue to these ghostly deleted records is that if you ever want the record back again, simply delete the ampersand and replace it with whatever letter was there originally. Then re-save the record with [←]. If you're certain you'll have no further use for the record, press SHIFT-CLR to erase the "remains" of deleted record number 3, and fill in the data for another record to take its place. Don't forget to save the new record with [←].

Generating Reports

R

The REPORT GENERATE option on the main menu is an important one. It lets you format the output from your files. If you want

to create a "phone book" from your ADDRESS BOOK file, or print a list of mailing addresses, or view a list of selected clients on the screen, the REPORT GENERATE option comes into play.

ENTER/EDIT also lets you examine your files, but only one record at a time. You must also view (or print) the entire record, including the field labels. The advantage to REPORT GENERATE is that you can list on the screen or printer all, or part of, the file; and you can present the information in any format.

The first time you request a report, you must specify the format line by line, and this requires a few minutes of planning and key pressing. Fortunately, you can save the format as a report file, so you only have to endure the procedure once.

For practice, let's generate an alphabetical list of the three sample records in our ADDRESS BOOK file. Those records were purposely entered nonalphabetically, so you could see how the SORT procedure works. (The SORT procedure is one that the manual fails to explain in full, so it's advisable to finish this chapter before studying this aspect of the manual.)

With the main menu on the screen and THE MANAGER disk in the drive, press R <RETURN>. The REPORT GENERATE option then loads, and you see:

THE MANAGER: REPORT FROM [K]EYBOARD OR [D]ISKFILE?

Essentially, this question means, Do you want to create your report using the keyboard, or do you already have a report file on disk? Next time, you can use one on disk, but for now:

You type: K <RETURN>
THE MANAGER: ENTER FILENAME?

Before responding, insert your data disk, because THE MANAGER is going to attempt to load the file. If your data disk isn't in the drive, a message asks you if you need a directory. Answer N <RETURN>, and place the data disk in the drive.

You type: ADDRESS BOOK <RETURN>
THE MANAGER: ENTER SEARCH CRITERIA?
You type: <RETURN>

If you are searching for records containing only specific data, then you need to specify the data. Suppose, for example, you want to

have listed only the addresses for people in your hometown. You would then SEARCH for the name of your hometown in the CITY field. (Refer to the manual for the ways to enter search criteria.) Since you want to list all the records, type <RETURN> to bypass the question.

THE MANAGER: IN ORDER BY [I]NDEX,
[S]ORT OR [F]ILE?

Your report can be based on three different files:

[I]

INDEX: If you have created an index for the file, then the records can be listed alphabetically by whatever field you have indexed. Since your work on the ADDRESS BOOK file included creating an index on the NAME field, you could list your records alphabetically by name from the index. (That would be the simplest way to generate the alphabetized report.) But since this is a learning experience, let's pretend there's no index.

[S]

SORT: If there's no index on disk, you need to SORT (or "order") the report by the desired field, in this case the NAME field.

[F]

FILE: A report based on the data file itself lists the records *in the order that they were entered*. For some uses, that may be adequate. But not for this one; an alphabetized address book is much easier to use.

You type: S <RETURN>
THE MANAGER: ENTER NUMBER OF SORT KEYS?
You type: 1 <RETURN>

A SORT key is simply the field by which you intend to order the records. It can be an alphabetical field, such as NAME, or a numerical field, such as INVOICE NO. or BALANCE DUE. THE MANAGER shows you the following display:

FIELD	LEN	ALPHA/NUM	ASCD/DESD
1	33	A	A

Press <RETURN> as the cursor moves from one area to the next. You are choosing the preselected values: field number 1; at the full character LENgth; in ALPHAbetical and ASCenDing order. After completing the specifications of the SORT key:

[←]
Saves Sort Keys

You type: [←]

The SORT instructions are now saved. (You can SORT on more than one key. If you keep a CHECKBOOK file, for example, you can SORT by the day of the month and also by the amount of the check. The list would then show the checks arranged by days and in ascending order by amount for each day. These would be *numeric* SORTs.)

THE MANAGER: OUTPUT TO [S]CREEN,
[P]RINTER, [D]ISK?
You type: S <RETURN>

If you have a printer, you can have the SORTed file printed immediately. Or, you can save it on disk for later use. The practice example selects the screen output so you can observe the results immediately, whether or not you have a printer.

THE MANAGER: ENTER LINE LENGTH? 39
You type: <RETURN>
THE MANAGER: ENTER NUMBER OF LINES PER PAGE? 24
You type: <RETURN>

These responses simply tell THE MANAGER that you want to use the full screen (thirty-nine characters by twenty-four lines) to display the records.

Formatting the Reports

THE MANAGER organizes your output into three "zones." These are called HEADER, LIST, and FOOTER. Each zone is subdivided into PRINT AREAS — places you wish to have data displayed. The first prompt appears as follows:

```
DEFINE HEADER      — 0 PRINT AREAS USED

PRESS SPACE SELECTS/RETURN EXECUTES
```

The HEADER is simply a title, page number, or some other information you want to add to the top of the screen (or page, if you're printing the report). You must supply the HEADER; it does not

come from your data file. For practice, let's add a title; press RETURN.

THE MANAGER: NUMBER OF LINES IN ZONE?
You type: 1 <RETURN>
THE MANAGER: . NUMBER OF LINES FROM TOP?
You type: 1 <RETURN>

This leaves 1 line for a title 1 line from the top of the screen (or page). You then see the display in Figure 7–5. (The responses have already been added and are explained ahead.) The DATA TYPEs you need be concerned with here are F (for field) and T (for text). Since we're adding text, rather than using data already in a field, select T <RETURN>. The cursor then moves to the TEXT/TITLE category. Type ADDRESS BOOK <RETURN>. For LENGTH OF AREA, type 39 <RETURN>, because we want to center the title in relation to the whole first line. Press <RETURN> for the LINE NUMBER and COLUMN NUMBER prompts. Answer Y <RETURN> to CENTERING (Y/N)? The cursor will now recycle through the prompts, until you signal that the HEADER is formatted to your satisfaction by pressing the [←] key.

```
                REPORT GENERATE
                HEADER ZONE

PRINT AREA #  1              10 AREAS OPEN

DATA TYPE (F-R-D-T): T  SUBSCRIPT:  1

TEXT/TITLE
ADDRESS BOOK
.........1.........2.........3.........4
LENGTH OF AREA: 39  LINE NUMBER     : 1

COLUMN NUMBER : 1   CENTERING (Y/N)?  Y

# OF DECIMALS :     ACCUMULATE (Y/N)? N

BREAK TYPE (NONE, LNFDS, PAGE)? N
```

Figure 7–5: Format Screen for HEADER.

When your screen looks like the one in Figure 7–5, press [←]. The next prompt — indicating that the HEADER has been formatted — looks like this:

```
DEFINE HEADER          - 1 PRINT AREAS USED
```

To progress to the LIST zone, where the data itself is reported, press RETURN. You should now see

```
DEFINE LIST            - 0 PRINT AREAS USED
```

Press RETURN to format the LIST zone.

THE MANAGER: NUMBER OF LINES IN ZONE?
You type: 3 <RETURN>
THE MANAGER: PRINT ALL OR BREAKS ONLY?
You type: A <RETURN>

The first question is unfortunately misleading, because the word ZONE is used equivocally. It no longer refers to the LIST zone, which is twenty-two lines long. Rather, it refers to the number of lines you want to allot to *each record*. To simplify the example, let's request a report of the names and phone numbers only. Line 1 is used for the name, and line 2 for the phone number. Line 3 is left blank, so the next record will be easier to read. In order to have a blank line between records, it's necessary to request one extra line in the LIST (record) zone.

ALL means that all the data requested will be displayed. (BREAKS applies to mathematical totals.)

You're ready to format the ADDRESS BOOK records for this report. Refer back to the record format we designed for this example, and take note of the field numbers.

Two Keys to Formatting the LIST Zone

Two essential points not explained by the manual are (1) that the SUBSCRIPT corresponds to the field number, and (2) that pressing F7 advances you to another SUBSCRIPT (field) and PRINT AREA.

The screen in Figure 7–6 instructs THE MANAGER to display (or print) F(ield) number 1 (that is, SUBSCRIPT 1) on LINE 1. When

your screen looks like Figure 7–6, press F7 to advance to PRINT AREA #2.

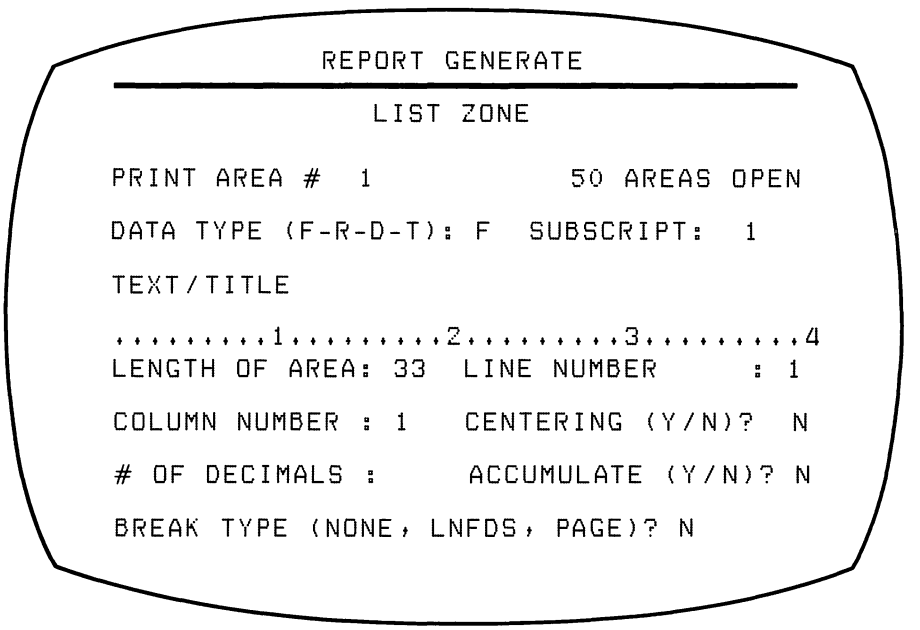


Figure 7–6: Format Screen for PRINT AREA #1.

The screen in Figure 7–7 instructs THE MANAGER to display (or print) F(ield) number 6 (SUBSCRIPT 6), which is the PHONE field, on line number 2. When the screen for PRINT AREA #2 looks like the one below, press [←] to indicate that you're satisfied with the LIST zone format.

After pressing [←], the screen displays:

```
DEFINE LIST        -2 PRINT AREAS USED
```

Press the SPACE BAR to advance to the FOOTER zone. Formatting the FOOTER is just like formatting the HEADER, except that you will want something different under the TEXT/TITLE category, such as "END OF PAGE 1." Format the FOOTER zone on your own. After saving the FOOTER format with [←], press the SPACE BAR again. The screen then displays:

```
DEFINE EXIT        -0 PRINT AREAS USED
```

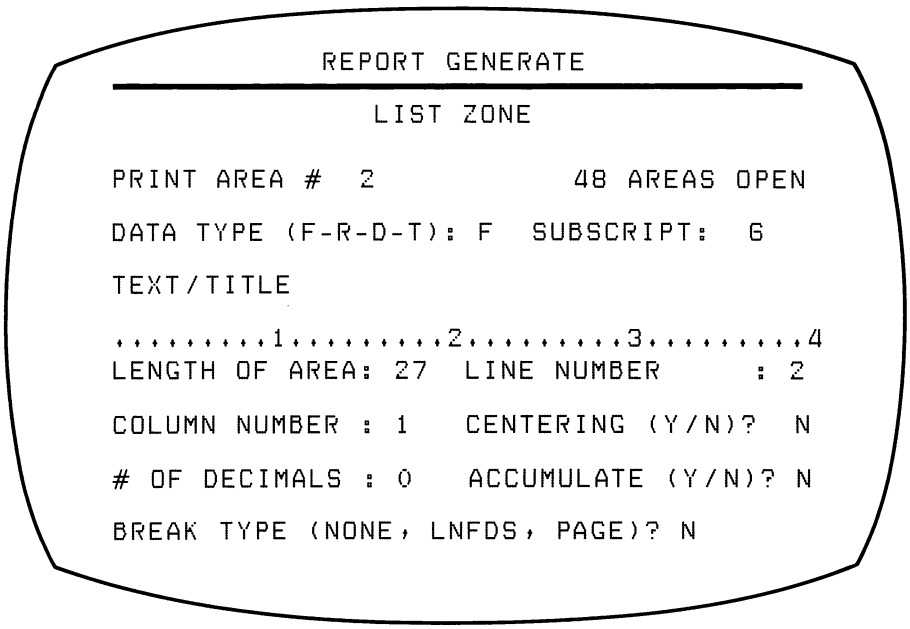


Figure 7–7: Format Screen for PRINT AREA #2.

You can recycle through the format zones by pressing the SPACE BAR, and reformat a zone by pressing RETURN when you see the name of the zone. When you're satisfied with the formatting, press RETURN. Answer Y <RETURN> to ARE YOU SURE?

THE MANAGER: SAVE THE REPORT CONDITIONS? Y
You type: <RETURN>

Warning

Although the manual neglects to mention it, you *must save the report conditions* in a "report file." In fact, if you answer N <RETURN>, the program will "crash" farther ahead in the procedure, and you'll have to turn off the computer and start over again. If you generate a report from an index, the index is already saved, so the problem doesn't arise. If you generate a report from the file, you do not need to save the report conditions. This requirement applies *only* to REPORT GENERATE with a SORT.

THE MANAGER: ENTER REPORT FILE NAME?
You type: ADDRESS BOOK <RETURN>
THE MANAGER: SORT THE FILE?
You type: Y <RETURN>

Now follow the prompts on the screen, inserting the program disk and data disk in the drive as instructed. After switching disks several times, you'll see the message REPORTING TO THE SCREEN. The report should look like Figure 7–8 below:

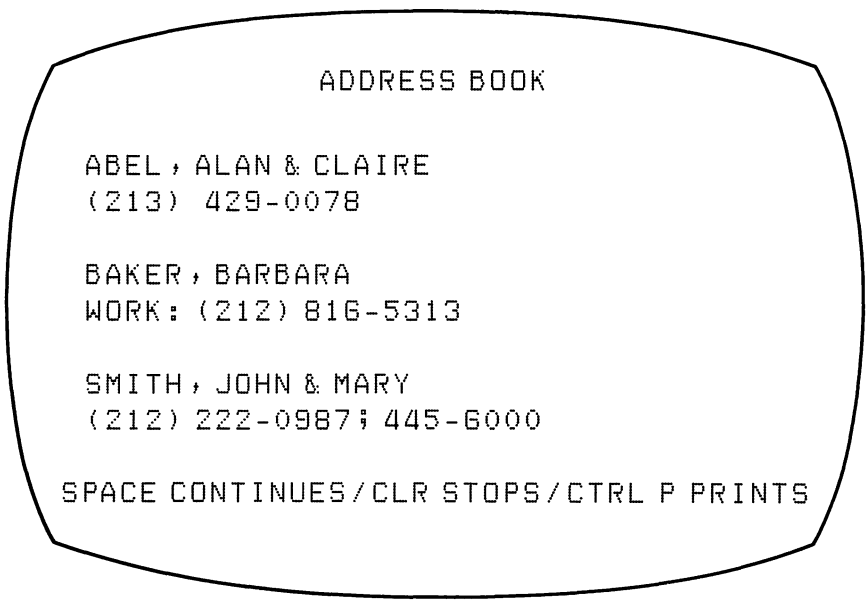


Figure 7–8: ADDRESS BOOK Report.

Note: If the message at the bottom of the screen did not appear, and the data disappeared quickly from the screen, then you failed to format a FOOTER. For some odd reason, THE MANAGER refuses to leave the reported data on the screen unless either (1) there's a footer, or (2) there are at least two screens of data being reported.

(The next time you want a phone book report from your ADDRESS BOOK file, select DISKFILE, when the first prompt appears on the screen.)

Press SPACE BAR to format another report, or to see the next screen, if there are more records to be displayed. Press SHIFT-CLR to end the report. Press CTRL-P to print the screen. (If you have selected output to the printer, the report prints continuously until completion.)

Deleting Zones

To eliminate a HEADER, FOOTER, or LIST format, display the formatting screen for a PRINT AREA, and then press SHIFT-CLR. The PRINT AREA will be erased one line at a time.

Managing Files

Despite a few quirks here and there and a flawed manual, THE MANAGER is a powerful database management system. It also offers the convenience of a self-contained disk file manager.

The MANIPULATE FILES option allows you to use the following commands:

BLANK A DATA FILE: Erases the data in a specified group of records without destroying the record format.

COPY A DATA FILE: Makes a copy of a single file on a backup disk (backing up the entire disk is accomplished from the main menu).

DISK DIRECTORY: Shows you the directory.

EXTEND A DATA FILE: Increases the number of records the file holds.

FIX A DATA FILE: Skilled programmers can enable THE MANAGER to work with data files created with other programs.

HELP FILE: (Same as the HELP screens available from the main menu.)

PRINT A DATA FILE: Prints the raw data as entered in the record format, including record numbers. For extremely valuable files, this can be used to create a hardcopy backup.

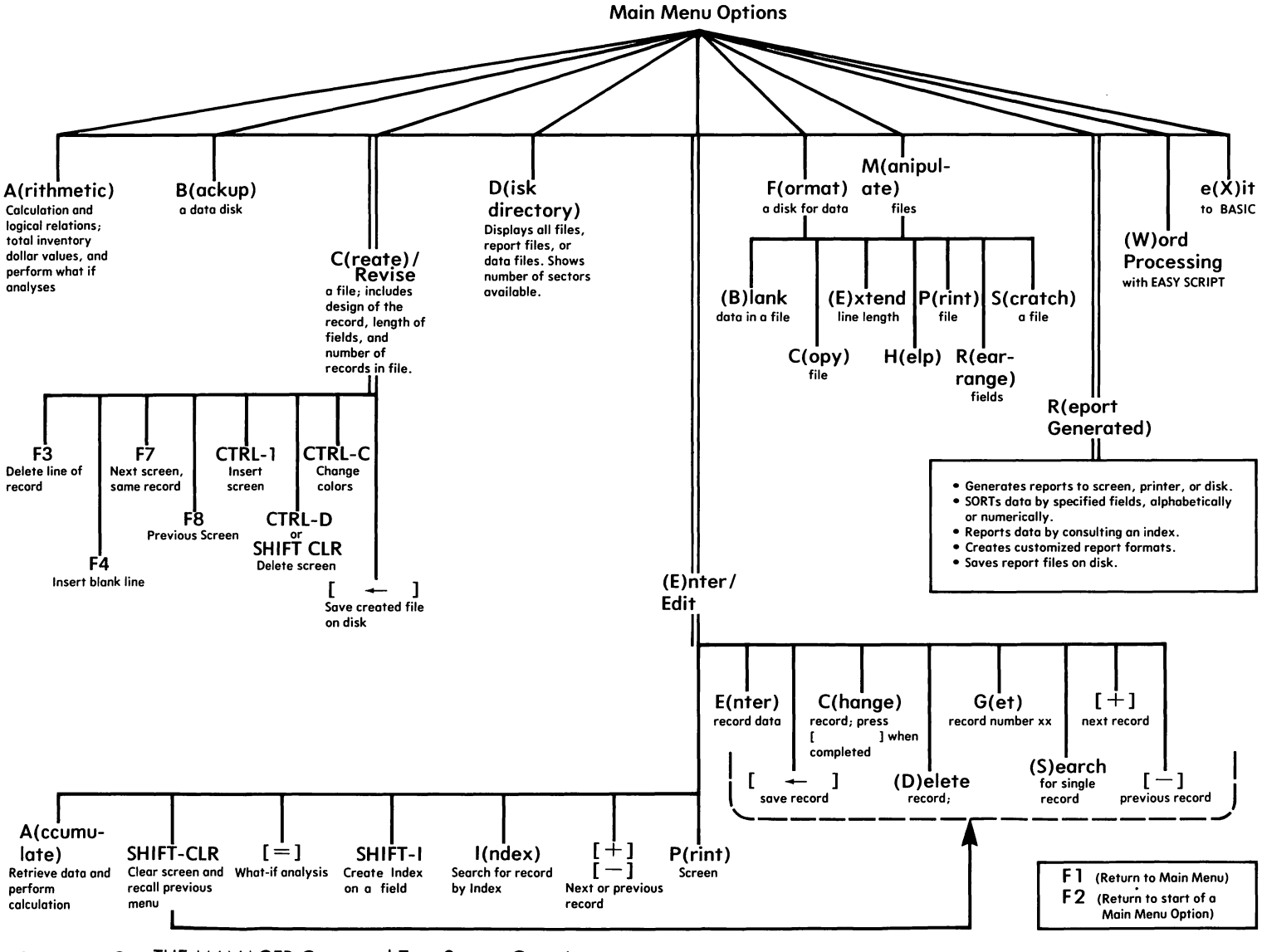
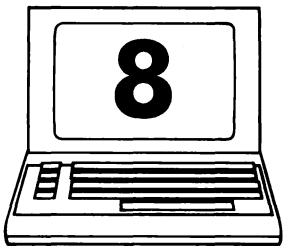


Figure 7–9: THE MANAGER Command Tree-System Overview.

REARRANGE A FILE: Transfers data from a "source" file to a newly formatted (or revised) "destination" file.
SCRATCH A FILE: Erases obsolete files from a disk.

These features are explained adequately in the manual, and they are not difficult to use. An overview of THE MANAGER features and many of its commonly used commands appears on the opposite page in Figure 7–9.



Telecomputing: Let Your Fingers Do the Talking*

Although you may already be delighted by what your C-64 can do, its most exciting application may be its ability to "talk" to other computers. With an ordinary telephone, even the most humble computer can acquire astonishing new power. Its reach becomes virtually boundless. Suddenly, your C-64 can use the world's most advanced software on sophisticated mainframe computers, exchange electronic mail, do electronic banking, and access huge data banks of information. Observe how easily *telecommunications* will fit naturally into your everyday life:

Monday

Problem: Your investment club is nervous about its investment in Amalgamated Doodads, Inc. There are rumors that Thingamajig International is about to release a competing product line. You'd like to sell the stock, but you need to hold on to it for another week to qualify for the quarterly dividends. You need to keep a close watch on it.

Solution: Using your Dow Jones account, you keep a "watch" on both Amalgamated Doodad and Thingamajig International. Each time you log on, all relevant data and news concerning these two companies is waiting for you. Time on line: 2 minutes.

*This chapter was contributed by Joe Campbell, the author of two books on computer communications and the author of a sophisticated communications program.

Tuesday

Problem: The pennant race is a close one, and your team is beginning a West Coast road trip. You can easily learn the scores from radio or television, but — since most of the games are played at night — details of last night's game won't be in the paper until Wednesday.

Solution: For the latest box scores, you log onto The Source, an information utility. You go immediately to the News and Information section. Here you request the UPI (United Press International) service, sports section. After responding to the simple menus, the box scores of the previous evening's games scroll across your screen. In addition, you can read the UPI reporter's story about each game — the same story you'll read in tomorrow's paper. Time on line for two box scores: 2 minutes 15 seconds.

Wednesday, 7:00 A.M.

Problem: While sleepily going through your morning ritual, you hear a news bulletin that an airline flight is reported missing. You feel your stomach tighten as you recall that a close friend is on that flight. There won't be another newscast for an hour and the airline's phone number is perpetually busy.

Solution: Log onto The Source again, this time going immediately to the UPI news service. Almost breathless, you watch the story come over the wire: your friend's flight had made an emergency landing at a small rural airport . . . all aboard safe.

Thursday

Problem: Your daughter is preparing to write an important term paper. You suggest that she start with the bibliography.

Solution: Using Knowledge Index, an online database service, she acquires the names of forty books and over a hundred magazine articles on her topic. Time on line: 11 minutes.

Friday, 9:30 A.M.

Problem: A new retailer in Hawaii needs a copy of your company's pricing for the coming week. The time difference makes it difficult for you to communicate directly; yet mail — even express mail — takes at least two days.

Solution: You notice that the dealer's letterhead lists ID numbers for two information services, The Source and CompuServe. By prior agreement, you deposit the entire price list (all twenty pages) into his "electronic mailbox" on CompuServe. When he "picks up" his mail next morning, he loads your price list into his own computer and prints it out in his office. Time on line: 12 minutes, 45 seconds.

Saturday

Problem: You have just written your neighborhood association's newsletter, and now need to print it on your friend's letter-quality printer. But the disk formatted by his computer is not compatible with the C-64 format.

Solution: Transfer your newsletter via modem to your friend's computer. As your text arrives at his modem, it is automatically written to his disk for subsequent printing. At 5 P.M. you pick up your printed newsletter.

Sunday

Problem: Tonight is the first meeting of the new Commodore users' group, but you don't know the time or place.

Solution: A hobbyist operates a remote bulletin board service (RBBS) in your area. When you call its number, you are immediately greeted with the announcement:

```
***WELCOME TO THE COMMODORE HOTLINE***
10/24/83 12:23
First meeting 10/24/83, 7:30 PM at
```

```
Marystown Middleschool. Contact Jim
Manning (522-1287) for details.
```

Time on line: 1 minute.

The Modem

The previous examples are illustrations of telecommunications or, as it has been called lately, *telecomputing*. At the heart of these powers is the common telephone and another simple device, the modem. Since you can't do much telecomputing unless you understand how these tools work together, this section starts with a brief explanation of how they work and then provides a more thorough discussion of their use.

A telephone's basic job is to convert sound (human speech) to electrical tones. This conversion takes place at the microphone inside the telephone's handset. The tones, in the form of electrical current, are then injected into the phone lines where they can be amplified, filtered, and manipulated. When they reach the receiver (the speaker in the handset), the electrical signals are changed back into sound. Figure 8–1 illustrates this simple conversion process. It would be convenient indeed if computer data had the same tone range as human speech. This would permit the direct connection of computers to the telephone lines. But computers transfer data by means of electrical impulses approximately *ten thousand* times higher in frequency than the human voice. To make matters worse, computer data is not composed of the relatively pure tones of human speech, but of a series of sharp pulses, or "bursts." Since the telephone network was designed to accommodate only the narrow band of voice frequencies, the high-pitched pulses of computer data would be completely rejected.

As a solution to this incompatibility, the microphone and speaker in the telephone handset have been replaced with components whose job is to convert the computer's high-speed current pulses into tones acceptable to the telephone system. On the receiving end, these tones are converted back to current pulses. In this way, computer data is represented as a stream of ordinary voice-band tones.

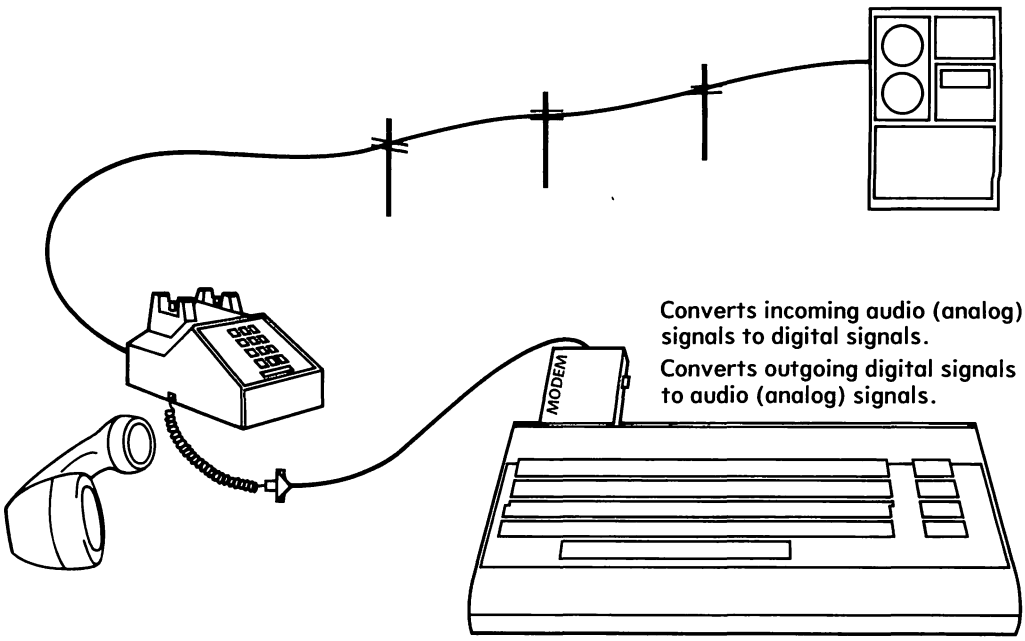


Figure 8–1: Modem. A modem MOdulates and DEModulates the signals between telephone lines and computers. *Note:* Handset cord is plugged into the VICMODEM.

> The name *modem* is a combination of *mod*ulate (converting the data to tones), and *demo*dulate (converting tones back to data).

The component that makes these data-to-tone and tone-to-data conversions is the modem. In other words, the modem serves as a special microphone and speaker for the computer's high-pitched voice. Although the modem is a single instrument, it is important to note that it is actually two separate devices — one for sending (the "microphone") and one for receiving (the "speaker"), and it can perform both functions simultaneously.

The VICMODEM

The VICMODEM is designed specifically for the VIC-20 and the C-64 computers. It connects to the computer through the expan-

sion connector (called the user port in the Commodore 64 *User's Guide*) located on the rear apron directly behind the COMMODORE 64 logo. Since the computer supplies the VICMODEM's power, installation is merely a matter of plugging the modem into its slot. As with all computer accessories, make sure the power to all components is *off* before connecting the modem.

Making the Phone Connection

The only equipment required to operate the VICMODEM is an ordinary telephone with modular connectors (RJ-11C) on the cord that runs between the handset and the phone cradle. If your phone doesn't have the appropriate RJ-type connector, you will have to purchase the VICMODEM Nonmodular Telephone Adapter, or an equivalent. Trimline telephones, which have the dialing mechanism located in the handset, are *not* suitable; their accidental use will cause no ill effects — they simply won't function correctly with the VICMODEM.

The VICMODEM has only a single switch, located on the inside edge of the modem. This switch is not clearly marked; the O position is *away* from the computer. Momentarily slide the switch to the A position (move it toward the C-64). The red connect indicator should illuminate. If the indicator doesn't glow, contact your dealer — either your C-64 or your VICMODEM could be at fault.

You are now ready to establish a connection with another modem — a *data link*. If you wish to initiate the phone call, put the VICMODEM switch in the O (for originate) position. If someone calls you, the switch should be in the A (for answer) position.

Actually, it doesn't matter which position the modem's switch is in, as long as the switch on the caller and the answerer are in *different* positions. The originate/answer convention is just an easy way to refer to the switch's position.

Let's assume you're initiating the call. First, place the VICMODEM in the O position. Now dial the telephone number. If a modem answers the phone, it will announce its presence with a continuous (and rather irritating) high-pitched tone. When you hear the tone, unplug the cord from your handset and plug it into the connector on the rear of your VICMODEM. The red

indicator should illuminate, showing that a data link has been established. That's all there is to it! No more buttons to push or numbers to dial. Modem operation is relatively straightforward and uncomplicated. Once the data link has been established between two modems, you need never think about them again until it's time to hang up.

> Federal regulations require you to report the connection of a modem to the telephone lines. Frankly, most users take this law about as seriously as the law that forbids removing the labels from pillows. While we don't advocate this attitude, be advised that phone companies are beginning to increase their rates on telephones on which modems are reported.

Now What?

You've plugged in your VICMODEM, called another modem, and successfully established the data link. You're ready to get down to some serious telecommunications. You type a few keys . . . nothing happens.

Strange though it may seem, the C-64 doesn't even know that the VICMODEM is connected. Even though your C-64 has a built-in connector for the modem, and although Commodore sells a modem for the C-64, the computer remains oblivious to it until you load software into the computer that enables it to send characters to the modem. Whenever you want a computer to perform some task that is not part of its native intelligence (that is, built into ROM), you need a program to provide the necessary intelligence. Here are the basic attributes of a simple modem program:

1. Characters that arrive via the modem should be displayed on the screen.
2. Characters typed on the keyboard should be dispatched to the modem.
3. Characters must be sent and received in various *transmission formats*, which we'll discuss shortly.

Although this short list describes a trivial program, it is powerful enough to handle most elementary communications requirements. Fortunately, 64 TERM, a program that fits this specification, accompanies each VICMODEM.

Caution

Early VICMODEMS are supposed to contain both VIC-20 *and* C-64 software, but the C-64 program is sometimes missing. Unless the VICMODEM's box explicitly states that it contains modem software for the C-64, ask to look at the tape before you buy. A modem without a modem program is valueless! If you've gotten stuck with one of these tapes, contact Commodore for an updated version.

The modem program that comes with the VICMODEM is on tape: VICTERM is on one side of the tape, 64 TERM on the other. If you have a disk drive, ask your Commodore dealer for a copy of the disk version when you buy your modem.

Load and run 64 TERM from either tape or diskette. After a brief copyright message, the screen clears and you'll see the following:

```
TERMINAL READY.
```

The message TERMINAL READY means that your modem program is operating. A *terminal* is the general name given to any device that performs the three functions listed above. For this reason, the terms *modem program* and *terminal program* are used interchangeably.

Going On Line

Included with your VICMODEM are several free offers, such as one hour each on the information services The Source and CompuServe, and on the Dow Jones News/Retrieval Service, a financial information service. These promotional offers expire from time to time, so your modem may not contain all three.

An *information service* is a computerized amalgamation of *written* information — news, stocks, reference material, electronic mail, bulletin boards, movie and book reviews, and so on. Recently, more exciting features such as electronic banking, direct conferencing, and computerized shopping have been added. By responding to menus or, optionally, by giving direct commands, you can select the information you wish to read. In some cases — such as news stories and bulletin boards — you can automatically *search* the system for a specific topic. This spares you the time and the expense of wading through irrelevant information before arriving at your subject of interest. But it is far more important that you become acquainted with the notion of telecomputing and some of its basic procedures than for you to be given a guided tour of several commercial services that are now available. Because CompuServe is the easiest to use (although not the most powerful) of the information services, we'll use it in all our examples in this chapter.

Most computer systems have two levels of security through which you must pass before you will be *logged on*, or admitted to the system. The first level is a *user ID*, generally a unique combination of numbers and letters that identifies you as an authorized user. The second level of security is the *password*, a single word consisting of eight to sixteen numbers and letters. The host system will ask you for your user ID and password. After you type your answers, it checks them for accuracy. If you give several incorrect answers, most systems hang up on you. Your CompuServe demonstration pack contains a pull-apart document on which you will find your user ID and a password. The Source gives you both ID and password over the phone. The Dow Jones demonstration pack contains an ID, but you have to phone for the password. All these phone calls are toll-free.

Although your password will initially be assigned, you should change it immediately. The information service provides a means of doing this. Be imaginative. Don't use variations of your name or the names of your family members, your address, or other obvious personal details. Change your password from time to time. You will get no sympathy from the information service if someone learns your password and then runs up a huge bill on your account.

Getting Started with CompuServe

Located in Columbus, Ohio, CompuServe maintains local telephone numbers in many cities around the country. Their network is called ComLink. A list of ComLink numbers should be in your modem packet, but if your city isn't listed, phone 800-848-8990 (in Ohio, 614-457-8650) to see if it has been added. If your town doesn't have a local CompuServe number, you may have to pay toll charges to the nearest number. Before you pay toll charges, however, read the section below on Telenet and Tymnet—two networks that also provide local access to CompuServe.

Logging On

CompuServe's user ID is composed of five numbers, a comma, followed by four numbers. The password may be eight to thirteen characters long, but *must* contain one character that is *not* alphanumeric — a letter or number. Your modem packet contains your ID and password. In these examples, the CompuServe user ID is 70005,5143 and the password SIXTY*FOUR. By the way, free-trial accounts with these services can be used only during the evenings. The more expensive daytime service is available only to regular subscribers. Each time you log on under the complementary user ID and password, you will be invited to subscribe on line. If you do so, you are allotted another two free hours of time to tide you over until CompuServe processes your paperwork.

> 70005,5143 is the actual CompuServe user ID of Joe Campbell, the author of this chapter. When you're on-line, feel free to leave a "postcard" in his mailbox. Technical comments and questions about this chapter are welcomed. His Source ID is STN984.

Load *64 TERM*, then dial the CompuServe access telephone number. When you hear the tone, plug the handset's cord into the VICMODEM. When the connect indicator illuminates, get CompuServe's attention by typing CTRL-C. Your screen should now look like the one shown in Figure 8–2 on page 238.

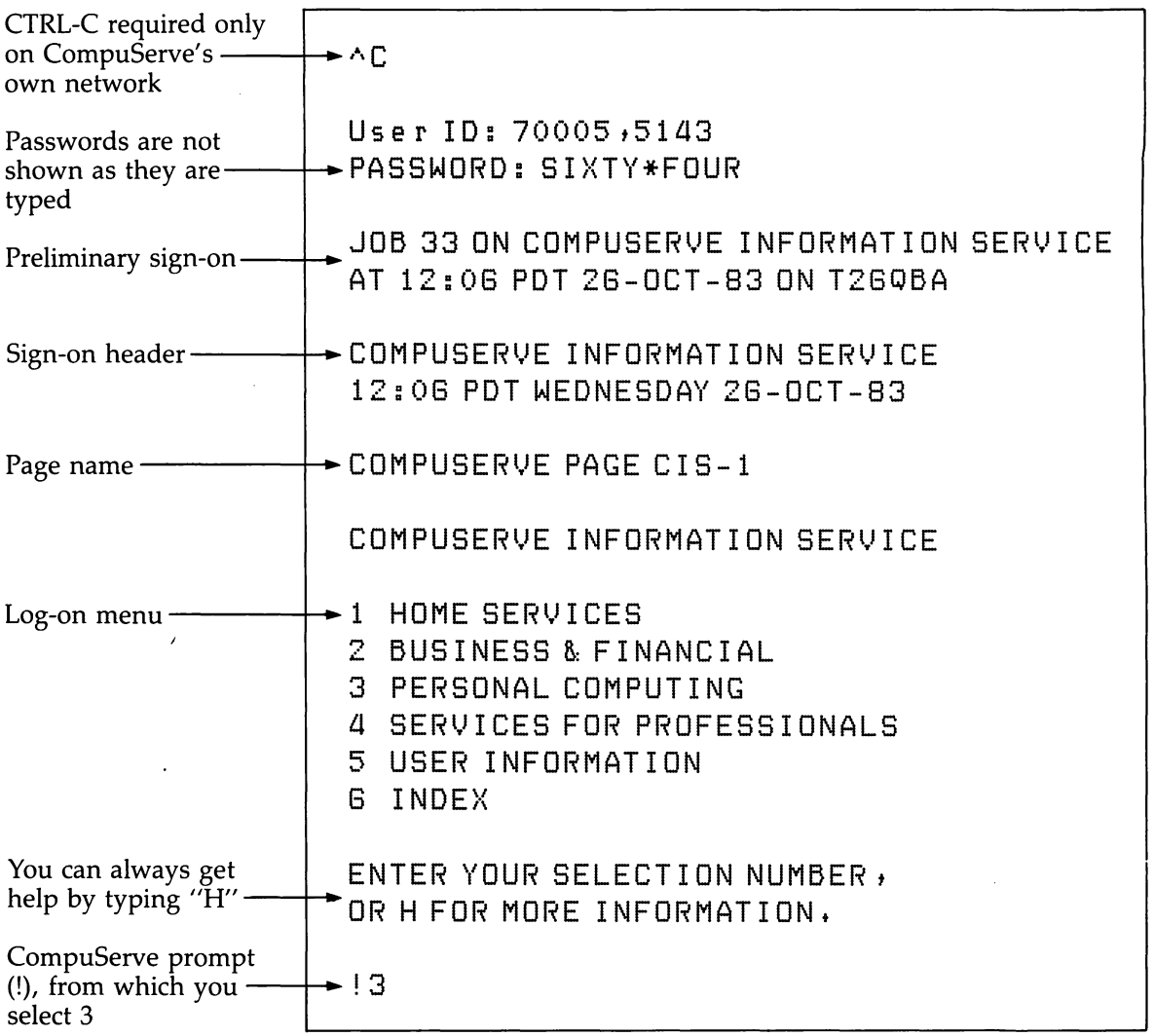


Figure 8–2: Logging on to CompuServe.

Most information services are a collection of *hierarchically-organized* menus. This means that there is a single trunk menu that leads to a group of branch menus. These branches may lead to yet more branches, and so on, until there are no more branches. At the end of any path will lie a list of functions from which to choose. In Figure 8–3, by starting at the main trunk menu, we climb through the branches until arriving at CompuServe's special Commodore section.

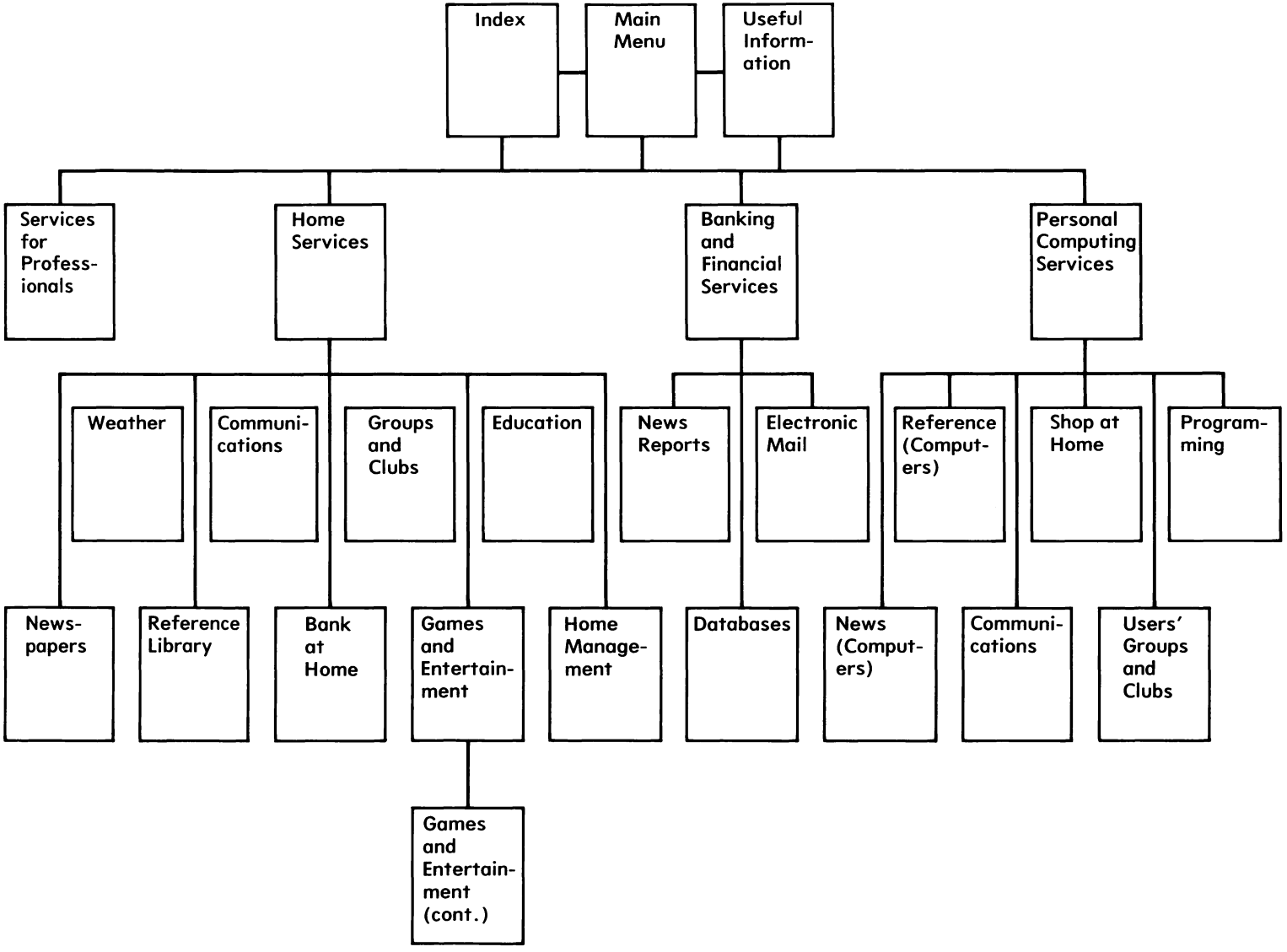


Figure 8–3: Overview of CompuServe's Menu Tree.

There are a few important points to learn from the on-line session shown in Figure 8–4. First, at times you will need to turn off and restart the host's output while you copy information from the screen. Most computer systems will stop sending if you type an X-OFF character — usually CTRL-S; transmission resumes when you send it an X-ON character — usually CTRL-Q. This feature is known as *flow control.* Not every system recognizes X-ON/X-OFF, and some systems use other control characters to represent them.

The next thing you will want to know about a host system is how to force it to stop whatever it is doing and accept your commands. *Every* system has a command that forces it to terminate a task prematurely. Luckily, most systems send only a few lines at a time before pausing for a command, but if you request it, they will provide continuous output. If we had requested continuous output in Figure 8–4, for example, we might have had to suffer through the entire bulletin board list. Remember, display time is connect time. As demonstrated in the example, CompuServe uses CTRL-P to stop the flow.

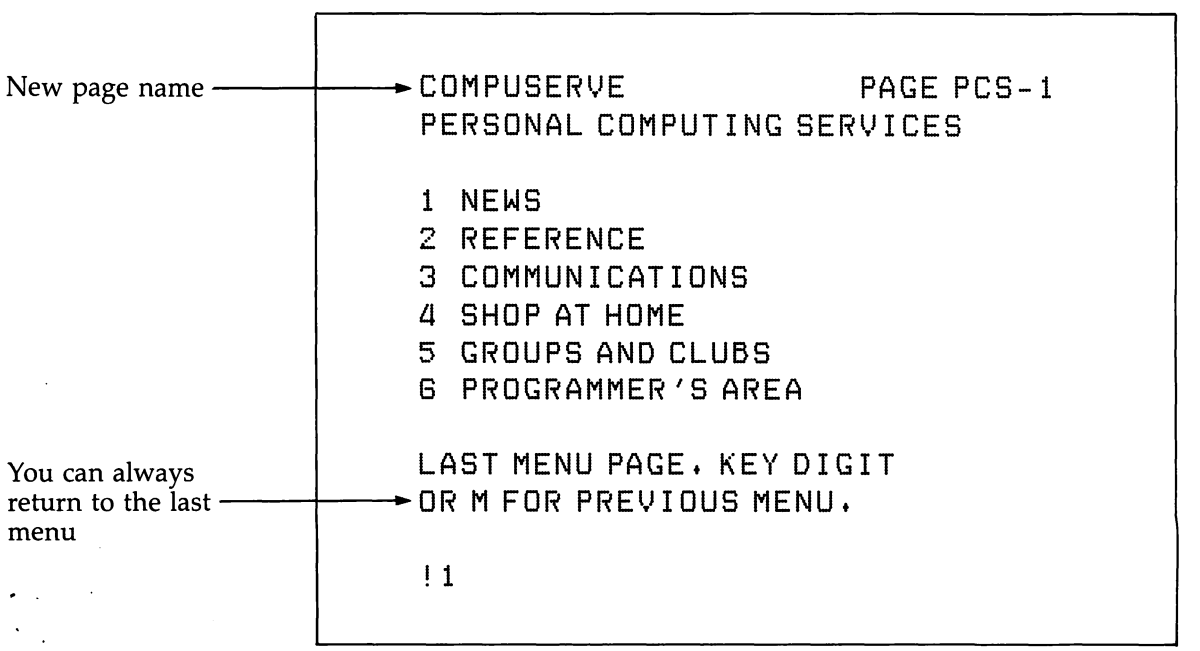


Figure 8–4: Series of (Annotated) CompuServe Screens (continued).

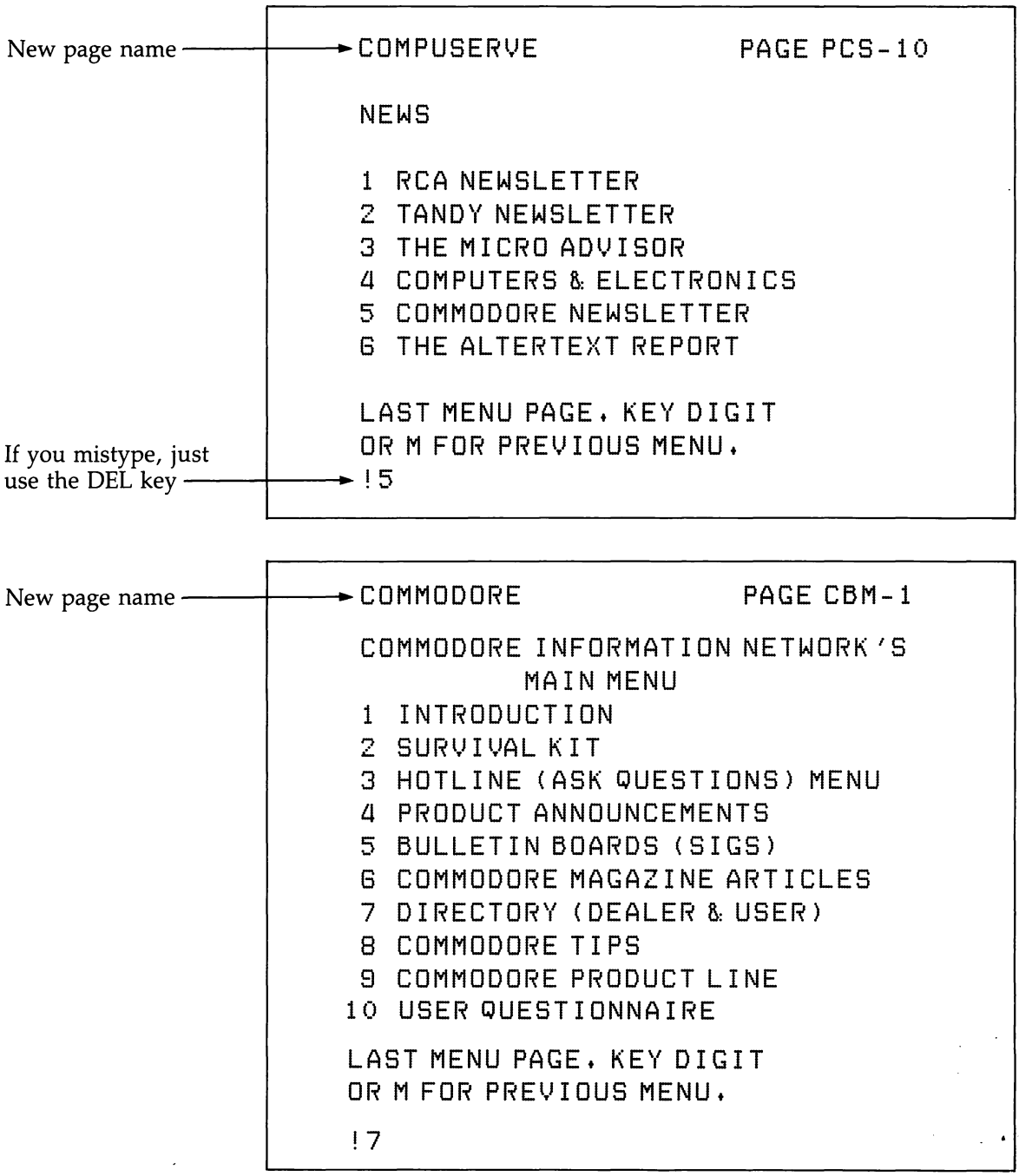


Figure 8–4: (continued)

New page name →

```
COMMODORE                PAGE CBM-10

COMMODORE  DIRECTORIES

1 COMMODORE DEALERS
2 EDUCATIONAL RESOURCES
3 USER GROUPS

LAST MENU PAGE. KEY DIGIT
OR M FOR PREVIOUS MENU.

!3
```

New page name →

```
COMMODORE                PAGE CBM-248

1 AL,AZ,AR,CA
2 CO,CT,FL,IL
3 IN,IA,KS,LA,MD
4 MA,MI,MN
5 MT,NV,NJ,NH,NM
6 NY,NC,OH,OK,OR
7 PA,PR,SD,TN,TX
8 UT,VA,WA,WI
9 CANADA,ENGLAND,MEXICO

LAST MENU PAGE. KEY DIGIT
OR M FOR PREVIOUS MENU.

!1
```

Figure 8–4: (continued)

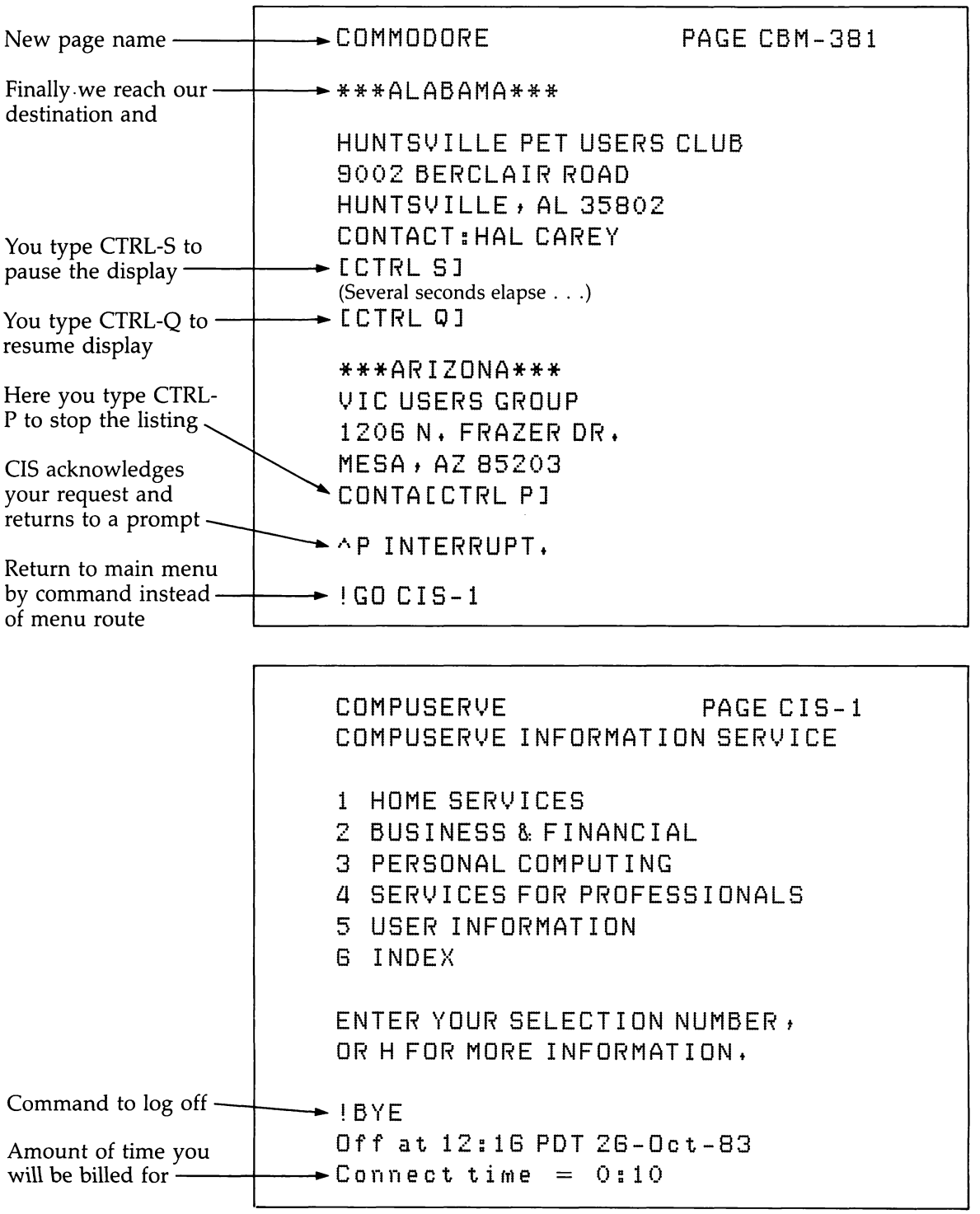


Figure 8–4: (concluded)

While menus are a splendid way for inexperienced users to navigate the system, it takes time for the host computer to redraw all the menus that occur between destinations. Your forty-column, twenty-five-line display allows 1,000 characters. At the VICMODEM's data rate of thirty characters per second, it *can take* as long as thirty-three seconds to print a full-screen menu. Thus, at $6 per hour, each full screen costs more than 5¢. That adds up quickly! The solution to this problem is to learn *exactly* where you want to go, then take the express route. With practice, you'll learn to swing freely among the branches, greatly decreasing your time in the tree. At the end of the session shown in Figure 8–4, we returned directly to the main menu merely by typing:

GO CIS-1

(All paging commands can be abbreviated, for example G CIS1.) *Every* screen on CompuServe has a unique page name consisting of a three-letter section name and a page number. The Commodore screen looks like the facsimile below:

```
                                        CBM-1

Section:
COMMODORE Business Machines
                                        Page 1
```

Begin building yourself a chart that shows the page name of the sections you visit frequently. The CompuServe manual contains such a chart, but the best and most current one is the on-line index beginning at IND-1. With the help of a few notes and your destination chart, you'll be amazed at how fast you can log on, send mail, and log off.

Terminal Software

Now that you have a feel for the modem and an idea of what to expect from an information utility, we need to examine the terminal software more closely. Look again at the list of three program requirements. Nowhere did we require typed characters to be displayed on the screen. As a matter of fact, this feature is seldom needed because most of the computers you will access

echo every character you send them. This means that as soon as the remote, or host, computer receives your character it immediately sends it back. The character you see on the screen is actually an echo of the one you transmitted. In other words, a character must make the round trip from your computer to the host and back to your computer before it appears on your screen. This may seem like a roundabout way to print a character, but it almost guarantees that you see exactly what the host computer is receiving. There are a few, mostly archaic, host computers that don't echo back characters. There are also times you would like to see the characters as you type them. So let's add one more requirement to our list for the modem program:

4. Key strokes may *optionally* be sent to your screen.

How does 64 TERM handle this requirement?

You type: SHIFT-F3

Result: the slightly intimidating screen in Figure 8–5:

```
                64 TERM

BAUD RATE 110 150 300
          1200 2400

DUPLEX      FULL/HALF   <-- Local display control

PARITY    NONE EVEN
            ODD MARK
            SPACE

STOP BITS ONE/TWO

WORD LEN 8 7 6 5 BIT

NEXT PAGE TERMINAL ON

EXIT BACK TO BASIC
```

Figure 8–5: 64 TERM Menu #1.

For now, looking at Figure 8–5, concentrate on the line indicated by the arrow. The term DUPLEX is intended to mean:

> HALF DUPLEX means display typed characters (local echo).
>
> FULL DUPLEX means do not display typed characters (remote echo).

64 TERM is automatically set to FULL DUPLEX. To change this setting:

> *You type:* D
> *Computer:* DUPLEX (highlighted)
> *You type:* ←CRSR→
> *Computer:* (highlighting moves to HALF)
> *You type:* <RETURN>

Then the original highlighting returns to the word DUPLEX. Finally, as indicated by the next-to-last menu entry, press T to return to the terminal mode.

> More descriptive terms than HALF and FULL DUPLEX would have been LOCAL and REMOTE ECHO. Actually, HALF and FULL refer to whether communication is taking place in both directions (full duplex), or only in one (half duplex). Older modems frequently couldn't send and receive at the same time. A telephone is a familiar full duplex device, and a CB radio is half duplex. Microcomputer modems are *never* really half duplex.

Type a few characters and notice that they are now displayed on your screen. If you were to phone an "echoing" host computer with your computer set for HALF DUPLEX, you would see your characters *ttwwiiccee*, once for your key stroke *and* once for the host's echo. Of course, this HALF DUPLEX setting will legitimately be needed if you stumble across a nonechoing relic.

The remainder of this 64 TERM menu falls under item #3 on our list, the ability to change *transmission formats*. Before a character can be transmitted by phone, it is reformatted by breaking it into its constituent elements (*bits*). These elements are transmitted individually, then reassembled by the receiver into a whole character. In order to communicate, two computers must agree upon the formatting characteristics: WORD LEN(gth) (the number

of elements in each character), STOP BITS (the length of the pause between characters), and PARITY (whether an extra bit will be added to the character as an error code).

Novices in telecommunications fret unnecessarily over these settings. As a practical matter, the following setting will usually work fine: **ONE** stop bit, **NO** parity, **8** bits-per-word. All major information services and bulletin boards will accept this universal configuration, even if they officially recommend a different one.

Occasionally, you may encounter a host system that, for no reason, ignores your attempts to log on, or unceremoniously hangs up the phone after you send it a few characters. If this happens, you have probably reached a system that checks the parity error code on each incoming character. Since the universal setting (ONE/NO/8) defeats this feature, the host system will interpret about half of your characters as errors. Such systems are programmed to reject callers whose characters exhibit a high error content. In such cases, ONE/EVEN/7 or ONE/ODD/7 will probably clear up the problem.

The remaining format characteristic on our menu, *baud rate,* is more important. Baud rate is the speed at which your modem transmits the individual elements of a formatted character. Unfortunately, the VICMODEM can operate only at the transmission rate of 300 baud. 1,200 baud modems are available from other companies, but their prices start at about $500. Installation of these modems also requires an RS-232-C interface adapter and cable. Consult your dealer about this possibility. Modems that operate at 1,200 baud can save you money in connect-time, because you're receiving information many times faster.

If you phone a modem that is set at a different baud rate, you will get gibberish on your screen. Most dial-up computers provide separate telephone numbers for different baud rates, and they're almost certain to have a 300-baud number.

As long as we're on the subject, let's take a few moments to finish our discussion of the remaining options in 64 TERM. Invoke the menus by typing F4. Now type N to bring up the other menu. Your screen should look like Figure 8–6 on page 248.

Here is a brief explanation of the items on menu #2.

LINEFEED/CARriage RETURN: Have you ever noticed how a manual typewriter creates a new line? There are actually *two* actions

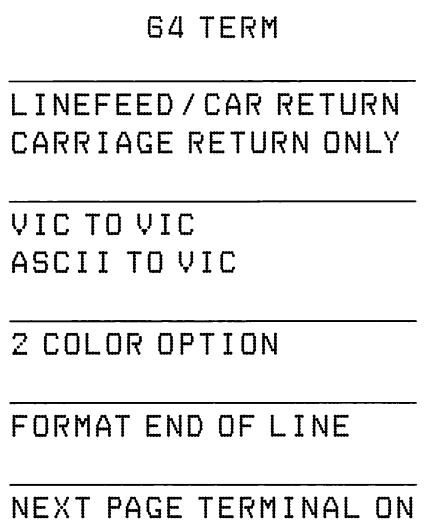


Figure 8–6: 64 TERM Menu #2.

combined into one motion: the lever not only returns the carriage but also advances the paper one or more lines by means of a built-in ratchet. The C-64 works the same way: when you press the RETURN key, the operating system actually sends *two* characters to the screen. First it sends the cursor to the far left of the display, followed by a *line feed* character to drop the cursor down to the next line. But since the modem is under the control of 64 TERM directly (not the operating system), the linefeed is not automatically added when a carriage return is sent. Unfortunately, many host computers require both characters; that is, they will not create a blank line with *only* a carriage return. Can you visualize how this would look? All the lines would print over each other!

The first selection on the menu automatically adds a linefeed after every carriage return. This and the remaining settings can be changed using the CRSR key and the RETURN key, as in the case of the Duplex option.

> Commodore's use of a single carriage return to represent a carriage return/linefeed pair is nonstandard, though by no means unprecedented, so many host computers have a section entitled Changing Terminal Defaults (on

CompuServe, *type:* GO CIS-6 <RETURN>) that allows you to adjust the host to your needs. Here, for example, you can instruct service to display both upper-case and lower-case letters.

ASCII TO VIC: As you may know, computers manipulate characters by first converting them to numbers. Most computers use the numbers recommended in the American Standard Code for Information Exchange, or ASCII. Commodore devised its own codes, called PET ASCII. The C-64 must therefore translate incoming and outgoing characters in order to communicate with non-Commodore computers. The VIC TO VIC option should be selected only when talking to other Commodore computers.

COLOR OPTION: If this option is selected, characters arriving by modem are displayed in a different color from typed characters.

FORMAT END OF LINE: When you have typed to the end of a line, the cursor automatically *wraps* around to the beginning of the line. Unfortunately, this breaks words unnaturally and makes the screen difficult to read. If a word does not fit on the current line, The FORMAT END OF LINE option will begin a new line. Figure 8–7 illustrates its effect.

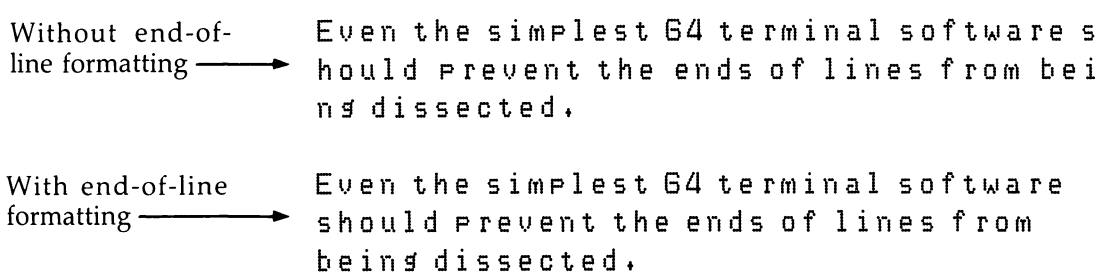


Figure 8–7: End-of-Line Formatting.

The Networks

When we called CompuServe, we used their private telephone network. The use of private networks is preferred because there is usually no extra charge for them. But if your town doesn't have

a ComLink phone number or if you wish to call another service, you will have to use one of the national *packet-switching* networks, Tymnet and Telenet. These networks maintain telephone numbers in hundreds of cities around the United States. When you dial up through one of these networks, a communications surcharge will be added to your bill. Average cost is $2 per hour at 300 baud. Some services include network charges as part of the basic subscription price.

When you dial up one of the networks, you will need to know two things: the *terminal type* for the C-64 and *host ID*. The C-64's terminal type is D1 for Telenet, A for Tymnet. The host ID is the network's code for the service you are calling. The procedure is not the same for both networks: Figures 8–8 and 8–9 illustrate the differences.

Whenever you subscribe to any computerized service, be certain to ask the subscription department for both network numbers. Even if the service offers a private network of its own (like CompuServe), you can use Telenet or Tymnet as a backup in case the private network malfunctions.

> For the network phone number nearest you, call: Telenet — 800-336-0437 (800-572-0408 in Virginia); Tymnet — 800-336-0149 in the East, 800-323-7389 in the West. A human will answer at these numbers: do *not* dial them with your modem!

Intelligent Terminal Software: Downloading and Uploading

In figure 8–4, our movement through the CompuServe system led to a listing of Commodore bulletin boards. Wouldn't it be useful if, while characters from this display were going to the screen, they were also entering the C-64's memory? Once in memory, they can be easily printed or stored permanently on tape or disk. Collecting incoming characters into a reserved memory space — known as a *capture buffer* — is called *downloading*.

Suppose that you wish to post the same notice on five different bulletin boards. Using 64 TERM, you'll have to type in the same text five times. Wouldn't it be more efficient to prepare

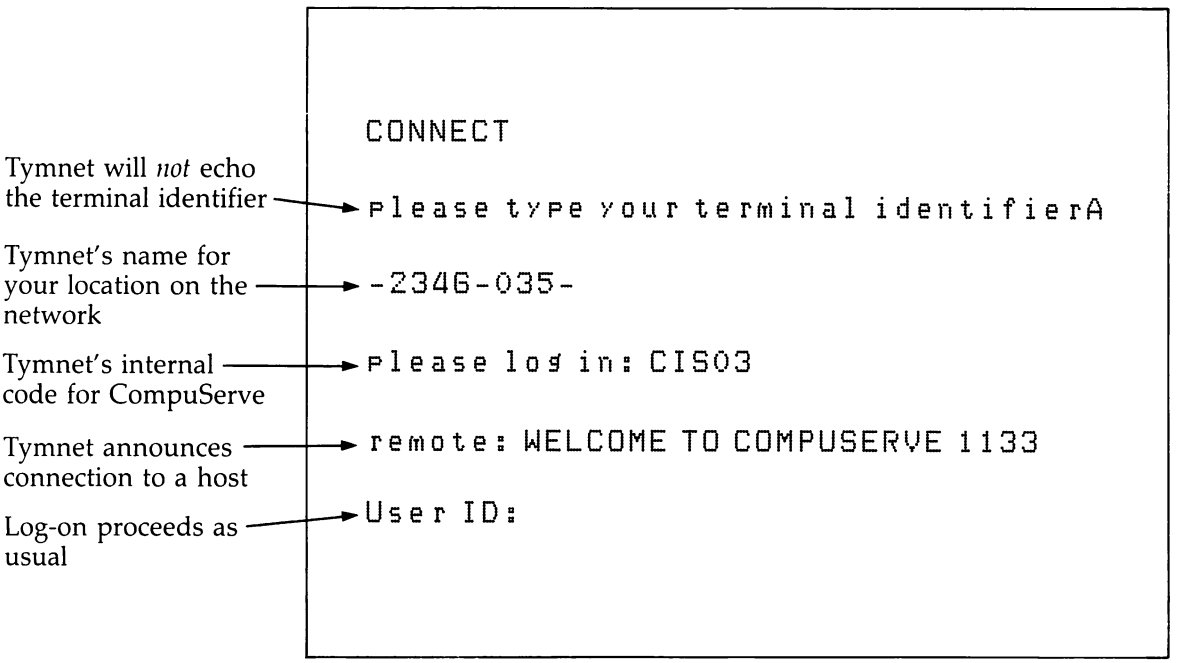


Figure 8–8: Access Through Tymnet.

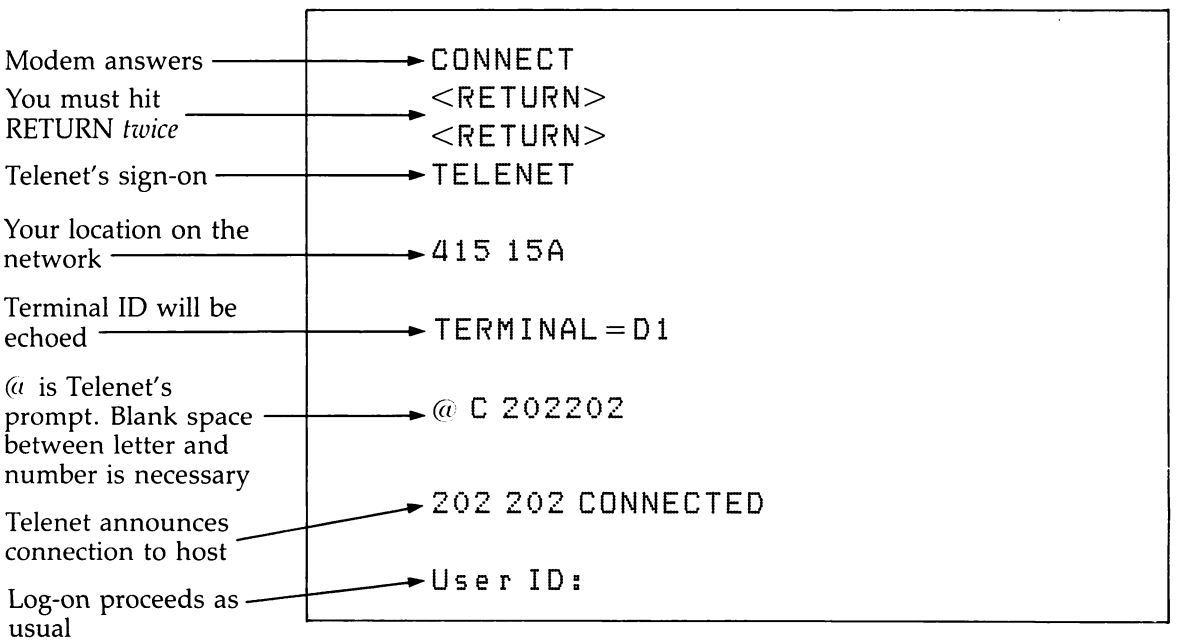


Figure 8–9: Access Through Telenet.

your notice one time with EASY SCRIPT, then *upload* it five times to the various bulletin boards?

These are some of the shortcomings of a simple terminal program like 64 TERM. They waste much of the power available in a microcomputer: tape drives, disk, and printers all sit idle while interesting and valuable information appears briefly on the screen. That's why these simple programs are called *dumb* terminal programs while other, more sophisticated programs are labeled *intelligent*, or *smart*.

> An alternative to the VICMODEM/64 TERM should be noted: The HesModem1 (Human Engineered Software, 71 Park Lane, Brisbane, CA 94005) includes 64-T, a semi-intelligent program, with the modem. This pair is functionally equivalent to the Commodore package except that 64-T can capture incoming characters into memory where they can be reviewed and printed. The capacity of the capture buffer is 20,000 characters, and the capture action can be turned on and off in order to bypass unwanted material. Nevertheless, since it doesn't have the ability to store the characters permanently, 64-T is only a slight improvement over 64 TERM. It costs about $70.
>
> 64-T is also available for about $30 (without a modem), directly from its creators, Midwest Micro, Inc. (311 West 72nd St., Kansas City, MO 64114). Although we have chosen to use SMART 64 TERMINAL PLUS as our example of an intelligent terminal program, we should point out that Midwest Micro's deluxe SuperTerm (about $100) deserves a close look, especially if you intend to use the modem for business.

SMART 64 TERMINAL PLUS (from Microtechnic Solutions, Inc., New Haven, CT 06515) is a full-featured intelligent terminal package. At $39.95, it is an exceptional value. No provisions are made for tape operations, however. SMART 64 is sold on a copy-protected disk, so don't try to make a duplicate of the disk with 1541 BACKUP (see Chapter 2). If the disk gets damaged, contact the manufacturer for a replacement.

In addition to all the basic features of 64 TERM, SMART 64 has more advanced features:

1. *Programmable function keys*: You can assign up to eighty characters to be transmitted when you press a function key. F1 is reserved for user ID, and F2 for passwords. Using the function keys cuts down the time required to type the information.

2. *Transmit* (*upload*): any disk file.
3. *Capture* (*download*): incoming text to memory and, optionally, write it to disk under a file name of your choice.
4. *Disk Wedge*: commands are available without leaving the program. Most commands are available, including SCRATCH, RENAME, COPY, VALIDATE, NEW, DIRECTORY, and LIST.
5. *Editing of memory buffer*: allows you to break the memory buffer into separate parts and write them to separate disk files.
6. *Transmission of both program and BASIC text files.*

Setting Up SMART 64

The SMART 64 disk itself is not in your disk drive during operation. Instead, you'll use SMART 64 to create a *user disk* containing such information as modem transmission formats, user numbers, password, the preselected name for the download file, screen colors, and custom printer file. These user disks are created by running a special program called SMART64.BUILD, which asks you for the necessary information.

The prompts request information in a relatively straightforward fashion. It is advisable, however, to finish reading this chapter before you actually begin building the user disk. Then you will be better able to understand *why* each piece of information is necessary. In most cases, though, the preselected (or default) values will work acceptably. Thus, you can answer most prompts with <RETURN>, except for the "Format Lines" prompt explained below.

A user disk must be prepared for *each* database. For example, one user disk is reserved for dialing CompuServe; this disk would contain your CompuServe user ID and password, and perhaps the function keys to transmit often used commands. Similarly, you'll want a disk for The Source, one for Dow Jones, and so on for other on-line data bases.

When you're ready to call a database, boot up SMART 64 itself. After about forty-five seconds of a sign-on message, you are instructed to insert your user disk. Remove SMART 64 and return it to its protective sleeve immediately. The program loads

your user ID, password, and other data from the user disk, and seconds later you'll see the main menu.

Items 9 through 13 in Figure 8–10 allow you to change the content of the user disk without having to exit the program and run the user disk modification program SMART64.CHANGE. Item 14 invokes the disk Wedge (see Chapter 2), and item 15 exits the program and returns you to BASIC. Item 3 allows you to print the contents of a sequential disk file. Items 5 through 8 are discussed briefly at the end of this chapter.

Entering Terminal Mode

Selection 1, GO ON-LINE doesn't, of course, put you on line unless you have already established a modem data link with a host computer. It simply puts you into the terminal mode. Whenever you elect to enter the terminal, the program assumes that you will want to capture data from the host. In preparation for this, you will have to answer a few questions about how you wish to download to proceed. In every case, the preselected values are displayed in parentheses.

```
Download File (COMM.DATA): --
```

If you hit RETURN, the file will be given the name shown in parentheses. If the file already exits from a previous download, or if your disk has no space remaining, you'll be given the opportunity to erase the file using the Wedge: *Want Wedge?*

```
0    bozo 11
511 blocks free
```

These are the basic file statistics of the disk you are using. In this example, there is only one file, bozo, occupying 11 disk blocks; 511 blocks remain.

```
Disk Blocks for Download (325)
```

If you wish to restrict the size of the download file, enter the maximum size you will allow.

```
Keying Limit ( 80) :
```

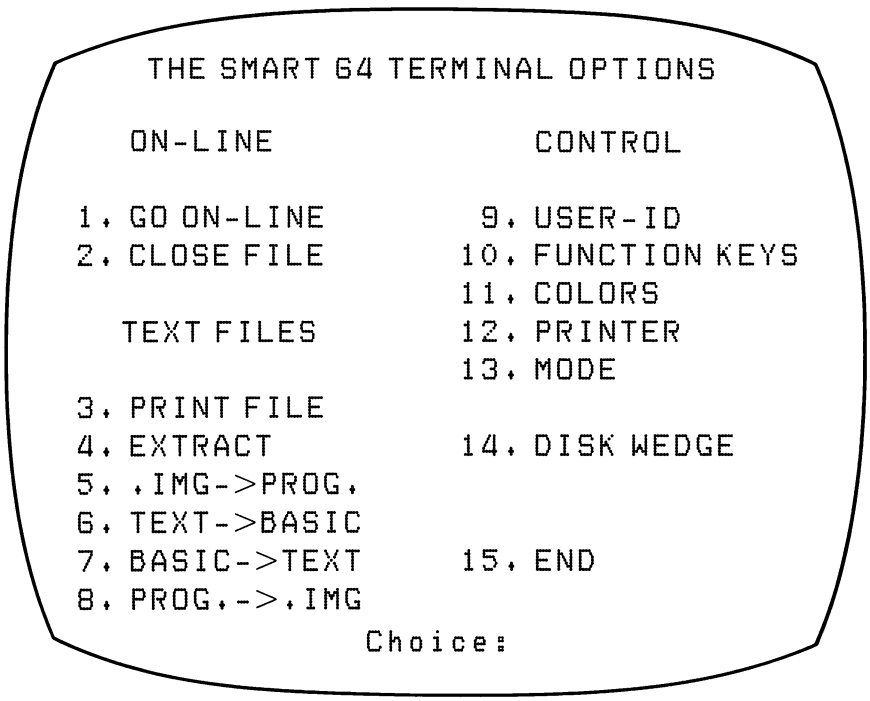


Figure 8–10: SMART 64's Main Menu.

Some host computers must be sent a <RETURN> after a certain number of characters. SMART 64 will automatically send a <RETURN> after the number of characters you specify in this option.

```
Clear Buffer option? (n)
```

This allows you to determine how SMART 64 will react when memory is full. If you choose the default value (n), the memory will be written to disk under the name you chose above. If you answer <RETURN>, buffer memory will automatically be cleared and begin filling up again.

```
Format Lines? (n)
```

Unlike its counterpart in 64 TERM, SMART 64 breaks a line in the middle of a word unless you tell it not to. Thus, you must press y in response to this prompt.

```
Send Linefeeds? (n)
```

Answer (y) if your host requires a linefeed character after each RETURN. This will be the exception.

```
Continuous Uploads? (y)
```

Always enter RETURN. (Contact Microtechnic Solutions, Inc., for an explanation of the circumstances in which noncontinuous upload would be required.)

Now that SMART 64 knows how you want to handle the details of the download, it transfers you into the terminal mode proper. You can now begin communication with your host.

Downloading Files

When first entering the terminal mode, you are *not* yet capturing incoming character in memory. You must first press RETURN. This causes an icon (it looks like a plumb bob) to appear at the right side of the screen. Hit C= RETURN again and the icon disappears. By thus "toggling" the capturing action on and off, you may bypass unwanted material (menus, for example). Remember, *you are not capturing data unless the icon is visible.* As memory fills, the icon will slowly sink. After memory has filled (unless you chose the Clear Buffer option above), SMART 64 will automatically write it to disk. Since disk operations can take several seconds, they are fully *supervised* — that is, when the disk write begins, SMART 64 sends an X-OFF to tell the host to pause. After the write is complete and SMART 64 is ready to receive more characters, it automatically sends an X-ON to restart the host.

The size of downloaded files is limited only by the free space on your user disk. By using the Disk Blocks for Download option above, you can restrict the size of the file.

Aside from the presence of the icon and an occasional pause while captured characters are written to disk, communication with the host proceeds essentially as in 64 TERM.

Exiting Terminal Mode

You may wish to exit the terminal mode momentarily. F8 will return you to the main menu. When you return to the terminal

mode, you will not have to answer the questions listed above, because the download file characteristics you named on your original entry are still active. Upon return to the terminal mode, you resume communication, including downloading, where you left off. Suppose you have finished a downloading session and now wish to close the file. Return to the main menu by typing F8, then select item 2, CLOSE FILE. The disk will whirr and a CLOSING FILE message will appear. Your captured characters are now safely stored in a disk file, the memory has been cleared, and SMART 64 is ready to capture again. When you enter terminal mode now — after closing a download file — you will have to answer the questions outlined in the previous section.

SMART 64's Important Keys in Terminal Mode

F1	Sends user ID
F2	Sends password
F3	Uploads selected file
F4	Each key will send 80 characters of your choosing
F5	
F6	
F7	
F8	Return to main menu
C=<RETURN>	Turns download screen icon on and off
C=-£	Sends X-OFF to host, then sends the contents of screen to printer
C=W	Invokes disk Wedge

Uploading Files

Sending files to the host is just as simple as receiving them. Typing F3 results (as always) in several questions:

```
Filename:
```

Enter the name of the file you wish to transmit to the host. If you name a file that doesn't exist, you'll be offered the Wedge.

```
PET ASCII? (y)
```

As previously explained, Commodore computers use a nonstandard character representation code. Answer no only if you are certain the characters stored in the file are in standard ASCII.

```
#Nulls after C/R (0)
```

Nulls are do-nothing characters used to waste time while a printer's head performs a carriage return. Some archaic computers may also require a few nulls. If a host shows loss of characters at the beginning of lines, start with twenty-five nulls, but the exact number must be ascertained experimentally.

When you press RETURN after the final question is asked, the disk drive starts working and (provided your file exists on the installed disk) you will see the text on your screen. The characters that print on your screen are always the host's echo, so if you are sending to a mute host, your screen will not show the outgoing text. (In this case, however, you would doubtless have prepared a custom user disk with the HALF DUPLEX option selected.)

A Sample Upload-Download Session

This section presents an actual correspondence between the author of this chapter and the software manufacturer via "electronic mail." The SMART 64 manual invites users "to contact Microtechnic Solutions via CompuServe, address EMAIL to 74145,1015." The reason for contacting the company is to find out more information about the software. The questions were prepared with a word processor, and saved in a file called MTS.LET. The screen facsimiles shown in Figure 8–11 reconstruct how the correspondence looked on the author's C-64 monitor.

That's all there is to it. Notice the total time on line was only three minutes. By composing the letter off line, we saved about twenty minutes in connect time.

The next morning, the answer was in the mailbox. Figure 8–12 on pages 263–265 shows that session (the log-on procedure and the main menu have been omitted).

Once back to the main menu, item 2 — Close File — is selected and the file is closed. The company's reply letter is now on disk, where it will be printed and added to the SMART 64 user manual.

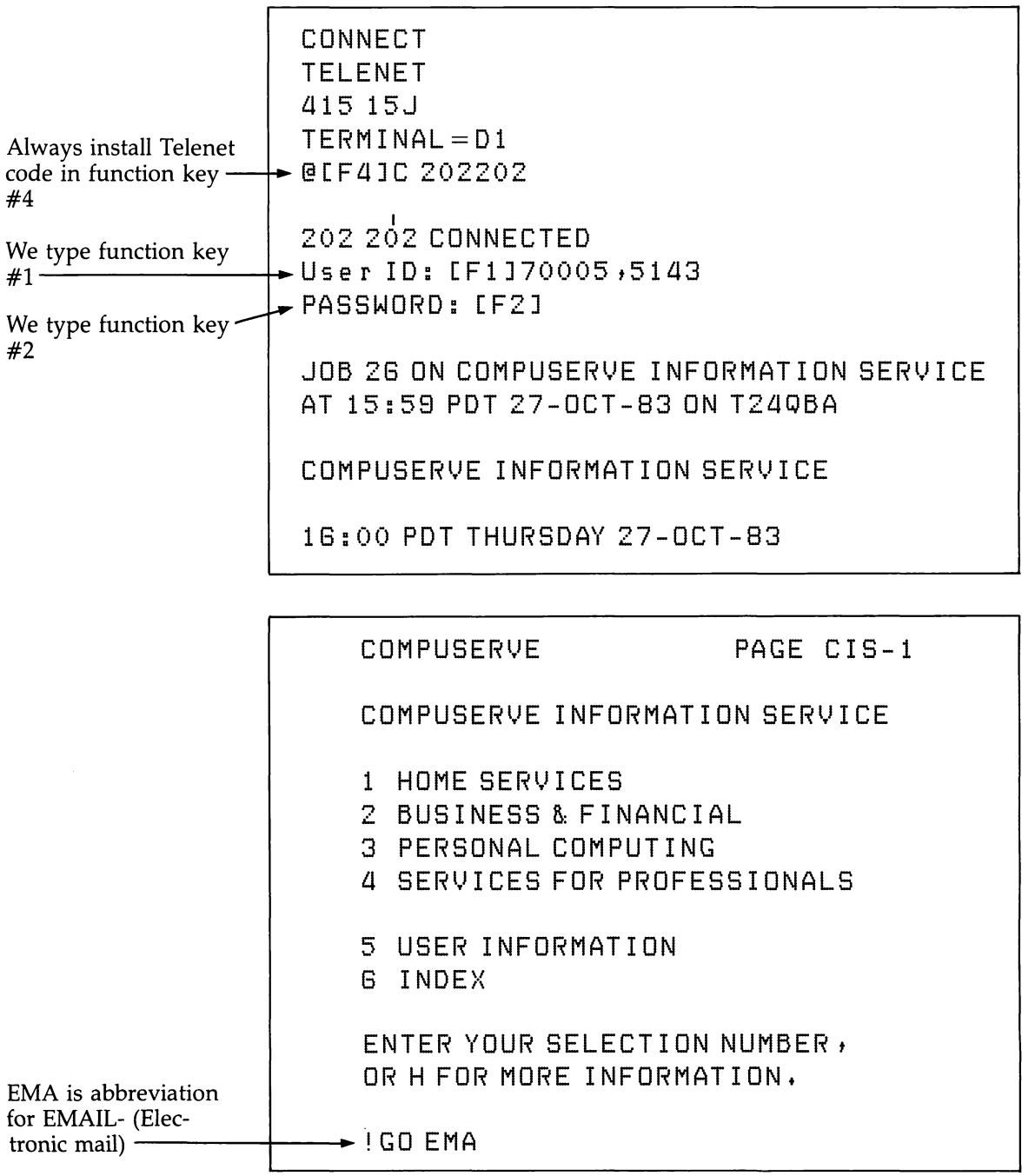


Figure 8–11: Uploading Electronic Mail (continued).

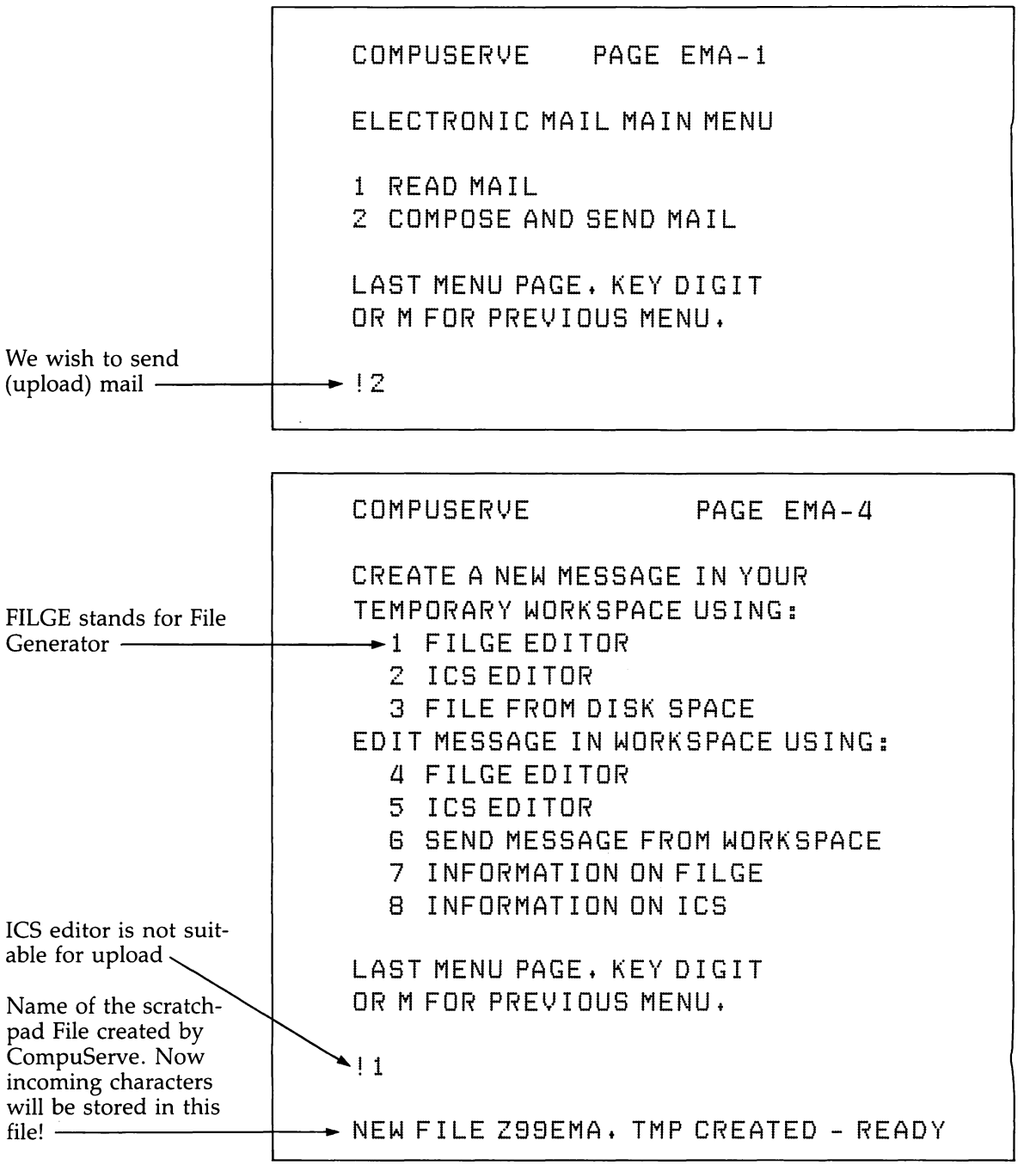


Figure 8–11: (continued)

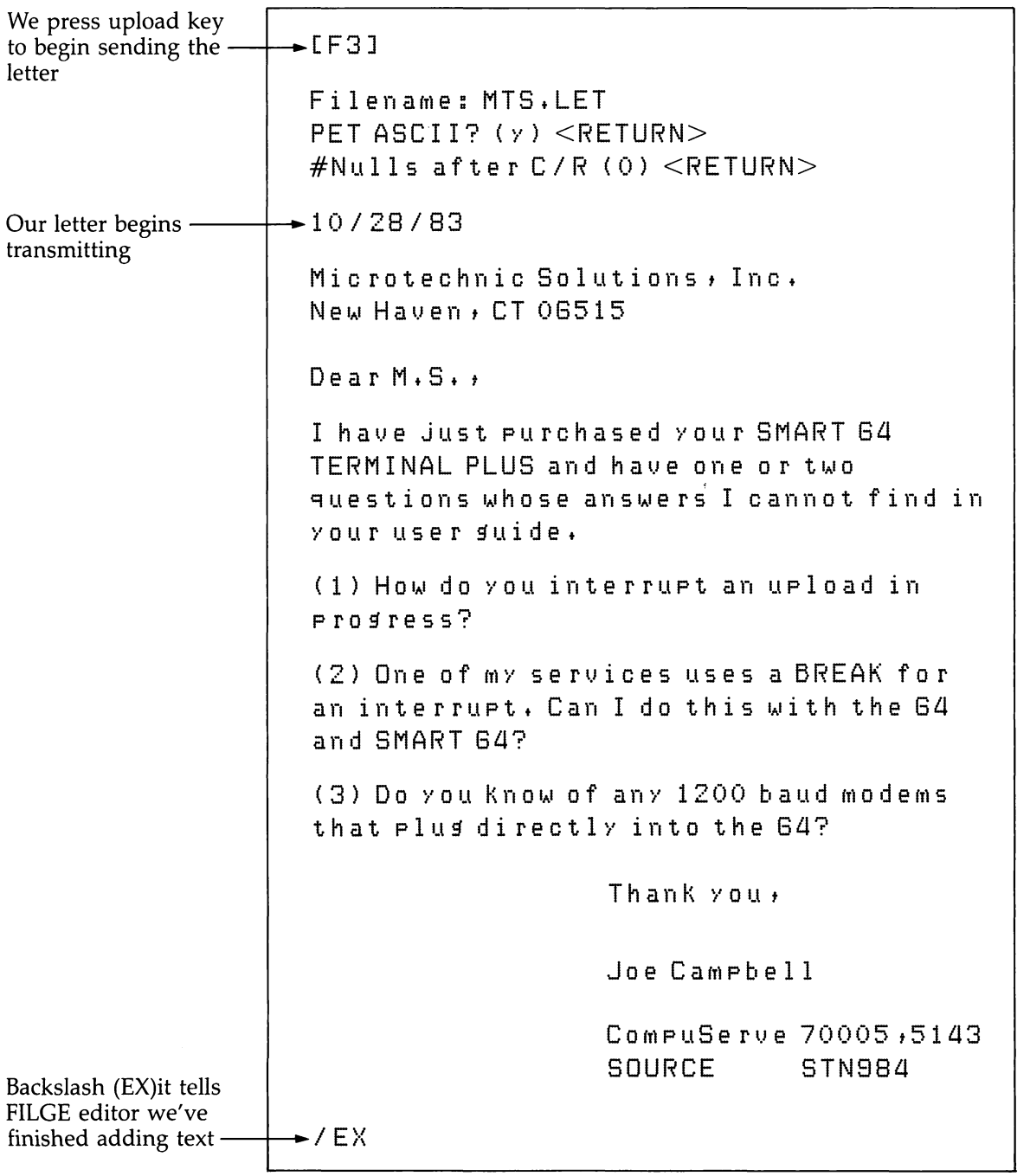


Figure 8–11: (continued)

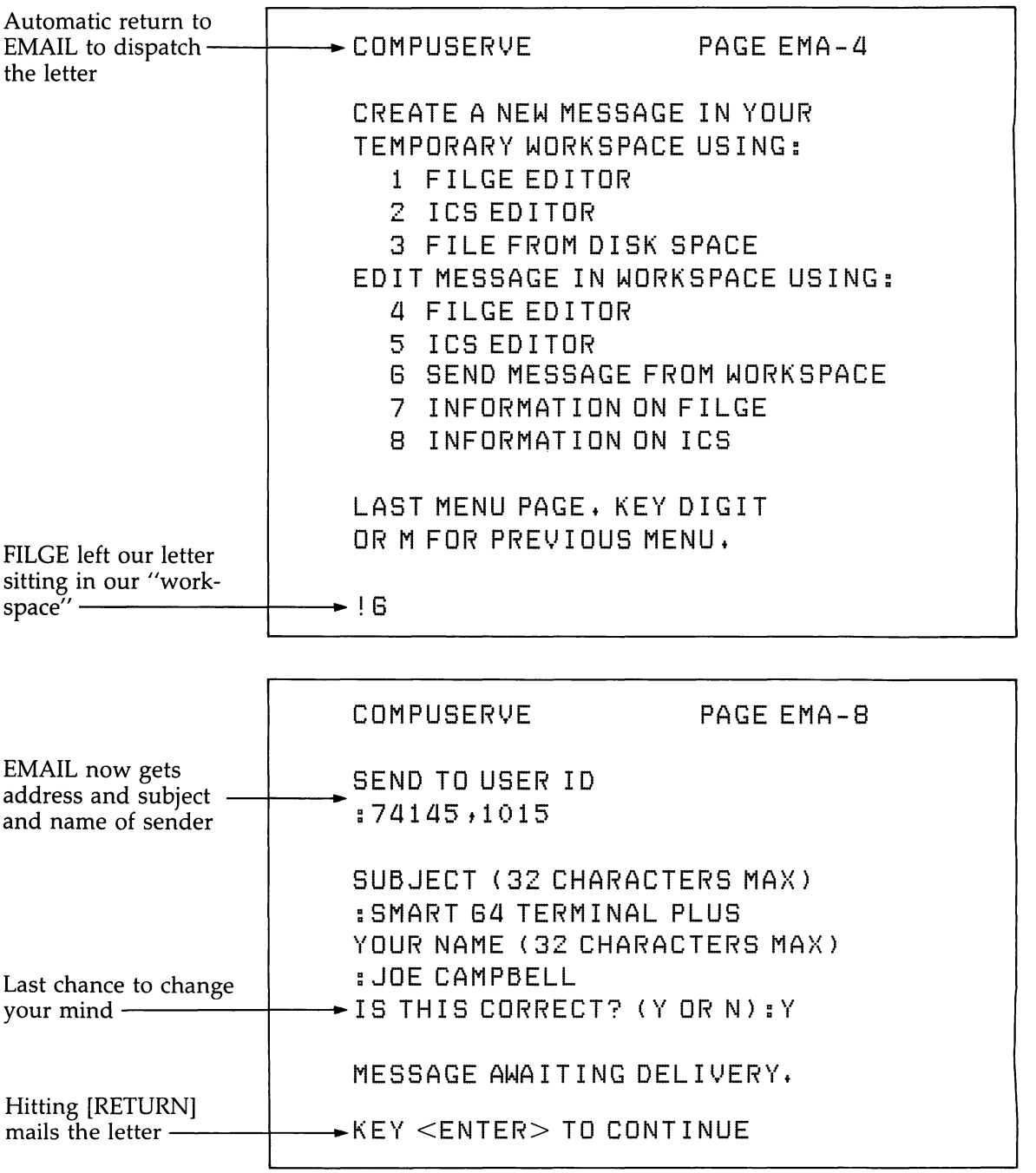


Figure 8–11: (continued)

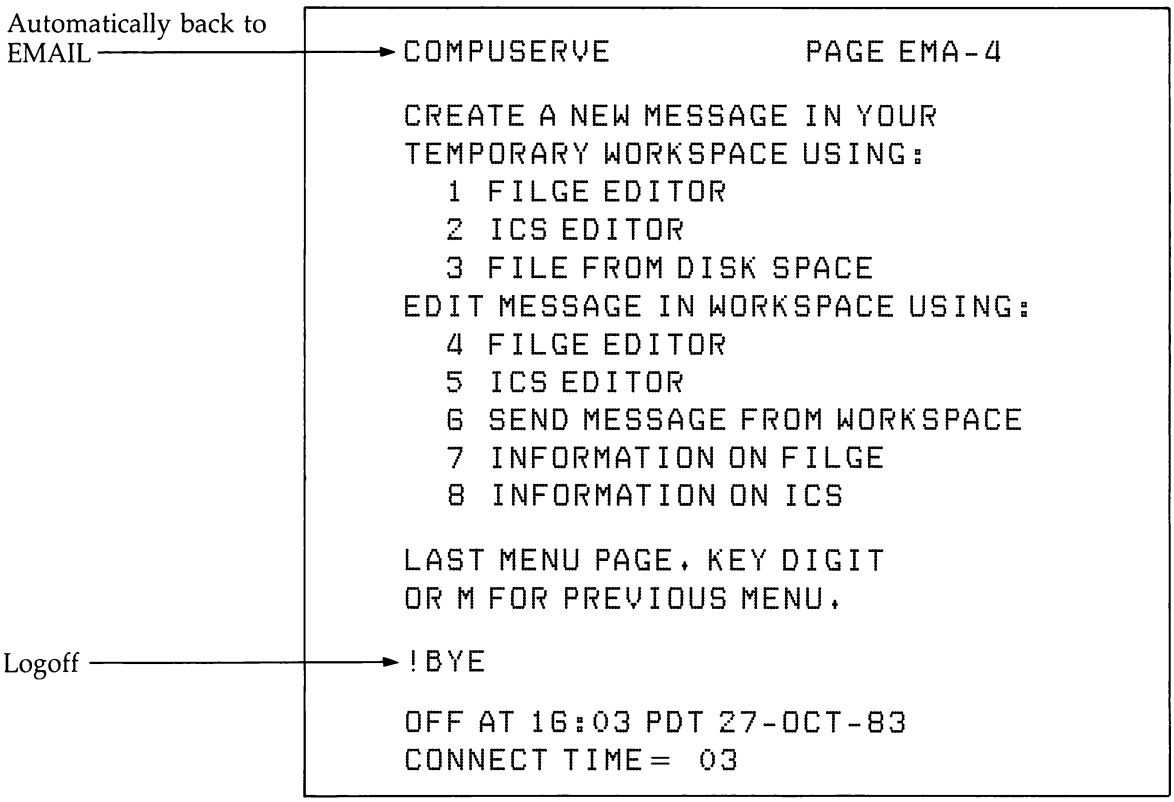


Figure 8–11: (concluded)

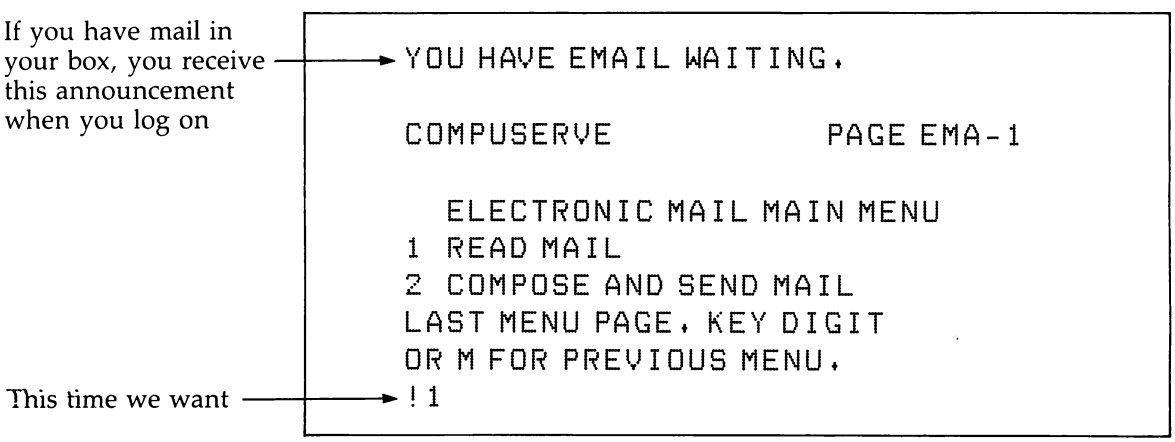
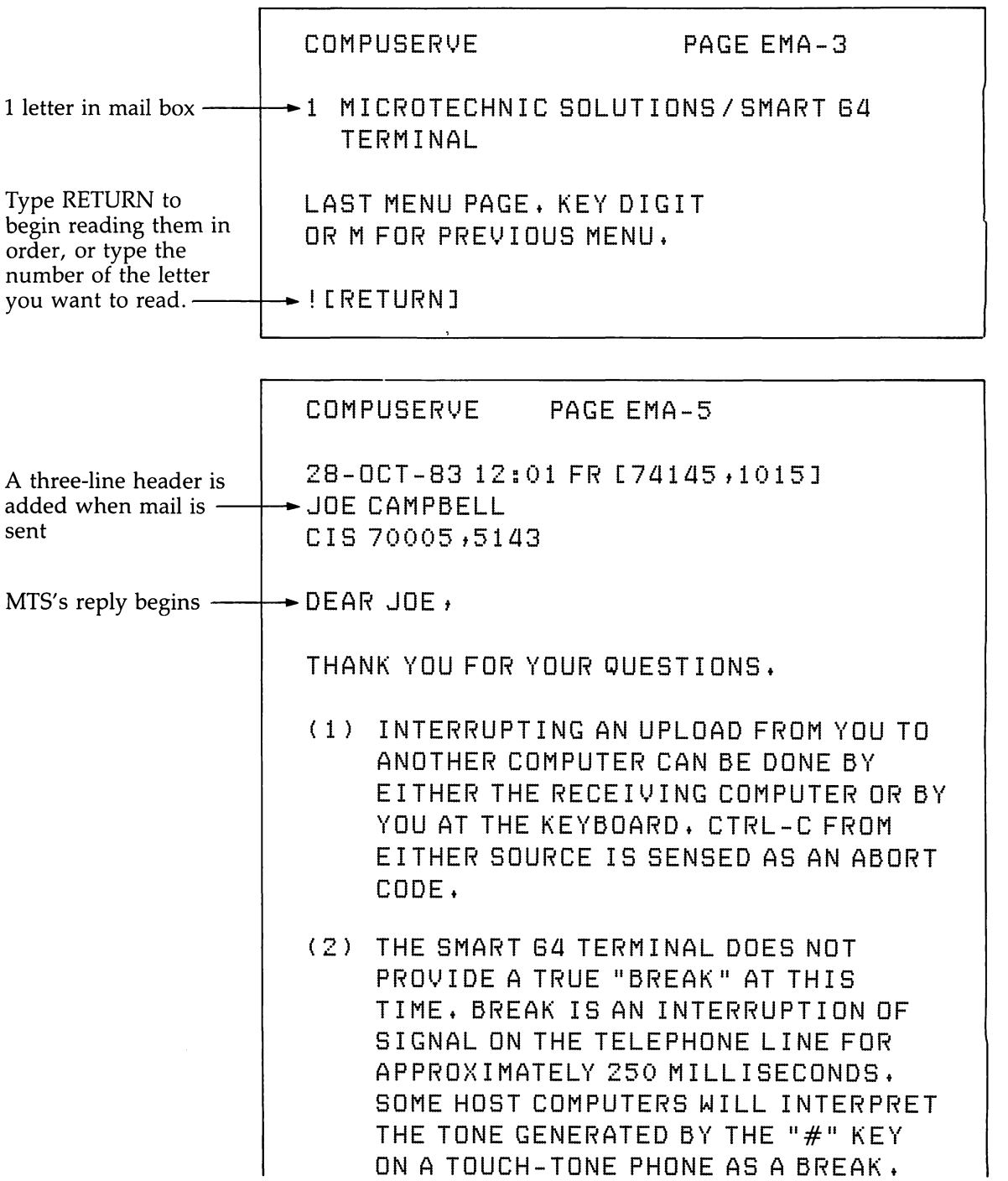


Figure 8–12: Downloading Electronic Mail (continued).


COMPUSERVE PAGE EMA-3
1 letter in mail box
1 MICROTECHNIC SOLUTIONS/SMART 64 TERMINAL
LAST MENU PAGE, KEY DIGIT
OR M FOR PREVIOUS MENU.
Type RETURN to begin reading them in order, or type the number of the letter you want to read.
![RETURN]
COMPUSERVE PAGE EMA-5
28-OCT-83 12:01 FR [74145,1015]
A three-line header is added when mail is sent
JOE CAMPBELL
CIS 70005,5143
MTS's reply begins
DEAR JOE,
THANK YOU FOR YOUR QUESTIONS.
(1) INTERRUPTING AN UPLOAD FROM YOU TO ANOTHER COMPUTER CAN BE DONE BY EITHER THE RECEIVING COMPUTER OR BY YOU AT THE KEYBOARD. CTRL-C FROM EITHER SOURCE IS SENSED AS AN ABORT CODE.
(2) THE SMART 64 TERMINAL DOES NOT PROVIDE A TRUE "BREAK" AT THIS TIME. BREAK IS AN INTERRUPTION OF SIGNAL ON THE TELEPHONE LINE FOR APPROXIMATELY 250 MILLISECONDS. SOME HOST COMPUTERS WILL INTERPRET THE TONE GENERATED BY THE "#" KEY ON A TOUCH-TONE PHONE AS A BREAK.

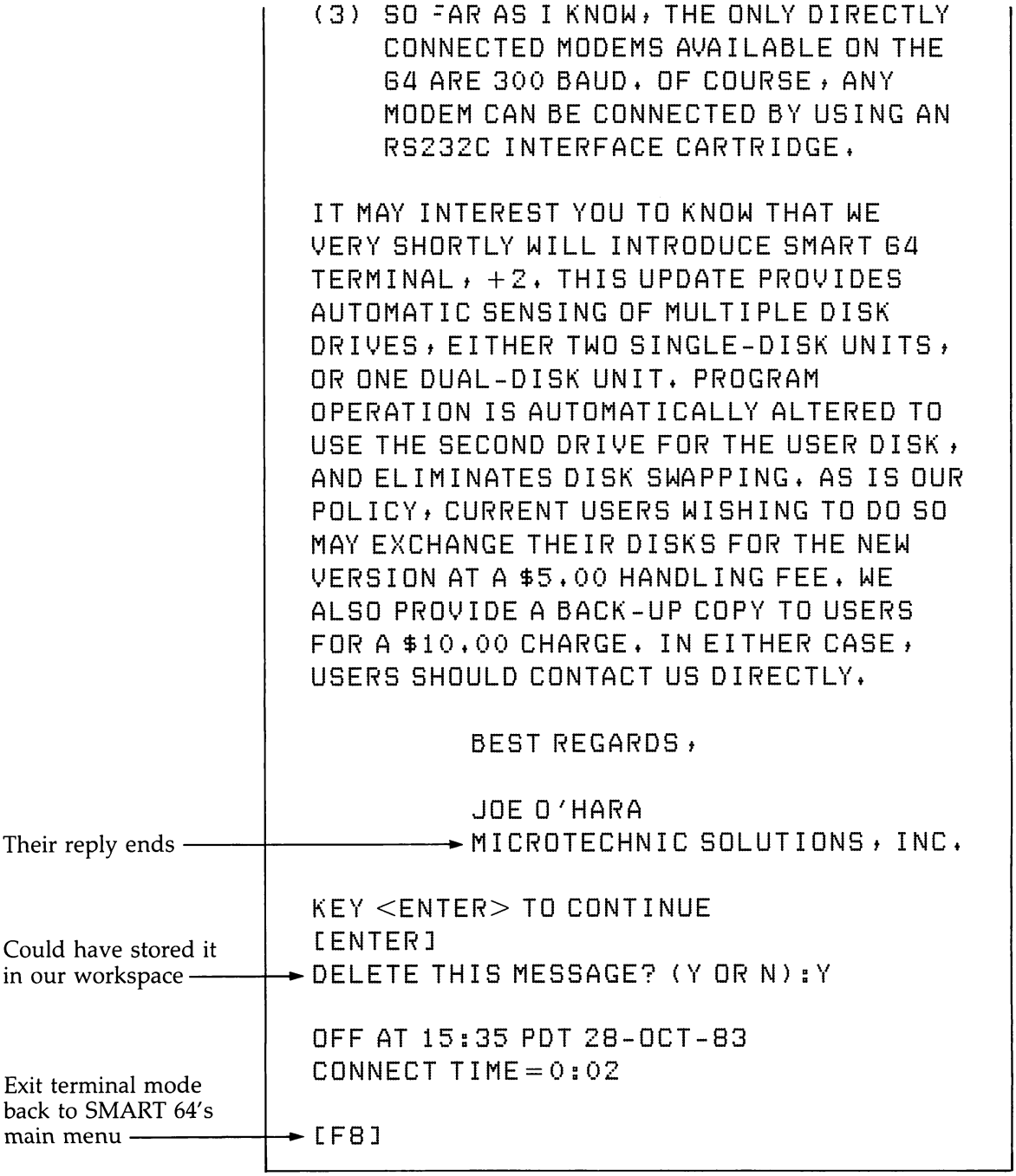


Figure 8–12: (concluded)

Transferring Programs

Most services like CompuServe and The Source as well as the computerized remote bulletin board services (RBBS's) maintain a rich library of software. Because of the data transmission formats of the host systems, however, it is not possible to transmit machine language programs directly. Instead, the program instructions are converted to text so that they can be transmitted like any other character file. (Commodore calls these ersatz files *image* files.) Once safely downloaded, these character files can be reconverted to their original form. SMART 64 TERMINAL PLUS supplies the utility programs to bi-directionally convert both BASIC and machine language files to text files.

Bulletin Boards

After your experience with CompuServe or The Source, you will have no difficulty navigating the simpler, more homespun bulletin boards. These systems are usually hobbyist endeavors. That they are labors of love is reflected in their friendly personalities. Here you will find message and gossip services, as well as the latest rage in games and public domain software. Indeed, bulletin boards are a treasure trove of software riches, offering semiprofessional programs as *shareware* — software for sharing.

Most boards offer some sort of terminal program. The main drawback to the software acquired from bulletin boards is the absence of users' manuals.

As you would imagine, the bulletin board population is unstable, so a current list of them is difficult to maintain. As you have seen, CompuServe's Commodore section maintains a list of Commodore boards. (For a partial list, *see* Appendix C.) The Source's Public section contains a huge list of bulletin boards of all types, listed by area of interest, hours of operation, and other pertinent operations. This brings up an often misunderstood point: don't be afraid to dial up non-Commodore bulletin boards. Most of the boards use very simple software and your C-64 will communicate fine as long as you set your terminal software to receive standard ASCII instead of PET ASCII. Of course, you will not be able to use the free software on non-Commodore boards, but a multitude of other, equally interesting things can be found there.

Troubleshooting

When using a modem, there's not very much that can go wrong, but here are a few problems you might encounter:

1. *You hear the tone from the other modem, but your CONNECT indicator doesn't illuminate.* This is usually caused by both parties having their ANSWER/ORIGINATE switches in the same position. One party must change the switch position.

   If this problem occurs only with one number, you are probably calling a defective modem whose tones are so far out of adjustment that your modem cannot recognize them.

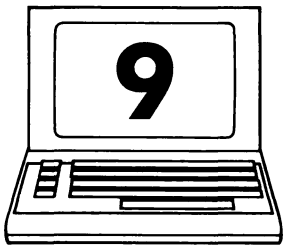
2. *During communications, spurious characters appear on the screen.* There are many possible causes for this phenomenon, but a noisy telephone connection is the most common. If the problem persists, hang up and redial.

   Another cause for the reception of spurious characters is easily overlooked. Modern telephone systems have several convenience features that actually inject noise on your line. The largest culprit is the call waiting/call interruption service that actually "hiccups" the line in order to signal another incoming call. This is guaranteed to cause random characters to be generated.

   If about half of your characters are incorrect, this is a sure sign that your terminal program is set to the wrong parity. Refer to the section above on setting the parity bit.

   A continuous stream of identical characters is usually the result of a mismatch in baud rates — you are probably connected to a 1,200-baud modem. Instruct the other party to set its modem to 300 baud.

   Finally, if someone picks up an extension phone on the line used by the modem, that will also generate screen errors.



Laughing and Learning: Games and Educational Software

The amount of games and educational software available for the C-64 is staggering, especially if you count the thousands of programs in the public domain obtainable through users' groups, and the thousands more that you can run by booting up the PET EMULATOR program (see the section entitled the Software Bonus Pack on page 270).

Commodore itself has released about four hundred programs in the public domain on twenty-seven diskettes, each one (usually) selling for $6.95. These programs cover categories such as English, math, computer science, technology, science, history, and educational games. Three disks of more than a dozen programs each contain puzzles, word and logic games, and other mind-teasers. The math and English disks (seven or eight of each type) are arranged by skill level. These disks can be found at dealers' stores or large toy and department stores that carry Commodore products. If you don't have a disk drive, bring the diskette to a users' group and ask someone to copy the programs onto your tape. Since the software is in the public domain, it can be copied legally.

The purpose of this chapter is to survey some typical examples of commercial software, because the cost (generally $20 to $40 per package) makes experimenting rather costly. Some of the companies offering commercial games and educational software are Spinnaker, Sierra On-Line, Sirius, Taylormade, Broderbund,

Infocom, Epyx, and HesWare. There are others, and their names and addresses are easy to find on the packaging.

One good way to shop is to write these companies, requesting a brochure that explains what their programs have to offer. Then visit a nearby dealer and ask for a demonstration to make sure the product lives up to the company's claims. If you follow this method, there's little chance of being disappointed. Unless you have money to waste, don't buy software on impulse, or because you like the illustration on the package.

Some Background

Before microprocessors (like the C-64's 6510 chip) became the brains of home computers, less sophisticated *dedicated* microprocessors appeared in familiar consumer electronic products, such as digital watches, pocket-sized calculators, and interactive toys. The microprocessors used were dedicated to a single function and couldn't be programmed to do anything else.

The most popular early toy was Texas Instruments' Speak 'N' Spell, which audibly chastised the player for misspelling a word. Later on, memory-teasers appeared; these challenging devices displayed a pattern of lights and sounds, and the player had to match the pattern by pressing the appropriate buttons. The toy "remembered" the pattern, and challenged you to do the same. If you failed, the machine emitted an irritating buzz; the reward was a few bars of electronic fanfare.

With the flexibility and memory capacity of microcomputers like the C-64, games have grown up. They can be as absorbing as novels and as complex as programming itself. Other types of games test a player's reflexes and hand-to-eye coordination. At the same time, software developers began to use the type of interactive, gamelike approach of Speak 'N' Spell to create a learning environment that children (and adults) could find stimulating and enjoyable. On microcomputers, learning can occur as a by-product of entertainment. As some educators have put it, children are happy to sit at a keyboard and learn math, spelling, geography and so on, for hours, so long as no one lets them know they're learning something.

The Software Bonus Pack

Commodore's own Disk Bonus Pack was mentioned in Chapter 2 because it contains the program 1541 BACKUP, which makes a duplicate copy of a diskette using only one drive. Many of the same programs are also on a Commodore cassette tape called the Software Bonus Pack. The cassette has all of the software discussed in this section, but for obvious reasons doesn't included any of the disk-related *utilities* (such as 1541 BACKUP).

The Software Bonus Pack, (for both disk and tape) contains games and educational software. These are not elaborate programs (the kind you'd expect for the typical $40 price tag). For $19.95, however, the two dozen (three dozen on disk) bonus programs are well worth the price. (It's called the Bonus Pack because it was initially included free with the VIC-1541 disk drive.)

Most of the programs can be LOADed and RUN like any BASIC program. A few (and they are well marked) are written in machine language and require the LOAD "NAME",8,1 syntax followed by a SYS command, which the Bonus Pack manual specifies in the instructions.

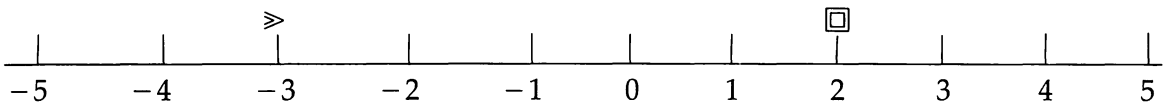
BITS AND BYTES teaches you about the relationship between binary and decimal numbers in a question-answer format. After you've read a screen, press the SPACE BAR to see the next one. The program doesn't let you proceed to the next screen, however, until you've answered a question showing that you've understood the current screen. The insight into how a byte is formed helps considerably in using the PEEK function and POKE statement in BASIC.

Two Hints: When the program loads, the graphic design is delightful, but the text is virtually unreadable because of the screen colors. The background and text colors can be changed instantly by pressing the F3 and F5 keys. You'll be much happier with a reliable black and white combination. F7 starts the program. Second, to leave this *or any other* program before it terminates, press STOP-RESTORE. Then load the directory to see the other programs, or type NEW <RETURN> to clear memory.

SNOOPY MATH teaches quick addition and subtraction of positive and negative numbers. The opening menu allows you to (1) read instructions for students, (2) read instructions for teachers, (3) play the game, or (4) quit the program. The next menu

invites you to choose one of five levels of difficulty from Cadet to Ace.

The object of the game is to help Snoopy shoot down the Red Baron. The playing field is a set of numbers:



Snoopy is at one position, the Red Baron at another. You have to hit the Red Baron by adding to Snoopy's position the number that allows him to reach the Red Baron. Snoopy gives you his famous "DRAT!" if you miss. The higher levels of difficulty allow you less time to type the correct number. If you select the Ace category, you virtually have to shoot (or add) on sight to score a hit.

LEMONADE is for a slightly older group. This program sets up a lemonade stand and challenges you to run it as a successful business. It provides information about the cost of supplies and ingredients (sugar, lemons, cups, and so on), as well as a weather prediction. The player then orders the amounts necessary, and establishes a price per cup for a given day. The computer responds with a profit and loss report for the order.

The player must learn to reason by experience, finding the right combination of supplies, ingredients, and pricing to come out with a profit. The games is guaranteed to stimulate a good deal more business acumen than junior's paper route.

MORTGAGE figures your monthly payments given the interest rate, principal, and duration of the loan. AMORT TABLE shows what part of your payment is interest, and how much (or little) of your house you'll own after each payment for the full term of the mortgage. ORGAN is a compelling demonstration of the C-64's sound chip in action. The program presents an organ keyboard with each note labeled by a different letter or number key. The function keys allow you to alter the octave, waveform, and other variables. You can then take over and learn to play the C-64 as if it were a monophonic synthesizer. The menu provides a choice for polyphonic (more than one note simultaneously) sound, but the results are not true polyphony. (The C-64 can actually be connected to your stereo speakers with inexpensive cables. For

guidance, ask at a users' group — see Appendix B — or stereo shop.) There are many more demonstrations of the sound and music capabilities of the computer in the Bonus Pack.

Loading PET EMULATOR enables the C-64 to run virtually any of the thousands of programs (both commercial and public domain) designed for Commodore's PET. The PET, which has been in use in schools and homes since 1977, has a broad and interesting library of software, especially in the educational field. Much of it, as noted at the beginning of this chapter, has been revised for the C-64 by Commodore and can be purchased inexpensively on disk.

A less obvious educational benefit of the Bonus Pack is that most of the programs are written in BASIC. After using the software, LIST the programs to your screen or printer and study them. You won't find a better opportunity to learn how professional quality software is designed and written.

Learning with PILOT and LOGO

Although BASIC is the lingua franca of the computer world, it's not necessarily the best or only suitable language for learning how to program. Many educators and programmers are convinced that either PILOT or LOGO can teach you how to work with computers better and faster than BASIC. These languages are available on diskettes for the C-64.

PILOT (or Programmed Inquiry, Learning Or Teaching) is a simplified computer language with commands that are abbreviated to one or two letters: for example, T for type, M for match, S for clear the screen, J for jump (to another part of the program), and so on. Altogether, there are twenty such commands or *operation codes* (*opcodes* for short).

Using PILOT, it's very easy to construct question-and-answer exercises for the classroom. This language is much more forgiving than BASIC and, thanks to the MATCH command, it will recognize misspelled words and certain cases of incorrect syntax. Taken together, PILOT's features reduce the student's frustration level and bring successful programming within the grasp of beginners.

PILOT also offers simplified sound and graphics commands, although it doesn't have *turtle* graphics, a dynamic type of drawing program developed in conjunction with LOGO (see below). Turtle graphics is available in a product called Vanilla PILOT, sold on disk by Skyles Electric. Commodore PILOT relies on simplified graphics commands, such as D (draw), P (point), F (fill) with color, and R (remove). PILOT also includes a "sprite editor," which enables you to use sprites without the PEEKs and POKEs. The V (voice) command is more difficult to use, because it requires understanding the SID chip. PILOT, however, is still the most approachable introduction to programming for grade-schoolers.

LOGO, invented by Seymour Papert, a professor of computer science at the Massachusetts Institute of Technology, was designed to be "simple enough so a five-year-old could write a program in the first few minutes of contact with a computer and sophisticated enough so a computer scientist would find it congenial and rich." LOGO was derived from a language called LISP, a *list-p*rocessing language favored by programmers in the artificial intelligence field.

LOGO has been accepted enthusiastically by many programmers as being superior to BASIC in many ways. For beginners, LOGO encourages a student to break down a problem into its constituent parts. Although that's the essence of good structured programming, BASIC commands are not always conducive to that approach. LOGO allows you to build a program in stages, using commands that your program itself defines in its earlier stages. Programmers can thus define their own commands *recursively*. Languages that let you define new commands are called *extensible*, because you can extend the primitive language according to your own needs. It's because of these characteristics that many programmers believe that LOGO leads to good structured programming. Structured programming is easier to follow, modify, and correct than programs without clearly defined structures.

LOGO introduced the idea of *turtle geometry* to programming. The *turtle* was originally a mechanical object that resembled a turtle, but in LOGO it became a cursor that responds to commands like

```
PENDOWN
FORWARD 10 LEFT 90
```

by drawing a line on the screen 10 units long and then making a 90-degree turn to the left. Rather than specifying the coordinates on a pixel grid, turtle graphics is dynamic; it forces you to visualize the line as it's being drawn.

According to Papert, the dynamic approach to graphics gives students of programming a better grasp of how to use the computer.

Whether you choose to learn with LOGO or not, this imaginative language offers a different type of programming environment than BASIC, and it has sufficient power to emerge as a genuine alternative for serious programmers.

FACEMAKER

Spinnaker (Cambridge, Mass.) has created several educational programs for youngsters from four to ten years old, and other programs with levels of difficulty that can challenge adults. Most games are available on both cartridge and diskette. FACEMAKER lets the child construct faces out of five features: mouth, eyes, ears, nose, hair. There are six choices of each feature, so the number of possible expressions is sufficient to keep your grade-schooler interested. After building a face, the child selects from seven facial expressions (wink, smile, frown, cry, and so on). The child chooses an expression by pressing RETURN, and the face is briefly animated, accompanied by appropriate sound effects.

When the expressions have been mastered, the child can play a memory game. The child must concentrate on a series of facial expressions, and then press the letter for each in the proper order. The face gives you the "raspberry" if you're wrong, and it reinforces a successful answer with "GREAT!" and a smile.

The program should be fun for a youngster who likes to draw and who knows the letters of the alphabet. You can't select facial expressions without reading and pressing the correct letter. The graphic possibilities, however, are too limited to justify Spinnaker's age estimate of four to twelve. From beginning readers to eight or nine years is more like it. As with all educational

Photo 9–1: FACEMAKER.

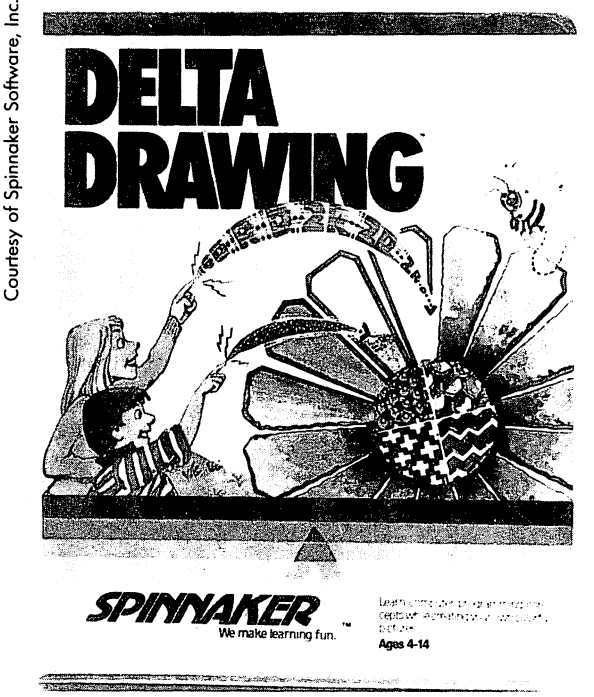


Photo 9–2: DELTA DRAWING.

programs, one undeniable benefit is that your child will acquire pleasant associations to using a computer.

DELTA DRAWING is another story. The program allows direct movement of a turtlelike cursor, which can draw lines and fill closed figures with color. The program can also draw mirror images, repeating figures, and can shrink or enlarge any segment of the screen by a factor of 16. If necessary, you can fill a single pixel, or picture element. This allows you to create complex spirals, mazes, and "fireworks" display, and to place some figures inside others.

DELTA DRAWING has a text mode in which programmed instructions and editing commands can be sent to the cursor. You can switch easily between text (T) and graphics (G) by pressing a key. Graphics programs in the text mode are created automatically as you draw, and these can be saved on tape (but not to disk), and printed with an appropriate dot matrix printer. The

manual claims compatibility with these printers: Atari, Commodore, Epson, IDS, and Okidata 82A. But if you're shopping for a printer to use with DELTA DRAWING, test it with the software before making the purchase.

The instruction manual is generally easy to follow, but there are numerous commands, and only a few of them have easy to remember names. An adult can spend at least eight hours getting familiar with the possible commands, but the rewards are substantial. The program's numerous options provide lots of room to grow. If you're impatient for interesting results, there are ready-made programs you can type in the text mode to create a sailboat, a desert island, and numerous graphic designs.

Games People Play

A rose may be a rose, but not all computer games are alike. The two basic types are known as *adventure games* and *arcade* games. The first type uses the computer to create a fantasy world in which the player participates by trying to solve a crime, find buried treasure, or rescue a kidnapped princess from an alien world.

In many instances, adventure games leave the graphics to the player's imagination. The games provide descriptions of the environment and characters, and then answer questions or respond to commands typed by the player. The diskette acts as a data base full of clues to solving the riddle. Since you, the player, are often the hero of these imaginative sagas, one game developer at Infocom refers to the company's products as "participatory novels."

The arcade-type game, however, is typified by fast action — on a number of skill levels — splashy graphics, and imaginative sound effects. The first of its kind was PONG, an electronic Ping-Pong game created by Atari's founder, Nolan Bushnell. (Atari, Inc., was a video game company before it entered the home computer field.) In today's arcade game, players are usually in combat against enemy invaders, or they must avoid traps, such as an asteroid shower, laser blasts from passing spaceships, or random bombs descending at various speeds on the screen.

These games were first introduced at video game arcades, hence, the arcade label. Thanks to the C-64's high-resolution graphics and built-in sound-synthesizer chip, you can get almost

the same vivid action on a home computer — without waiting in line for your turn, and without dropping in all those quarters.

If you're a game player already, you know that the distinction between adventure and arcade games is becoming blurred. In some games, you'll find elements of both styles.

Adventure Games

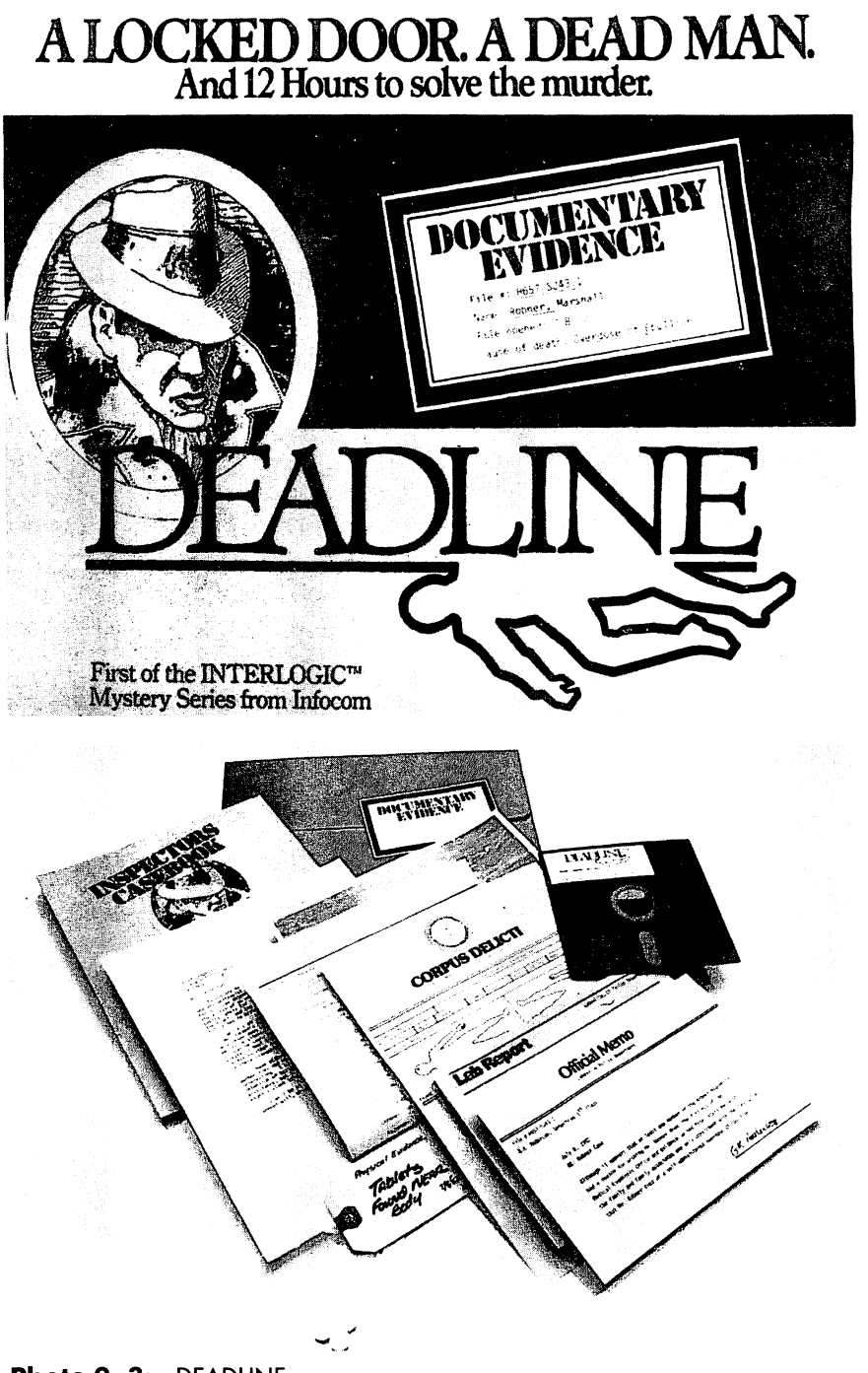
LABYRINTH, available on the Software Bonus Pack discussed above, is a game that depends on being able to visualize spatial relationships. When the game opens, the player tells the computer to draw a "rat's maze" that can vary in its complexity. The player sees a top view of the maze and must then enter it. At this point, the screen shows what it looks like from inside the maze — a rat's-eye view, if you like.

Once in the maze, you can move forward (press F), to the left (press L), or to the right (press R). If you lose your sense of direction, which is inevitable in one of the larger labyrinths, you can press H for help, which consists of looking at a top view of the maze and seeing your position in it. To challenge yourself, or your opponent, set a limit to the number of H's you can press.

If you're the type who gets lost following a road map, escaping from a 10 × 10 LABYRINTH with only three "helps" should make wending your way through a strange part of town seem like child's play.

The classic adventure game for the C-64 is ZORK I, now part of a ZORK trilogy (II, and III have been added as sequels). ZORK deposits you at the edge of "The Great Underground Empire," where you are challenged to find hidden treasures and return without losing your life. ZORK is an interactive disk-based game in which you are matching wits with the programmers, who have mapped out the world in advance. There are pitfalls, treacherous dungeons, intriguing characters, and hidden dangers, as well as hidden treasure.

The game progresses in a series of questions and answers. ZORK has a vocabulary of about six hundred words, and it answers questions starting with WHAT and WHERE. It also obeys commands, such as S (move south), E (move east), N(orth), and W(est), ENTER and EXIT. After you issue the command, ZORK describes your new location and the objects it contains. Discovering the



Courtesy of Infocom, Inc.

Photo 9–3: DEADLINE.

treasure earns you points, and returning above ground to store it in the trophy case earns you more points. The game keeps track of the number of moves, and the number of points. You receive a rating when you quit. You can also save your current game location onto a data disk and restart at the same point later.

There are no graphics, but the evocative descriptions help you visualize the environment with accuracy. ZORK has a sense of humor, too, so if you make any stupid moves, you can expect a sarcastic response. A DIAGNOSE command gives you a report of your physical condition — you lose points for getting injured or killed. Pencil and paper are necessary tools for surviving this adventure. Without drawing your own map of the underground realm, as it is revealed in your explorations, you can't possibly find your way back.

ZORK is absorbing and intellectually stimulating, but you have to enjoy matching wits with the programmers who have hidden the treasures, secrets, and clues on disk. If you loved Tolkein's *Lord of the Rings* or enjoyed the numerous exploits of Inspector Sherlock Holmes, ZORK is for you. But if you're short on patience or easily frustrated, ZORK will quickly drive you to type some unseemly expletives on the keyboard. The game responds to swearing with an apt, "You ought to be ashamed of yourself!"

ZORK maps and a book of clues can be purchased from Infocom. These are described in the *User's Guide* enclosed with the game diskette. Infocom also offers other brain-teasers. DEADLINE requires you to solve a murder within twelve game hours (calculated by a clock related to your moves). WITNESS is similar to, but not a sequel to, DEADLINE. SUSPENDED finds you awaking from suspended animation in an alien world with six robots of varying abilities at your service. You've got to solve a nest of fantastic problems. The game has several skill levels, and the goal is to improve your problem-solving score.

Arcade Games

If you're going to play arcade games, you need to add a joystick to your system. Although Commodore doesn't manufacture a joystick, there are a variety of devices to choose from. There's not much difference among joystick firing buttons, but the throttle action varies considerably from one model to another. Some are

Courtesy of Infocom, Inc.

Photo 9–4: SUSPENDED.

loose and responsive; others are harder to move around but seem to offer tighter control. Beware of an extremely sluggish joystick; after a few weeks of game playing, you can actually develop tendonitis in the wrist. Largely, the choice of joystick is a matter of money and personal taste. An in-store demonstration is worth a thousand words.

Arcade games derive their excitement from graphics, sound effects, and the quick action on the screen. Most of these games are played with joysticks (one or two, depending upon how many players the game allows), which determine the movement of your gamepiece and when you "fire." The action generally involves shooting a moving target, although lately there are more "passive" action games that emphasize *skill* rather than *kill*. Your reflexes determine how well you can avoid flying objects, potholes, trap doors, and a variety of unexpected obstacles.

Two examples of the less aggressive games are distributed by Sierra On-Line. The object in FROGGER (on disk or tape) is to maneuver your frogs safely across a busy highway and a fast-running stream to a safe, dry lilypad. The obstacles include various vehicles moving along the highway and plenty of dangers floating downstream, such as crocodiles, snakes, and otters. Of course, you can always jump on a passing log for a breather. After escorting five frogs safely home, you advance to the next skill level, where the pace picks up considerably. The graphics are vivid in detail, and the background sound and music is exciting.

KICKMAN (on disk or cartridge) is gentler and more aesthetic, but not as fast. The scene is a city street, represented in three dimensions. A clown on a unicycle is performing on the street, and from overhead colored balloons are falling. The object is to have the clown catch and balance the balloons on his hat. The balloons fall at different rates of speed, but the clown can kick them up in the air again if he gets to them before they hit the street (hence, the title). If you're not careful, the clown falls off his cycle and the balloons go flying.

With overtones of futurism and space wars, HesWare's GRIDRUNNER is a more typical action game. The playing field is the Grid, an electric power station orbiting Earth in the next century. The Grid is represented as an X-Y graph on the screen. You operate a patrol ship charged with the task of protecting the Grid from Droid pods sent out to steal electricity. You have to

shoot them down without getting hit in return. ATTACK OF THE MUTANT CAMELS, a sequel to GRIDRUNNER, makes the action more interesting by adding mutating pods and Droids that destroy the patrol ship when they're hit.

Another popular game combines the adventure and action styles. Epyx's TEMPLE OF APSHAI (on disk) takes you through some treacherous rooms, allowing you to pick up helpful items along the way. Survival depends on wits as well as speed. Graphics is combined with a narrative theme.

APPENDIX A:
How to Type in Programs from Listings with Graphics

When typing certain keys, such as SHIFT-CLR, the C-64 screen represents what you've typed with a graphics character — in the case of SHIFT-CLR a "heart" in reversed video. As noted in Chapter 3, programming in the quote mode also produces graphics characters on the screen. This can be the source of confusion when you read the LISTing of a program created by a friend or published in a magazine. Unless there are explanatory notes attached to the program, you must figure out from seeing the graphics character which keys to type. The following chart will help you through those LISTings. Find the character you see on the chart below and press the keys indicated.

SEE:	PRESS:
	SHIFT + CLR HOME
	CLR HOME
	SHIFT + ↑ CSR ↓
	↑ CSR ↓
	SHIFT + ← CSR →
	← CSR →
	CTRL 9
	CTRL Ø
	CTRL 1
	CTRL 2
	CTRL 3

SEE:	PRESS:
	CTRL 4
	CTRL 5
	CTRL 6
	CTRL 7
	CTRL 8
	C= 1
	C= 2
	C= 3
	C= 4
	C= 5
	C= 6

SEE:	PRESS:
	C= 7
	C= 8
	F1
	F2
	F3
	F4
	F5
	F6
	F7
	F8

APPENDIX B:
Commodore Users' Groups

A *users' group* is an aggregate of people who meet on a regular basis to share information about a particular brand of computer. The first Commodore users' groups appeared in late 1977 with the production of the PET (Personal Electronic Transactor), Commodore's first personal computer. Typically, Commodore groups vary in size from a dozen to hundreds of members. The Toronto PET Users' Group (which includes C-64 users as well), located near Commodore's corporate headquarters, is one of the oldest groups and has 10,000 members. Of course, not all members attend the regular meetings.

Originally, the users' group was a response to an environment in which knowledge was at a premium. Because the home computer field was so new there were no books and very few manuals that showed people how to use the equipment. There was also little or no software available, so users had to write their own. The users' groups began as informal schools of home computing, and the students were enthusiasts and adventurers.

Now, users' groups are broad-based, including members of varying expertise, from experienced programmers to absolute beginners. You should never shy away from joining a group because you feel you don't know enough. The basic aim of the group is for the members to educate each other. Besides, even experts were once beginners, so there's no reason to be embarrassed. Users' groups are ideal environments for sharing ideas, programs, and problems. Although some groups have the name PET or VIC-20 in the title, they may also include C-64 users.

A listing of over 250 users' groups in the United States, Mexico, and the United Kingdom can be found in all issues of *Commodore: The Microcomputer Magazine*. This magazine is also available on-line through CompuServe in the Commodore Information Network.

APPENDIX C:
A Partial List of Commodore Bulletin Board Systems

An electronic bulletin board system (BBS) is a privately operated computer system with an auto-answer modem that users can call via modem to exchange information, post ads or messages, trade programs, or simply browse out of curiosity. (For instructions on operating the modem and terminal software, see Chapter 8.)

Some phone numbers on the list may no longer be operative, but it's a place to start. Asking your local Commodore dealer and inquiring at a local users' group is another way to plug into a BBS in your area. There are undoubtedly many more BBSs than those included in this list. To obtain a copy of the *Bulletin Board Directory of North America*, available through mail order only, write to: BBS Directory, P.O. Box 4150, Beach Station, Vero Beach, FL 32964 ($5.95).

COLORADO
Front Range Bulletin Board (Ft. Collins)
(303) 223-4305
24 hours a day, 7 days a week.

ILLINOIS
Commodore Public Bulletin Board System (Chicago)
(312) 397-0871
24 hours a day, 7 days a week.

Video King (Chicago)
(312) 674-6502
24 hours a day, 7 days a week.

INDIANA
AVC Computer Center Bulletin Board (Indianapolis)
(317) 255-5435
24 hours a day, 7 days a week.

MASSACHUSETTS
MASSPET Bulletin Board (Boston)
(617) 824-4878
7 P.M.–9 A.M. Monday through Friday,
24 hours on weekends.

MISSOURI
Commodore Communications (St. Louis)
(314) 625-4576
24 hours a day, 7 days a week.

Kansas City PET User Group Bulletin Board
(816) 257-2502
24 hours a day, 7 days a week.

TEXAS
RPCC
(214) 996-6808
24 hours a day, 7 days a week.

UTAH
Commodore Utah Bulletin Board (Salt Lake City)
(801) 277-3913
24 hours a day, 7 days a week.

WASHINGTON
Northwest Commodore User Group Bulletin Board (Edmonds)
(206) 743-6021
24 hours a day, 7 days a week.

WISCONSIN
C.U.S.S.H. (Racine)
(414) 554-9520
24 hours a day, 7 days a week.

WYOMING
Southeast Wyoming Commodore Bulletin Board
(307) 637-6045
24 hours a day, 7 days a week.

CANADA
ASC Microsystems Bulletin Board (Thunder Bay, Ontario)
(807) 345-7199
6 P.M.–8:30 A.M. Monday through Wednesday;
9 P.M.–8:30 A.M. Thursday through Friday;
24 hours on weekends and holidays.

Nortek Bulletin Board
(416) 487-2593
24 hours a day, 7 days a week.

PSI Word Pro Bulletin Board (Toronto, Ontario)
(416) 624-5431
7 P.M.–9 A.M. Monday through Friday,
24 hours on weekends.

TPUG Bulletin Board (Toronto)
(416) 223-2625
24 hours a day, 7 days a week.

Glossary

access: The means the computer uses to retrieve information from an external memory device.

acoustic coupler: A modem with rubber cups in which the telephone handset is inserted.

address: The name of a memory location in the computer's RAM.

alphanumeric: Consisting of either letters or numbers or both.

applications: The tasks, such as word processing, telecommunications, database management, and so on, for which computers are typically used.

ASCII: American Standard Code for Information Interchange. A seven-bit code adopted as a standard for exchanging information over communications lines. The seven-bit binary number represents 128 decimal numbers, each of which stands for a particular character. The C-64 uses a variation called PET ASCII.

BASIC: Beginners All-Purpose Symbolic Instruction Code. A high-level language designed to simplify programming.

baud rate: A measure of how fast information travels over communications lines, roughly equivalent to the number of bits per second.

BBS: Bulletin Board System. A computer with an auto-answer modem that can be called free of charge.

binary: A two-value numbering system in which a digit can have the value 1 or 0.

bit: Short for "*b*inary dig*it*," the smallest unit of information in the computer's memory.

block: A sector of a floppy diskette that can store 256 bytes of information.

boot up: "Start up" the computer, or "load" software.

branching: A programming technique which causes the computer to follow a series of instructions that are out of sequence.

bug: A programming error that prevents the program from working.

byte: A string of eight consecutive bits representing one character; the computer's memory is measured in bytes.

cold start: Restarting or resetting the computer, erasing everything in memory.

copy protect: To prevent commercial software from being copied in a disk drive.
CPU: Central Processing Unit. The microprocessor chip that controls the operation and processing functions of the computer.
crash: When a program stops working, and the computer must be shut off to restart the program.

daisy wheel: Type of printer that prints whole characters like a typewriter (see also *dot matrix*).
data: Information.
database: An organized collection of information that can be "managed" by software.
debug: Remove programming errors (see *bug*).
decay: Rate at which a note or sound falls from its peak volume.
default: A value preselected by software, when the user doesn't override the selection.
diskette: A floppy disk used with a disk drive for storing data.
directory: List of files on a diskette.
DOS: The Disk Operating System that allows the computer to work with the drive.
dot-matrix: A printer that creates characters out of tiny dots made by the pins of a matrix.
downloading: Receiving a file via modem from another computer.

EMAIL: Electronic mail. Text sent via modem to be stored in another computer and "picked up" by the addressee's modem.
envelope: The ASDR (attack, sustain, decay, release) of a musical note.
external memory: The Datassette recorder or VIC-1541 disk drive; also known as "mass storage," external memory retains information while the computer is turned off.

file: An aggregation of data stored on tape or disk.
floating point: positive or negative number that can contain a decimal point.
floppy disk: See *diskette*.
flowchart: Drawn prior to writing the program, the flowchart uses standardized symbols to illustrate the major steps of the program.

formatting: Dividing a disk into tracks and sectors, enabling it to store and retrieve information.
function keys: The F1-F8 keys, whose functions vary with different software. The function keys are programmable.

global: Ranging over the entire file. In word processing a global change replaces one word with another throughout the file.
graphics: Images created on the TV screen or monitor.
graphics mode: The mode in which the computer displays uppercase letters and (along with the SHIFT or COMMODORE keys) graphics characters (see *typewriter mode*).

hard copy: Text, LISTed programs, or graphics printed on paper.
hardware: The computer itself or its peripheral devices, disk drive, modem, printer, and so on — computer *equipment*.
header: The first line on a page of word-processed text, usually containing a page number and "running head" (or title). Commodore manuals sometimes use *heading* to mean formatting or naming a disk.
high-level language: A programming language that uses English-like commands to make programs easier to write and understand.
host computer: The computer of an information service that receives calls from the public.

input: Information read into the computer's memory.
integer: A whole number, without its decimal component.
interactive: A type of program that allows the user to carry on a dialogue with the computer.
interface: A cable connecting two devices — usually with different data specifications, allowing them to work together.
I/O: Input/Output.

justify: Aligning a margin on the left or the right.

K: Kilobyte. Although the prefix *Kilo-* means "one thousand," 1K is actually 1,024 bytes of memory.
KERNAL: Part of the Commodore 64's operating system.
key word: A word in the BASIC programming language.

language: A set of words and rules (or syntax) for using key words in a way that the computer understands.

logging on: Establishing a communications link via modem with another computer.
LOGO: A high-level computer language that encourages structured programming and learning through graphics.
loop: A repetitive series of instructions in a program.

machine language: The code directly understood by the computer; BASIC instructions are translated into machine language (or code) by the BASIC interpreter.
memory: A chip that allows the computer to store (permanently or temporarily) data and instructions (see *RAM, ROM*).
menu: A list of options offered by a program.
microcomputer: A vague term usually applying to computers that can sit on a desktop.
microprocessor: A programmable integrated circuit on a silicon chip (see *CPU*).
mode: Method or category of operation, for example, the edit mode versus the format mode (Chapter 5); the immediate mode versus program mode (Chapter 3).
modem: MOdulate-DEModulate. A device that converts digital information into audio signals, or audio signals into digital information, for the purpose of sending and receiving data over communication lines.

nanosecond: A billionth of a second. The unit of time in which computer operations are sometimes measured.
nesting: Containing one loop with another loop in a program.
numeric: Characters consisting of numbers only (see *alphanumeric*).

on line: In direct communication with the CPU.
operating system: The set (or sets) of instructions that enables the computer to perform processing and input/output functions.
oxide: A magnetic coating on which data is encoded on tape or disk.

packaged software: Software that is sold commercially.
pattern-matching: A means of locating files on disk or tape.
peripheral: A device (hardware) connected to the computer.

PILOT: Programmed Inquiry, Learning Or Teaching. A programming language designed for use in the classroom, especially with children.

pixel: Picture element. The smallest resolvable dot out of which screen images are composed.

port: An electrical connector allowing access to the computer.

program: A series of numbered instructions the computer can understand and execute.

protocol: The rules or conventions governing the communications between two computers with different data specifications.

QWERTY: The common typewriter keyboard, adopted by the Commodore 64, in which the letters q,w,e,r,t, and y run from left to right in the top letter row.

RAM: Random Access Memory. Memory that can be written in, read from, and erased.

random access: A means of retrieving data such that access time is not related to the position of the data, as on a floppy disk; high-speed memory.

read/write head: The device that stores and retrieves the data on tape or disk.

ROM: Read Only Memory. Memory that can be read only, but neither written in nor erased.

run: Execute a computer program, often with the RUN command, or by loading a program that runs automatically.

scratch: Commodore jargon for erase or delete a file on tape or disk.

screen: The display device of the computer, either a TV screen or the Commodore 1701 monitor.

serial access: Retrieving data in the sequence in which it was stored, as in the operation of the Datassette recorder.

SID: Sound Interface Device. The chip that governs the creation of sound.

smart: Computer "intelligence." The computer's ability to make comparisons and take actions determined by the result of the comparison.

software: The programs that make computers useful and fun.

sort: To order or sequence data by a particular category or field (Chapter 7).

sprite: Commodore jargon for a graphic image that exists autonomously in memory and can be moved about on the screen.

string: A sequence of characters, usually appearing between quotes in a PRINT statement. Strings are taken literally, even when they contain numbers, as opposed to being given numerical values.

syntax: The grammatical rules or conventions governing the use of statements in a computer language. (The word has a similar connotation in "natural" languages — English, French, German, and so on.)

system: The interrelated components that work with the computer; also, a program, as in "the system disk" — the disk with the program on it.

terminal software: A program that turns your computer into a terminal for communicating with another computer via modems.

typewriter mode: When the computer displays letters in upper and lower case (see *graphics mode*).

uploading: Sending files via a modem to another computer.

utilities: Programs that assist in the operations of the computer and its input/output devices.

variable: A name that stands for a memory location where information is stored.

VIC: Video Interface Chip. The Commodore 64 chip responsible for images on the screen.

volatile storage: Data in storage is lost when the power is turned off — a characteristic of RAM memory.

warm boot: Restarting the computer while it's running — without turning it on and off again (see *cold start*).

write-protect: To attach an adhesive tab to a floppy disk to prevent both writing and erasing of data on that disk.

Selected Bibliography

In addition to describing many ways to use the C-64, this book has presented an introduction to programming. Its basic premise is, as must be clear by now, that *applications* are no less important to discuss than programming. Ideally, this book will be the first one a new C-64 owner reads.

The books listed here, however, are devoted exclusively to programming and present the subject in much greater technical detail. The first book on the list is essential for serious programming. The last two consist primarily of sample programs — some of them useful, some just for fun — and do not include systematic instruction.

The books numbered 2 through 7 are ordered according to *my opinion* of their completeness and readability. Other opinions, including yours, may differ, so be sure to acquaint yourself with all of the books on programming the Commodore 64 before choosing one.

Following the books is a list of periodicals that contain regular features on the Commodore 64. Periodicals are an excellent source of new product reviews, instructional articles, and public domain software. Whenever possible, browse at your local newsstand or request a sample copy from the publisher before subscribing. If a periodical's line of interest coincides with yours, you'll eagerly await the new discoveries and delights found in each month's issue.

Books

1. *Commodore 64 Programmer's Reference Guide*. West Chester, Penna., and Indianapolis, Ind.: Commodore Business Machines, Inc., and Howard W. Sams & Co., 1982.
2. *Your Commodore 64*, by John Heilborn and Ran Talbott. Berkeley, Calif.: Osborne/McGraw-Hill, 1983.
3. *Commodore 64: Getting the Most from It*, by Tim Onosko. Bowie, Md.: Robert J. Brady Company, 1983.
4. *Commodore 64 User's Handbook*, by WSI Staff. Cleveland, Ohio: Weber Systems, Inc., 1983.
5. *The Elementary Commodore 64*, by William B. Sanders. Chatsworth, Calif.: DATAMOST, Inc., 1983.

6. *Computer Programming with the Commodore 64*, by L. R. Carter and E. Huzan. TEACH YOURSELF BOOKS/Hodder and Stoughton in the United Kingdom, and David McKay Co. in New York, 1983.
7. *Commodore 64 Computing*, by Ian Sinclair. Englewood Cliffs, N.J.: Prentice-Hall, 1983.
8. *More Than 32 BASIC Programs for the Commodore 64 Computer*, by Tom Rugg, Phil Feldman, and Western Systems Group. Beaverton, Ore.: dilithium Press, 1983.
9. *101 Programming Tips & Tricks for the VIC-20 and Commodore 64*, by Howard Adler. Woodsboro, Md.: ARCsoft Publishers, 1983.

Periodicals

Commodore: The Microcomputer Magazine, Commodore Business Machines, Inc., West Chester, Penna. (Subjects: Hardware, software, tips on programming, and in-depth articles about new products. Covers all Commodore computers.) Six issues yearly.

Commodore: Power/Play, Commodore Business Machines, Inc., West Chester, Penna. (Subjects: Games and game programming for the C-64 and VIC-20.) Four issues yearly.

Compute's Gazette, Compute! Publications, Inc., Greensboro, N.C. (Subject: All aspects of computing with the C-64.) Monthly.

Creative Computing, Ahl Computing/Ziff-Davis Publishing, Inc., Morristown, N.J. (Subject: Software and applications for the C-64, VIC-20, as well as other brands of home and small business computers.) Monthly.

Index

D

G

H

I

J

K

L

M

N

O

T

U

V

W

X

Z

> $14.95 FPT USA

Len Lyons
with Joe Campbell and Herb Moore

Using the Commodore 64™

A COMPLETE GUIDE TO GETTING THE MOST OUT OF YOUR COMMODORE 64™

The Commodore 64™ is one of the most popular and versatile home computers available today. **USING THE COMMODORE 64™** is a comprehensive, practical source of information and advice that will teach you about:

- the fundamental components of your Commodore system and how they operate
- programming in Commodore BASIC
- sound and graphics applications for entertainment and education
- word processing with EASY SCRIPT
- money management with Easy CALC RESULT
- telecommunications with the VICMODEM, 64TERM, and SMART 64
- record keeping with THE MANAGER

Written in an engaging and lively style, USING THE COMMODORE 64™ is an indispensable reference for Commodore owners of all ages and levels of sophistication. If you want to get the most out of your 64 at home, in the classroom, or in the office, this is the one book you'll need.

Len Lyons is a well-known professional writer and editor with extensive experience writing for both technical and general audiences. He is the author of THE COMMODORE 64™ CONNECTION. **Joe Campbell** is a professional writer, computer programmer, and author of a book on telecommunications. **Herb Moore** is a computer programmer and author of several popular works on sound and graphics programming.

ISBN 0-201-05156-7

ADDISON-WESLEY PUBLISHING COMPANY