THE NEW, COMPLETE, PLAIN-ENGLISH GUIDE TO TAPPING THE FULL POTENTIAL OF YOUR COMMODORE 64

THE COMMODORE 64 SURVIVAL MANUAL

BY WINN L. ROSCH

How to give your C-64 the power of expensive computers

☐ Foolproof step-by-step tips and secrets

☐ Hundreds of instant cures for problems

☐ Exactly what to do when things go wrong

PLUS ☐ Programming ☐ Hardware ☐ Software

☐ Peripherals ☐ Software reviews ☐ Money-savers

THE COMMODORE 64 SURVIVAL MANUAL

Bantam Books of Related Interest
Ask your bookseller for the books you have missed

THE COMPLETE BUYER'S GUIDE TO PERSONAL COMPUTERS by Tim Hartnell and Stan Veit
THE FRIENDLY COMPUTER BOOK: A SIMPLE GUIDE FOR ADULTS by Gene Brown
HOW TO GET THE MOST OUT OF COMPUSERVE by Charles Bowen and Dave Peyton
THE ILLUSTRATED COMPUTER DICTIONARY by The Editors of Consumer Guide®
MASTERING YOUR TIMEX SINCLAIR 1000/1500™ PERSONAL COMPUTER by Tim Hartnell and Dilwyn Jones

THE COMMODORE 64 SURVIVAL MANUAL

Winn L. Rosch

Illustrations by Steve Henry

A Hard/Soft Press Book

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

THE COMMODORE 64 SURVIVAL MANUAL
A Bantam Book / August 1984

Sprite and sound assistance: Michael Callery

ISBN 0-553-34127-8

Published simultaneously in the United States and Canada

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

PRINTED IN THE UNITED STATES OF AMERICA

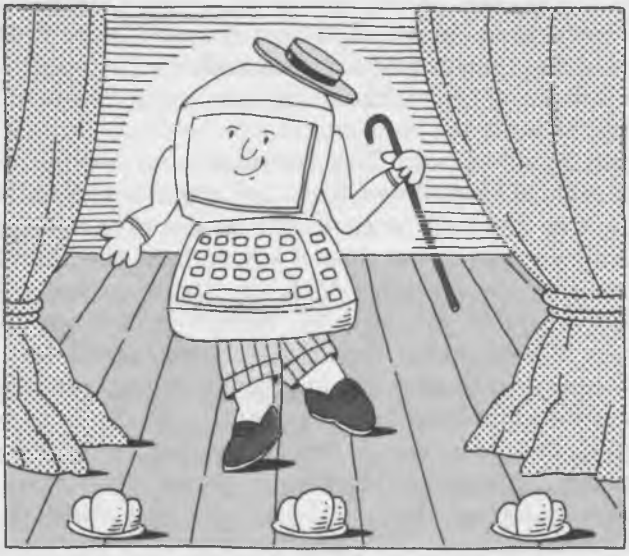
HL 0 9 8 7 6 5 4 3 2 1

To Granny

TABLE OF CONTENTS

1

INTRODUCING THE COMMODORE 64



Nothing can be all things to all people, but the Commodore 64 computer comes awfully close. With the appropriate attachments the 64 will do your paperwork, race through endless columns of math, and even let you battle with alien invaders. It will balance your budget, teach your kids the alphabet, play the latest video games, talk with other computers halfway around the world, type anything from homework to church newsletters, play Bach or rock, and fill your television screen with masterpieces reminiscent of Picasso and Piet Mondrian (not to mention a two-year-old with finger paint).

With the introduction of the Commodore 64, having a powerful and useful computer in the living room, kitchen, recreation room, den, or office is no longer just an engineer's dream, philosopher's wonder, or a rich man's luxury—it's something hardly anyone can be without. No longer can there be any doubt that computers are going to enter your life in a big way. This one is already there. It's not a box to watch and "gee whiz" at; it's a machine to use, take advantage of, and enjoy—once you learn how to use it.

Like most machines, however, the Commodore 64 has its shortcomings. The most immediate one is that it's not as easy

to operate at first as you might have expected. Fortunately, if you have a little patience and the right help—like this book—learning to use the Commodore isn't hard.

In fact, learning to use the Commodore can actually be fun. Unlike the bigger, more expensive business computers, the 64 doesn't give you an evil green stare and dare you to find some businesslike purpose to justify buying it. Rather, it shows you that learning about computers and computing itself does not have to be a chore. It can be as simple and as enjoyable as a game!

No matter what your ulterior motive for buying a home computer, the Commodore 64 has the added benefit of being a fine machine for kids to learn the basics of computing and programming. More than just a toy or tool, the Commodore 64 gives kids an *incentive* to learn about computers. Certainly as an adult you have an incentive to learn to use a computer. When you've finished learning, you'll be able to use the machine to handle your bookkeeping, manage your files, or type your letters. But running a business or balancing a budget will probably not be the reason your kid will want to use a computer. The Commodore 64, however, excels at the one thing your kids will want to learn about: developing their own games. And along the way they'll learn all the fundamentals of computer programming. They may even learn how to think more clearly by organizing their approaches to problems!

That isn't to say the Commodore 64 is not a powerful computer in itself. Should you wish to use it for more sophisticated applications, you'll find locked beneath the Commodore 64's keyboard a microcomputer-on-a-chip armed with more memory than most $10,000 business computers had a few years ago. In addition, the Commodore engineers supplemented that inherent computer power with extra graphics and sound talents. These capabilities enable programmers to develop software with dazzling pictures and melodious tunes. They make the Commodore especially fun to own.

The Good and the Bad of the 64

The Commodore 64 is probably your first computer. But how can you be sure that you bought the right one? When you thought about buying a home computer and you looked around at what was currently available, you were probably struck by how similar all the different computers are in their abilities

and attributes and how they all claim to be the best of everything for everybody. You don't have to be a skeptic to know that all of them can't be "best" and that probably none of them is.

No matter what advertisements or salesmen have to say, no computer, neither the Commodore 64 nor the IBM 3081 mainframe, is the solution to every home computing problem. Like all the currently available home computers, the Commodore 64 has its strong points and its weaknesses. Some may be important to you; others may not. But you should be aware of both the abilities and limitations of any home machine. Accordingly, here's a quick tour of what is wrong and what is right about the Commodore 64.

Chomping at the Bits

The number of *bits* of digital data a computer can handle at one time is a measure of the computer's power. The Commodore 64 is an 8-bit computer. That means that it handles data 8 bits (or 1 *byte*) at a time (see Box 1). In itself, an 8-bit computer is neither good nor bad. More bits are better (in general), but more bits also means more costly. The number of bits a computer uses at once must be a compromise between price and power.

All the letters of the alphabet, numbers, and punctuation marks (as well as over 100 more symbols) can be coded using less than 8 bits. Eight bits is enough to code television pictures with quality high enough for network broadcasting. Eight bits is enough to allow the use of BASIC, the most common computer language, and most other languages as well. In other words, 8 bits is enough to handle most things

BOX 1: TAKING A BYTE

One *byte* is equal to 8 *bits* of digital data. It's a convenient and often-used measurement in computing because 8 bits are enough to code 256 different characters—enough for all the uppercase and lowercase letters of the alphabet, all the numbers from 0 to 9, and over a hundred special symbols that can be used to control the computer. For most purposes, therefore, each byte corresponds to 1 character of memory.

you're likely to want to do with a home computer, at least for now.

Sixteen-bit computers (the usual next step up from 8) can do some things better than 8-bit machines. All else being equal, they can manipulate numbers faster. That's handy if you want to solve three-dimensional simultaneous mathematical equations, but an 8-bit computer is fast enough to add up your bank balance thousands of times while you do it just once.

More bits means that a computer can handle more memory. More memory means that 16-bit computers can handle larger programs with hundreds of thousands of bytes much faster than 8-bit machines. That's not to say 8-bit computers can't use really big programs; they can, but they have to spend precious time shifting portions of programs from one part of their memory to another.

A Matter of Timing

The number of bits that a computer uses at one time is not the only factor that determines how fast it can compute. All computers have built-in clocks that control the exact instant each "thought," or electrical impulse, courses through the machine, how fast each decision is made, and how quickly each number is added. The faster this clock runs, the faster the computer computes and, not surprisingly, the harder it is to design and keep it working properly. Rather than ticking off minutes and seconds, these computer clocks run at rates of *millions* of ticks (or *cycles*) per second. Each million cycles is called a *megahertz* (MHz). The higher the megahertz rating of the computer's clock, the faster it can work its way through computations.

The Commodore 64's clock operates at a around 1 MHz. Although a million of anything may seem like a lot, the Commodore 64's speed is hardly remarkable. Many more expensive personal computers operate four to ten times faster than the Commodore.

The time factor also bears a direct relationship to the graphics function. You won't be able to generate Disneylike animated cartoons with a Commodore 64 or any other 8-bit computer that pokes along. If you do get involved with graphics like fast-paced games, your first concern will be squeezing the most speed out of the BASIC language Commodore sup-

plies for programming. But you'll soon have to explore faster, more sophisticated programming methods and languages.

However, for most uses the Commodore 64 will do just fine. Hundreds of sophisticated programmers around the world have figured out what makes it tick and have learned how to get the most from this very versatile machine. The result is that 99 percent of the time the Commodore will be more than fast enough for your needs. You'll have to catch up with your 64, not the other way around.

Memory—and Expandability

The number of bits a computer can use at once indirectly determines the maximum amount of memory that the computer can handle at one time. This usable memory is usually described as a number of *K* or *kilobytes* of memory (see Box 2). Most 8-bit computers can directly handle 64K of memory. Most 16-bit computers can handle over a million bytes—1,000K.

When integrated circuit memory chips were expensive, computers often came equipped with less than their maximum

BOX 2: WHAT'S A KILO?

Just when you thought computers were going to add order to this chaotic world, you bump your head into the term *kilobyte*. Anyone who knows anything about the metric system knows that the prefix "kilo" means "thousands of." Why, then, is a kilobyte 1,024 bytes?

Computer engineers don't think like real people. They count in "base two"—meaning any number over one confuses them. They deal in digital "bits," each of which can be either a one or a zero. Rather than using a number system based on tens, where each column of digits represents a power of ten, computer engineers figure in powers of two. The golden number for the K of bytes, 1,024, just happens to be an even power of two (2^{10}), and engineers thought it was close enough to exactly one thousand to get away with calling it a kilo.

Actually, we're kind of lucky. We get an extra 24 bytes of memory for every K. And the engineers didn't force us to learn a new prefix to mean a memory multiple of 1,024. Just try saying "kilodidecaquad bytes."

memory capacity but with provisions for "expanding" them by adding extra memory chips to boost the number of bytes. Although the price of memory chips has fallen substantially, most 16-bit machines are sold with less than their maximum memory capacity and hence are "expandable." The Commodore 64, being an 8-bit computer, is sold completely equipped with 64K of memory and therefore can't be expanded. Like other design features of the Commodore 64, that's both good and bad—bad because the 64 will never be able to handle some of the very complex programs now being written for 16-bit computers; good because you don't have to buy memory expanders to make the most of the 64's abilities.

The Ins and Outs of the Commodore's Ins and Outs

When compared to much more expensive "business" (rather than "home") computers for ease of use in businesslike applications, the Commodore 64 puts up a poor showing. Many users give it low marks for its keyboard (through which you enter all commands and data into the computer) and its display (which shows you what you're doing and the results of your programs).

Compared to business computers, perhaps the weakest (or certainly the most criticized) part of the Commodore 64 is its keyboard. To be honest, typing on the 64 is not as smooth and sure as it is on a $5,000 computer (or even a $1,000 typewriter). Compared with some others, the Commodore 64's keyboard is just different enough to be bothersome. On the other hand, the basic Commodore 64 sells for less than just the keyboard of one of those expensive business computers, so expectations of perfection are perhaps a trifle unrealistic. And many more expensive computers have keyboards that don't measure up to the Commodore's.

Another weakness of the Commodore 64 when compared to small business computers is that its video display is only 40 characters wide. Most business machines boast 80 characters across their screens. But this "weakness" is not accidental. The biggest limitation of the sharpness of on-screen computer graphics is the screen itself and not the computer. Forty characters across is about the absolute limit of what can be clearly displayed on an ordinary television set. If the Commodore put more characters across its display, it would require an expensive video monitor, costing at least as much as the

whole 64 computer. And if you're going to use a color television set as your computer monitor, more characters would be just indecipherable!

The Commodore 64's relatively narrow display is most objectionable when using word processors and electronic spreadsheets. Either what you see on the screen won't reveal what the final printed sheet will look like, or your spreadsheet may look too narrow to make much sense. Forty characters across, however, is more than many home computers offer. With appropriate adapter modules, you can give your Commodore 64 an 80-character-wide display, but you will also need to buy a special monitor. Most standard television sets (black-and-white or color) will not work, since they are not sharp enough to display 80 columns across.

Taking Orders

Ease of operation is important in helping new users make a graceful transition from reality to the computer world. The Commodore 64 can be both blessing and bane to budding computerphiles.

Above all, the Commodore 64 is designed to be easy to use. Its software and memory arrangement are intended to make it easy for the programmer to take advantage of all the entertainment-oriented features of the machine. Rather than just playing games as well as most home computers, the Commodore 64 helps make games easy to create and write. With a little practice you can turn the pictures and sounds hiding in your imagination into on-screen reality.

While other computers may make you learn special languages called *operating systems* just to use them, nearly all commands you give the 64 are written in a *high-level language*—BASIC. Although for the first few days you'll have problems with translation (as you probably would in learning any new language), this approach gives a cohesive unity to the operation of the computer. You don't have to worry about shifting between a computer language and the operating system and keeping two or more different vocabularies straight in your mind.

But the Commodore 64 system is not without flaws or traps. The 64's disk-drive commands, for instance, can be a lot more tedious to master than those of many other machines. Typing each hard-to-remember instruction in exact, letter-perfect order is a test of anyone's patience. Certainly using

the disk operating system (DOS) included with Commodore's optional C-1541 Disk Drive will make life easier, but the program is not stored in the computer's memory and so must be loaded manually every time you turn on your system. More sophisticated machines load such programs with a single keystroke or completely automatically.

Is the Commodore 64 Really a 64?

To most people the most important part of the Commodore 64 is its last name. In poring through literature and magazines, they've learned one thing. Just as horsepower is the measure of a car's power, the amount of memory—the number of K (or thousands of bytes or characters)—measures a computer's power. The more the better, and 64K is a lot for a home computer. That 64 in the Commodore name, they believe, means the Commodore has 64K of memory. Those people (but certainly not you and I) are wrong! The Commodore 64 *does not* actually have 64K of memory!

There seems to be a contradiction somewhere, doesn't there? Just a few pages ago you read that the Commodore 64 has a full 64K of memory inside it, the absolute limit for most 8-bit computers. Yet if you've ever tried a Commodore 64, you probably noticed that the first thing that it tells you is that it has 38,911 bytes (or thereabouts) of memory free. That figure is the total of all of the random-access memory that is available to the Commodore 64's brain to work in the BASIC programming language. What's the discrepancy? Where's the rest of that 64,000 bytes of memory?

In searching for the missing K, the first thing you need to understand is the difference between different kinds of memory. The working memory of the Commodore 64 is called *RAM*, for *random-access memory*. Each byte of that RAM is a place where a single character—a letter of the alphabet, a number, or a special symbol used to control the computer—can be stored temporarily. When running BASIC, the Commodore 64 has 38,911 places that it can store characters or data, just as the monitor screen advertises. Some of the RAM is used by computer functions, like remembering what is on the screen. The rest of the 64K is allotted to *ROM—read-only memory*.

ROM is different from RAM because characters can only be retrieved (or in computer talk, *read*) from its memory locations. The characters are permanently stored there when the ROM is made. The characters in the Commodore 64's ROM make

up the programs that run the machine. Not only is the program for BASIC coded in the ROM, but even the knowledge that the Commodore is a computer and how computers work is stored there.

Putting that information in ROM is a good idea because it cannot (absent lightning bolts and frenzied, frustrated children attacking with hammers) inadvertently be changed. That means that nothing you can do on the Commodore 64's keyboard will hurt its BASIC language or the other programs stored in ROM that are vitally important to the computer's operation.

If you were expecting a full 64K of memory, it sounds as if you've been shortchanged. You paid for 64K of memory and only got 38,911! In truth, however, rather than being shorted, you've gotten a bonus! For every byte of memory allocated to ROM in the Commodore 64, there is an additional byte of RAM hidden inside, enough to bring the total RAM memory to a full 64K. The only shortcoming is that the additional RAM cannot be used when you are running programs in the BASIC language.

The reason you only get 38,911 of RAM memory to work with in BASIC is that the Commodore 64's 8-bit brain needs to use the information stored in its ROM as part of the 64K of memory it can handle at a time. It can't dip down into the extra RAM when it has its hands full using the ROM memory.

That extra hidden RAM is not worthless, however. Programs that don't use BASIC can be run on the Commodore 64 and take advantage of every byte of it. That means that complex games and sophisticated *applications* programs (those written to handle a specific application or job, like word processors or electronic spreadsheets) have a full 64K of RAM available to them. That's as much memory as you'll find in some business microcomputers costing $5,000 or more! Experienced programmers can switch between using the RAM and ROM memories and even use part of the RAM to duplicate the features and functions contained in the ROM. It's even possible to change these built-in features and functions, and the way the Commodore 64 operates, to suit yourself (though it isn't exactly easy).

In other words, the Commodore 64 really is not a 64K computer. But far from being less (as you might suspect from the price), it actually gives you more memory than it claims. You get a full 64K bytes of RAM *and* nearly an additional 20K bytes of ROM thrown in.

You Need More Than Just a Computer

Just as important as the computer itself are the accessories, or *peripheral devices*, that can be connected to it to make a complete computer system. An inexpensive computer is not much of a bargain if it requires expensive peripherals.

A Commodore 64 system can be complete in the one-unit computer itself. It neither needs nor uses any additional memory to take full advantage of its microprocessor brain. That means you don't have to buy an expensive expansion chassis—ever. And the peripheral devices that Commodore offers (the color monitor, the Datassette, the disk drive, the printers, the modems, and the rest) are reasonably priced.

The availability of *software* is even more important. Computer manufacturers often brag about how many programs are available for their machines. The more programs there are, the more uses the computer can have. But all of the "available" programs are not really available to you because some may be hard to find (if you can find them at all) and others, though advertised, may not have even been manufactured yet.

Software doesn't just happen. It is written. The people who write software usually do it for specific reasons—usually to make money or to make better use of the computer that they own. In the long run, the best-selling computer model will end up having the most software written for it. The reason is that a bigger, more alluring market will attract more of the programmers who write for profit. More hobbyists will develop their own ideas. As one of the best-selling home computers, the Commodore 64 already has a large software base. It will only get larger in the future.

Commodore Shopping Notes

People buy computers for many different reasons—because the machines are on sale, because the kids threaten to hold their breath unless they get the new toy this minute, because a computer would look cute on a desk, because everybody else has one, because—well, just because. But running out to the store and buying a computer is not something to do on the impulse of the moment. If you plan on just taking the box marked "computer" home with you and starting a new life in computing, you'll be sadly disappointed. You must carefully

consider what you want a computer system for and what you plan on doing with it before you decide to buy one. Even after you set your heart on a specific make and model, like the Commodore 64, and the clerk in the store is poised about to squeeze that last bit of life out of your charge card, there are more decisions to make. You must decide what else, beside the computer itself, you should buy to fulfill your computing dreams.

You will be completely equipped to start computing if you leave the store with nothing but the Commodore 64 box under your arm. Included with the basic computer is all that is necessary to connect it to your television set—a cable and a switch box for your television's antenna wire. With that minimal purchase you'll be able to play games and balance your budget by typing programs published in books and magazines into your new computer. But the high hopes you have about making a million from the computer software you are going to write may turn into a flock of angry butterflies in your stomach on the first night you warm up your Commodore 64 and come face-to-face with a greeting commonly known as the *Dreaded Ready*.

At first it may not look like anything that can make someone quake in terror and break into tears. It's just a few words, blue on blue, the last part of the Commodore 64's welcome-to-the-world-of-computing message, but the Dreaded Ready marks the vast gulf between the old world and the new, between the days when machines did their business at a flick of a switch and the computerized present, when machines stoically await your commands and programs before doing anything at all. Both you and your computer are eager to get to work, but there's an overwhelming language barrier. Your new computer tells you that it's "ready" to do anything you want, but you don't know what to tell it to do or how to tell it to do anything.

If you've never really entered the realm of computing or never actually fingered a computer keyboard, you'll need more than just a computer. You'll need something that will put your new investment into action right out of the box so that you can gain confidence. It's important that your first fingertip keyboard dance bring you immediate results and that you know you can get this mysterious machine to work, after all.

If you don't want to go through the ordeal of learning to program before having fun with your new computer, you ought to buy a canned application program—one written for

any specific application, from balancing your checkbook to planning a diet to playing a game of electronic Parcheesi. A canned program (one you can buy off the shelf like a tin of peas, corn, or succotash) will make your computer an instant genius and put all of its power in your hands without requiring you to learn any programming skills. Better still, the program will show you the sort of things you may be able to do once you've learned the basics of BASIC. It will also give you a way of showing off your new computer and reassure you that you can run the machine.

Application programs come recorded on *cassette tapes*, on *floppy disks* (also called *diskettes*), and in *cartridges*. The packages of most application programs for the Commodore 64 will tell you what form they're in, most often by advising you that a disk drive or cassette recorder is necessary. Cartridges plug directly into your Commodore 64 without any additional accessories. However, you'll need a Commodore Datassette cassette unit to use cassette-based programs and a Commodore C-1541 Disk Drive to run the diskette-based programs. Depending on the application programs you choose, you may want to add either the Datassette or disk drive to your shopping list, too.

There are other reasons for considering either the Datassette or disk drive with your initial computer purchase. Even if you're headstrong and have decided not to "stoop" to buying programs that someone else has written (you know you can write your own), even if you've done your homework and have found listings (programs printed out step-by-step so that you can easily type them into your computer) for programs that you want your new machine to run for you, even if your five-year-old child is already a programming whiz and writes software for the Navy in his spare time after kindergarten, you'll be in for a big surprise the first time you flick the power switch off on your Commodore. Every time you perform that one simple act—tapping the power switch off—you wipe the computer's memory slate clean. Every time you turn the computer on, it starts with a clean memory. That means when the power switch goes off, all the programs you have spent hours typing into the machine have been whisked away into oblivion. Every time you switch the computer back on again, you'll have to spend all those hours all over again putting the programs back in.

A *mass storage device,* which in English means either the Datassette or disk drive, lets you preserve all the keystrokes or typing you've done. With cassette tapes or magnetic disks

you can record all the instructions and information that you have so painstakingly typed into your computer and play them back into your Commodore's memory whenever you want or need to.

Of the two, the cheaper alternative is the Commodore Datassette cassette tape unit. It can preserve your own programs and give you access to any applications programs supplied on cassette tapes. Although cassettes have some severe disadvantages when used as your primary means of storing information, they are a workable and (probably most important) cheap choice for keeping your precious keystrokes safe. That way you won't have to retype all your work every time you turn the computer on.

If you do choose to add a Datassette to your shopping list, don't forget to throw a blank cassette or two into the bag with the machine. You won't need batteries or cables or anything else with the Commodore Datassette. The cable is built in, and no batteries are required. The computer supplies the cassette with the electricity it needs.

If you include a disk drive in your first computer shopping spree, don't forget at least one blank disk. Thankfully, though, all cables are included.

You won't need a *monitor* for your new Commodore 64 if you have a standard television set. Included in the computer's packing box are the necessary cables and a switch for connecting the Commodore 64 to your television's antenna terminals. With a color television, the Commodore 64 will show you full 16-color graphics and text. (But don't expect a rainbow to appear on your black-and-white television. The Commodore 64 is smart but not a miracle worker.) Although it's quite likely that a color monitor will give a sharper picture than the typical color television set, you may want to judge whether the on-screen image of the Commodore 64 connected to your television needs any help before you spend money on an accessory that costs more than the computer itself.

You'll probably want to add some sort of AC power distribution system to your shopping list, since invariably you'll run out of outlets. The Commodore 64 itself requires one, and every accessory that you add (except a joystick, paddles, or the Datassette) will require an additional outlet. If you think you can get away with using just a cheap extension cord, think again. The 1541 Disk Drive and Commodore printers are equipped with three-prong, grounded plugs, and the extra prong is designed to be used to eliminate possible grounding problems in the computer system. You'll probably want a

multioutlet power strip to handle everything. Better yet is a surge- and noise-protected outlet strip. Not only will it give you the extra outlets you need, but it will help prolong your computer system's life by eliminating or reducing power-line problems known as *spikes* and *surges*.

If you plan on playing games with your Commodore as well as working with it, you'll want to add a *joystick* or two to your computer shopping list. Although it's possible to play most Commodore computer games on the keyboard alone, it can be so cumbersome as to make the simplest of games frustrating, exactly the opposite of what play should be. A joystick gives the player a much more natural way of reacting to and controlling a game. Most Commodore games are joystick compatible; the box the game comes in will tell you what the program requires and what will work with it. Moreover, a joystick is quite easy to integrate into game programs that you (or the kids) write and may become a necessity sooner than you expect. *Paddles,* on the other hand, are used by relatively few Commodore games, and you'll probably want to invest in them only if one of the games you choose to add to your recreational library requires them.

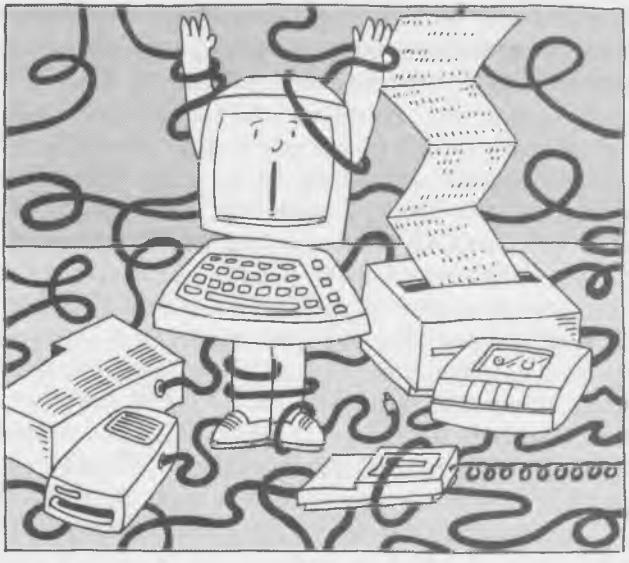
Although a *printer* is a worthy addition to any computer system, you may want to get acquainted with the Commodore 64 before investing in one. Don't rush out and buy one with your computer unless your mind is absolutely set. There are so many different types and models of printer available, each designed to excel at one or more particular job, that you should have a precise application in mind when choosing one. Moreover, you have your choice of using a simple-to-connect and relatively cheap (in every sense of the word) Commodore printer or enduring the hassles of connecting another manufacturer's machine. Most printers, except those sold by Commodore, are not directly compatible with the Commodore 64. Even for experienced programmers, figuring out all the permutations of a Commodore-to-printer connection can be tricky.

Computers should be taken seriously. There's no doubt that the home computer revolution is destined to change our lives. Nevertheless, as you eye the racks, shelves, and display cases at the computer, hobby, or toy store for the applications software packages that will help you save a fortune, start and run a new business, or make the world safe for humanity, drop at least one game program into your shopping cart. If you make a wise choice, your kids will thank you. And you'll probably be just as thankful when, sitting in front of the

keyboard, you need to do something with your new investment/toy and you've run out of the patience and imagination programming requires. The game not only will give you a chance to put the machine to use but also to work out your own frustrations and need for challenge. And you'll learn how other programmers have chosen to take advantage of the vast capabilities of the Commodore 64.

2

BEATING THE SYSTEM



Somehow the word "system" doesn't usually conjure up pleasant connotations. You may have lost your shirt playing the horses with your uncle's surefire system. Or you may have muttered oaths about your ever-late local commuter bus and rail system. Well, tighten your seat belt and dig out your security blanket. You're about to face another system: the *computer system.*

After you've ogled the racks and rows and cabinets of computer equipment at the computer store and decided that you absolutely need one of everything to go along with your Commodore 64, you will become painfully aware that a computer system is much more than a single plastic box. There are enough accessories to plug in, shove in, and key in to keep you adding to your computer system (and subtracting from your bank balance) for years.

One working definition of a system is a many-headed ogre made from an assortment of different but related parts that are all conspiring against you. The essence of a system is that the whole is greater than the sum of its parts. Likewise, a computer system is more than just the plastic keyboard you brought home. It's built from cables and peripherals and software and a seemingly wild-eyed disregard for common sense. And it can be your undoing when it goes awry. If you

jump into computing without understanding the "system half" of your computer, total strangers may start betting whether you reach the poorhouse or the asylum first. The trick to your personal survival in the home computer revolution is to understand what a computer system is and to know how it works.

The power of any computer system is determined by the combination of its two main parts—hardware and software. Computer hardware consists of all the pieces of equipment that have a real, physical existence in hard reality. You can see it, feel it, touch it, and covet it. Software has no physical existence at all. It is just a set of rules or ideas. Although it can be represented either as scribblings on paper or coded as magnetic fields on tape or disk, neither representation is really the software. Rather, it's a procedure, a method, a rule or set of them.

In any computer system the hardware and software fulfill different, complementary purposes. The hardware determines the capabilities of the system or what the system *can possibly do*. On the other hand, the software determines what the computer system actually *does*. Its instructions tell the hardware what to do.

Although software can be refined and developed to expand the capabilities of the computer system, it can never exceed the limits of what the hardware is able to accomplish. Hence, the major limit on the computer system's abilities—the factor determining its power—is the hardware.

Although it's filled with hardware power, the Commodore 64's greatest computing strength is not what's inside but what can be connected to it—the hardware devices that can be added to extend the system's outer edge. By itself, the Commodore 64 is just a box that changes numbers and words into different forms. But when you expand that basic computer system by connecting the Commodore 64 to peripheral auxiliary devices, the system can *do* things—print mailing labels, newsletters, and books, play games on a color video screen, even talk to other computers over telephone lines.

Understanding such a computer system and how it works is best learned by examining its constituent parts, or the *peripherals,* and getting the big picture of the whole shebang. In this chapter, we'll look at the hardware—the framework that gives the Commodore 64 computer system its potential—so you can better appreciate the possibilities locked inside your new computer. We'll consider each piece of hardware and what it does, and we'll point out the strong and weak points of the actual units Commodore and other manufacturers offer.

Journey to the Center of the System—The Commodore 64

The heart of your computer system is the Commodore 64 itself. Inside its plastic case is the brain that controls everything—a digital electronic *integrated circuit* (*IC*) called a *microprocessor*. Like the main brain of any computer, this microprocessor tackles a multitude of jobs besides just tossing numbers around. With the help of similar ICs, it's the computer system's communication center, a switchboard that routes signals from one peripheral to another and from your fingers to the system itself. It's a memory unit that keeps track of details and a clock to make sure everything happens at the right time. It's also the central command center where all the data processing of the computer system is done. Because of its centralized processing functions, the role the Commodore 64—or any computer—actually plays in the system is often called the *central processing unit* or *CPU*.

Next to the built-in electronic micro-brain, the most important part of the Commodore 64 itself is the *keyboard*. Only through the keyboard can you communicate with the computer system.

Packed in the box with the Commodore 64 are several important accessories, the major one being a black box labeled *power supply*. Its function is to turn household current into the type of electricity used by the Commodore's internal circuitry, since the circuits inside the 64's plastic case would instantly turn into smoke if full line voltage were supplied to them. Inside the power supply is a transformer that reduces the 120 volts of your house current to about 5 volts and a rectifier that changes the AC current into DC.

Because only low voltages are found inside the Commodore, it's unlikely to do any damage to your health should you inadvertently drop it into the bathtub with you or compute with it in the rain. Neither is recommended by Commodore or by us. Although such activities will probably not cause you any damage, the water may have some undesirable effect on the computer, and your warranty probably won't cover it.

Making the power supply a separate unit instead of including it inside the central processing unit also gets the greatest single source of heat out and away from the delicate electronic thinking circuitry. Heat is the major cause of premature death among electronic components, so the external power supply is a lifesaver.

But a separate power supply is not without faults. For instance, when you turn off the Commodore 64, the power supply is still on and attached to your electric outlet. If you touch the black power supply box any time it's plugged in even after your computer has been turned off for hours, you'll note that the power supply is warm. That means even when your computer is switched off, the power supply is still sucking a little electricity from your wall and adding to your electric bill. Moreover, although the power supply has built-in protection against catastrophic effects from its failure, there is still a possibility that it might get overly warm due to an internal malfunction and conceivably cause a fire. The possibility is extremely minute, but it should be enough to encourage you to unplug the power supply any time you're not using your computer.

Also included in the box with your Commodore 64 are two cables and a silvery switch box. These are used to connect your computer to a *video display*—either a television set or a computer monitor.

A Critical Look at the Monitor

The keyboard and joystick are your way of communicating to your Commodore 64. The monitor is the Commodore's way of communicating back. When you are using the keyboard, the monitor repeats visually (or *echoes*) every keystroke you give your 64. That's so that you can tell the central processor has received all your commands and understood them. When a program is running, the monitor may also tell you what's going on inside the computer and, finally, the result of using a particular program.

Commodore gives you two choices for a monitor. You can either buy a "dedicated" *monitor* (a video screen used for nothing but computing), or you can use an ordinary television set.

Television sets and monitors look very much the same. Although both have *cathode ray tubes* (*CRTs*), the screens that the pictures are formed on, television sets and monitors differ in the signals they require to create their on-screen pictures. Televisions use broadcast *RF* (*radio frequency*) signals. Monitors use *video* signals.

Video signals have several separate parts, roughly corresponding to the brightness and color of the picture, and a synchronizing signal that keeps everything straight. Some

monitors use three separate wires to receive signals representing each of the three primary colors of light to be displayed on their screens. Because the primary colors of light are red, green, and blue, such monitors are called *RGB* monitors. RGB signals are capable of producing sharper pictures than any other kind of video signals, and they are the choice for exacting video displays. They also require special, expensive CRTs to display their high quality, which means RGB monitors can cost hundreds or thousands of dollars. Fortunately for you and your kids' college educations, the Commodore 64 is *not* directly compatible with RGB monitors.

If the color, brightness, and synchronizing signals are all combined in one cable, the result is a *composite video* signal. It's the same video signal used by monitors, cameras, videocassette recorders, and disc players in home video systems—and by some Commodore 64s. Because the method used to combine the signals (called the NTSC color system, for National Television Standards Committee or "Never Twice the Same Color") degrades the picture somewhat, the displays of monitors with composite video inputs cannot be as sharp as those of RGB monitors.

Somewhere between RGB and composite video is the signal used by some newer Commodore 64s—those that are shipped with monitor cables with three plugs on one end and one on the other. It uses one wire for sound and *two* wires for the picture—one for brightness and one for a combination of all three colors. Because the NTSC color system is not used, Commodore 64s using this system will produce sharper images. The VIC-1701 and 1702 color monitors handle both these special Commodore signals and composite video; most non-Commodore monitors cannot properly use video signals with the brightness and color split apart.

Broadcast television signals are often called *RF* because composite video signals are used to *modulate*, or vary, a radio frequency (RF) carrier wave, which determines the frequency or channel on which the television signal is broadcast. In other words, RF television signals go through an additional performance-degrading process that composite video signals do not. Television sets are designed to use such RF signals by employing a "tuner" that pulls in television waves that have been broadcast (either from a television station or through your local cable company) and can sort through the signals of television stations using different wavelengths or channels. Because true video monitors do not have the built-in tuners

of television sets, televisions are consequently cheaper. It doesn't make much sense, does it?

Progress is blurring the line between television sets and monitors, however. Many so-called monitors are just television sets with an extra place to plug in a wire to bypass the built-in tuner. And a new trend in television called "component video" splits the traditional television set into two units—one for the screen and one for the tuner.

To sum up: in general, a monitor can display a sharper, better picture than a television set because the composite video signals do not have to be converted into broadcast form and squeezed through a tuner as they do with a normal television. But your Commodore 64 is not picky. It will function fine with either a monitor or a television. Inside the Commodore is a tiny television transmitter (or *modulator*) that permits it to send an RF signal to any standard television set. Commodore 64s with two-plug video cables will also plug directly into most available "composite video" monitors. But note that many monitors, both monochrome and color, do not have sound capabilities. With them, your Commodore will remain mute unless you plug the audio part of the Commodore-to-monitor cable into your stereo system.

Commodore 64s with three-plug monitor cables are a breed apart. With other manufacturers' monochrome monitors, you'll have no problem other than a dangling extra plug. But the three-plug Commodores simply can't produce a color picture from a color monitor that demands a composite video input. If you want to see the Commodore's true colors, you'll have to use either the Commodore VIC-1701 or 1702 monitor or your television set.

Your choice of a monitor depends on a number of factors. Your budget may dictate that you use your television set. Even an inexpensive black-and-white set will give an adequate display for writing and running your own programs. A color television is a better choice for game playing and many of the canned programs you may buy.

If you plan to do a lot of programming, word processing, or "number crunching" like bookkeeping or financial planning with an electronic spreadsheet, a monochrome monitor is preferable, particularly if you invest in an 80-column adapter for the 64. You'll probably find that a green phosphor screen is easy on your eyes. Amber screens are even better, particularly if your office or work space is brightly lit.

Monochrome monitors are reasonably priced, from about $100 on up. Although some monochrome monitors are advertised as "high resolution," you needn't worry about how high

the resolution really is. Unless you choose to use an 80-column display adapter, the characters that the Commodore 64 displays don't require genuine high resolution. Even inexpensive monochrome monitors have specifications that claim "horizontal resolution" figures of 600 lines, which is much better than the Commodore 64 needs. Without an 80-column adapter the Commodore 64 requires only 320 lines of resolution.

Even Computers Have Paperwork—The Printer

A printer turns the electronic thoughts of the Commodore 64 into *hard copy*—ink on paper. With hard copy, you can look at what you've done without churning electricity through your computer. And you can rest assured that your work is safely out of the clutches of potential electronic malfunctions.

If you're interested in printing what you've completed as soon as possible after investing in your new computer system, the best choice is to buy one of Commodore's own printers. They all plug directly into Commodore's special (nonindustry-standard) serial port and spring to life immediately with standard Commodore instructions.

Commodore currently offers several printers. The VIC-1515 and VIC-1525 and the newer MPS-801 all use virtually the same mechanism—a clever but light-duty design that puts about 20 letters on paper every second. (Commodore claims it's 30 characters per second.) The 1525 and MPS-801 print 80 columns of characters across 9.5-inch-wide standard computer paper with tear-off edges. When the perforations are removed, the paper is the standard 8.5 inches wide. The 1515 prints only 40 columns across, the same as the standard Commodore monitor screen, and uses narrower, non-standard paper. None of these printers will conveniently handle single sheets of regular typing paper or stationery.

If you're serious about printing, you'll want the somewhat more expensive Commodore C-1526. It will handle standard-width paper 80 columns across and prints almost three times faster than its little brothers—80 characters per second claimed but closer to 60 characters per second in normal operation. The 1526 will type on individual sheets of paper.

Commodore also offers a directly compatible printer/plotter, model VIC-1520. Capable of wielding four different-colored pens, it excels at graphics. Although it prints text in four sizes, it makes the VIC-1515/1525/MPS-801 look like speed demons.

In theory, however, with the addition of the Commodore RS-232C adapter cartridge or a parallel printer adapter, nearly any printer can be operated with the Commodore 64. But some programming expertise may be necessary to put any non-Commodore printer through its proper paces. A lot of good luck and a favorable conjunction of the planets would be very helpful when undertaking the project.

Still, you might want to use a non-Commodore printer for one (or more) of the following reasons:

- Higher printing speed. Commodore claims its fastest printer runs 80 characters per second. Other manufacturers' printers that sell for under $500 boast double that speed.
- Higher-quality output. None of Commodore's machines claims to be "letter quality," meaning comparable to what a standard typewriter would produce. Inexpensive letter-quality machines are widely available.
- Quiet operation. All Commodore printers are just as noisy as most typewriters in operation. Other machines offer less irritating sounds.
- Use of wider paper. Commodore's printers only handle paper 8.5 inches wide after trimming. Other machines will print on 14-inch computer paper, which is particularly useful for electronic spreadsheets.
- Special features. Color printing, for example, is unavailable from current Commodore printers.

On the other hand, Commodore printers have special, nonstandard graphic character sets to match the special characters that the Commodore 64 puts on your screen. Printers made by other manufacturers are apt to print totally unexpected characters when given graphic commands by your Commodore 64.

Choosing a Printer

Sorting through compatible non-Commodore printers is quite a chore, because the term "computer printer" refers not to one kind of machine but several. The type that you're most likely to encounter (and the kind the Commodore offers) are called *impact* printers.

One way or another, all impact printers rely on the same printing principle as typewriters. They use the impact of a hammer of some kind to force ink from a ribbon onto paper

to make an image. Like typewriters, all impact printers have a number of desirable qualities. Their design and function is straightforward, easy to understand, and reassuringly familiar. Many can print on any paper you might have lying around your home, from onionskin to thin card stock. And they can easily make multiple copies using carbon paper or "carbonless" multicopy sets.

Impact printers reveal their typewriter heritage in another way. The hammer hitting against the ribbon and paper makes lots of noise, a sharp staccato rattle somewhat akin to eighteen tap-dancing mice.

The alternative to the impact printer is (as you may have guessed) the *nonimpact* printer. Currently two different technologies are used to make affordable (under $500) image makers—*ink-jet* printers and *thermal* printers.

Ink-jet printers spray tiny drops of ink from several nozzles onto the paper instead of hammering it on. The noise ink droplets make hitting paper is as close to the sound of silence as printers get.

Thermal printers wipe tiny pinlike styli across a specially treated paper that discolors to form an image when the styli heat it up. If you're thinking of putting a thermal printer in your home, be sure to find a reliable supply of the exact kind of paper it requires, and be aware that its images will fade over time.

Most nonimpact printers share some characteristics: simpler mechanical construction than that of the impact machines, relatively quiet operation, and the inability to make carbon copies. That makes them good neighbors but bad businessmen when you need duplicate invoices.

Your next choice in choosing a printer is between *matrix* (or *dot-matrix*) and *daisy-wheel* (sometimes called *fully formed character*) printers. All affordable daisy-wheel printers are impact machines. Matrix characters can be made by impact and thermal printheads as well as ink jets.

The characters made by a daisy-wheel printer look like the product of an old-fashioned typewriter or the printing found in magazines and books. That's because these printers carry their characters on the "petals" or spokes of a *printwheel* in much the same way as old-fashioned typewriters carried their characters on individual keys. The curves of each character are round, smooth, and continuous, and often each letter is tipped with *serifs*. The machines that make such characters are usually called *letter-quality* printers because the documents they spew forth resemble business letters. They are

the printers to choose when you want your hard copy to look important.

Matrix characters look much rougher than daisy-wheel characters because each individual character is made up of a collection of tiny dots. The name "matrix" itself refers to the dot pattern. Each dot is placed at a certain preassigned position in a matrix—essentially a set of boxes stacked into vertical columns and horizontal rows, like the pigeonholes used for sorting mail.

The quality of the characters printed by a matrix printer is primarily determined by the number of dots in the matrix. The denser the matrix (the more dots in a given area), the better the characters will look.

A 5 × 7 (horizontal by vertical) matrix (like the Commodore 1525's) is sufficient to render all the uppercase and lowercase letters of the alphabet unambiguously if inaesthetically. Such a minimal matrix is too small to let descending characters (g, j, p, q, and y) droop below the general line of type and therefore makes these characters look cramped and scrunched up. Better matrix printers, like the Commodore C-1526, use a readable but still somewhat inelegant 9 × 9 matrix that allows for descending characters.

When cost or speed counts more than quality, fully formed character printers suffer the disadvantage of being comparatively slow and generally more expensive. Home-oriented daisy-wheel machines start at about twice the price of the Commodore 1525 or MPS-801 and peck along at 12–20 characters per second. This is quick compared to your own nimble fingers but truly turtlelike when compared to matrix machines. For the same price, a matrix printer will coat paper with characters at from four to ten times this speed.

Although the letters and numbers made by matrix printers look rougher than those of fully formed character printers, matrix machines possess another advantage over the competition. The dot patterns that make up individual characters are computer controlled and can be changed by the computer without mechanical adjustments to the machine. You can switch a daisy-wheel printer from Roman to italic typeface or from pica to elite type size merely by swapping the daisy wheels themselves. The switch is even easier with matrix printers. If the printer is capable of doing what you want it to, you just send a computerized instruction to change whatever you want.

The instructions to the matrix printer need not be limited to making letters or numbers. Some matrix printers have

extra built-in character sets called "block graphics" that let you draw pictures out of building blocks of simple shapes—squares, rectangles, triangles, horizontal and vertical lines, and so on. Each of these shapes is electronically coded and recognized by the printer as if it were a letter of the alphabet. The printer merely lays down line after line of these block characters to make a picture, much like filling in each square on a piece of graph paper with different shapes. The pictures look a little chunky because the building blocks are big, just slightly under ⅛ inch across. The Commodore 1525 only hints at the character-changing capabilities of matrix machines. You can easily switch it between an uppercase and lowercase character mode and one that displays graphic symbols and uppercase characters only.

Most matrix printers, including the Commodores, even allow you to decide where to place individual dots on the printed sheet. With the appropriate instructions, the printer can draw graphs in great detail or even make pictures resembling the halftone photographs printed in newspapers. The software built into the printer allows every printable dot position to be controlled to print (black) or not to print (white). An entire image can be built up like a television picture, scanning lines several dots high (as high as the number of wires in the printhead) down the paper. Because each individual printed dot can be assigned a particular location, or "address," on the paper, this feature is often called *dot-addressable graphics, dot graphics* or *bit-image graphics*. The quality of the printed picture can be very good because the dots are very small (from 72 to 288 of them per inch). Resolution, measured in dots per inch, is the measure of dot-addressable quality and tells how sharp the printed detail can be.

Masses of Storage

Perhaps the greatest advantage of a computer is its disadvantage as well; a computer is only what its program makes it. Without the program (or software) a computer can do nothing. The big problem is getting the program into the computer. There are four ways of doing it: using the keyboard to enter programs manually, using cartridges, using tape cassettes, and using floppy diskettes.

The first two methods have difficulties. Both are one-way affairs. Without additional equipment, you can't preserve your work or your high score when you or your cat or the power

company hits the off switch. Entering programs by hand may seem simple, but when the program you want to enter is hundreds of lines long, trying to type those lines into the machine without making a mistake is one of the fastest ways to have a nervous breakdown.

Like cartridges, the Datassette (also known as the VIC-1530) and the floppy disk drive (aka the 1541) let the computer run prefabricated programs. But they can also be used to store both the program that you write yourself *and* any other data that you type into your computer. Because they let you record and preserve massive quantities of programs, text, and data, cassette and disk memory units are often called *mass storage devices*.

The easiest mass storage device to understand is the Commodore Datassette unit. It resembles a normal cassette recorder, and in essence that is exactly what it is—a portable cassette recorder in a nifty streamlined case.

So that the cassette unit can handle your computer's output instead of music or voice, your Commodore converts the data in its memory into special tones that can be recorded just like any other sound. When you want to put the information or program back into your computer's memory, you just play back the tape. Your Commodore listens to the tones and shuffles them back into data.

Cassette tapes have disadvantages, however. No matter whether music or data are recorded onto a tape, each selection must be recorded one after the other, sequentially. Without elaborate "automatic music sensing systems," the cassette player must read all the selections (cut, program, data, or file) in the exact order they were recorded, from the beginning of the tape to the end. Because computers are sensitive to the exact speed that the tape travels, you can't set the tape to whiz by in fast forward and expect its electronic ear to tell one weird squawk from another. Once your computer starts looking for a specific piece of data and must hunt through a whole cassette, you'll immediately recognize the cassette tape's big disadvantage.

A disk drive combines the best qualities of a tape recorder with the best qualities of a phonograph. Like tapes, a computer's floppy disks can be recorded, played back, and erased magnetically with utmost fidelity. Like records, floppy disks are "random access." Just as you can drop the needle down anywhere you want on a record, your computer can select data from anywhere on a floppy disk. And it can drop its "needle" (actually a moving magnetic head, like the mag-

netic head of a tape recorder) anywhere for any amount of time and then go on to the next selection. Because of their random-access abilities, disk drives are much faster than tape cassettes for the mass storage of data. In fact, disks are so superior, and so much more convenient than cassettes for mass storage, that the majority of Commodore 64 owners have bought disk drives.

In truth, any tape recorder could put the data created by the Commodore 64 onto tape, because the Commodore Datassette does not work on any exclusive or unusual principle. However, the Commodore Datassette does plug directly into the Commodore 64 and provides all the right signals and levels that the Commodore 64 needs. Although you might modify some other cassette machine to work with the Commodore 64 (if you have an extensive electronics background), the easy, expedient, and inexpensive solution is to use the standard Commodore product. It just plugs right in, conveniently drawing its operating power directly from the Commodore 64. Few cassette recorders are so much cheaper that rewiring would be worth the trouble. Of course, if you already have a cassette recorder lying around and otherwise going to waste, you can link it to your Commodore 64 with several adapters that are available. Because most of these adapters just connect the data and motor control lines but not the electric supply, you'll still have to buy batteries for your recorder.

Almost any cassette tape will work acceptably with a Commodore Datassette. However, short cassettes are generally recommended so that you're not tempted to store so much data that you have to wait for hours to find what you want. Computer cassettes are available in five- and ten-minute lengths.

If you want to use plain ordinary audio tape cassettes, choose an inexpensive ferric oxide, Type I formulation. A more exotic tape formula will be wasted on the pedestrian tastes of the Datassette and may even fail to work.

Digging into Disks

All disk drives may look tantalizingly similar, but you're probably going to have to use the Commodore 1541 Disk Drive with your Commodore 64 computer. Although nearly all home computers use the same-size floppy disks, 5.25 inches in diameter (inside the black plastic envelope), nearly all machines use a different electronic *format* to record data onto the magnetic surface of the disk.

The format is the particular way the recorded information is stored on the disk. In essence, each machine records data on a disk as if the disk were divided into separate concentric bands (called *tracks*), each of which is divided into a number of short lengths (called *sectors*). Different computers use different numbers of tracks and sectors even though they use the same kind of floppy disk. Hence, disks *written* (recorded) on one brand of machine will not *read* (play) on another brand. Sometimes disks are incompatible between two different machines from the same company.

Commodore disks are generally compatible—information stored from many Commodore PET computers can be read by the Commodore 64, for instance. However, the disk drives are not directly interchangeable between Commodore models, and only the 1541 is designed to plug into the Commodore 64. However, the earlier-model Commodore 1540 Disk Drive, originally designed for the VIC-20 computer, can be modified by your dealer to work with the Commodore 64. Other Commodore disk drives can be made to work with the 64, too, but they require an experienced technician to adapt them.

All 5.25-inch floppy disks are not the same. Disk packages tell the tale in a confusion of terms: data densities (single and double), sorts of sectors (hard and soft), and numbers of sides (single and double). Actually, all disks are equal "at birth," since all the different varieties can be punched out of a single sheet of the same raw material. The difference is how the individual kinds of disks are tested. The greater the amount or *density* of the data to be recorded onto a floppy disk, the more stringent the test made on each disk and the more disks rejected. Disks designed for greater recording densities are more expensive because more are rejected in testing. The same is true for the number of *sides*. Single-sided disks have magnetic material on both sides, but they are tested only on one side. Double-sided disks are tested on both sides, and that makes them more expensive.

Commodore 1541 Disk Drives are designed to use the least expensive diskettes: *single-sided, single-density*. That means you can use a disk rated at any density with any number of tested sides, and it should work. If you do use disks with extra sides or density, you'll simply be paying for quality that you don't need.

The one important difference in floppy disks that you must watch out for is *sectoring*—hard or soft. Most floppy disks now available are called *soft-sectored*. That means there is a single small hole punched in the disk a short distance from

the big center hole. The smaller hole is used by the disk drive as a landmark from which to find its place when looking for data on the disk. Hard-sectored disks have anywhere from 10 to 16 of these holes. A hard-sectored disk will *not* work with your Commodore 64 because the additional holes will confuse your disk drive.

Besides the 5.25-inch variety, disks 8 inches across are also readily available, but for obvious reasons they will not fit in your Commodore disk drive. The size difference is quite obvious, so you're not likely to make a mistake when you go to a store to buy disks. However, if you order disks through the mail, be sure to specify the size disks that you need.

To sum up: When buying floppy disks for your Commodore 64, ask for *5.25-inch, single-sided, single-density, soft-sectored* diskettes.

The Modem Connection

Although there are hundreds of brands of computers and dozens of languages that they speak, enough standards exist that computers can talk among themselves and share information. The primary standard that will soon become extremely familiar to you is called the American Standard Code for Information Interchange and is nearly always abbreviated ASCII (pronounced as-key). This code defines digital values assigned to each letter of the alphabet, numbers, and special commands. Nearly all communication between computers is done using the ASCII code.

However, in its immense wisdom, Commodore has chosen not to follow the ASCII code all of the time. Nevertheless, enough of the code is preserved in the Commodore 64 that it can communicate with other computers and be understood.

Many large databases (mostly mainframe computers with extremely large memories that are willing to share information with other machines for a price) can be connected to the Commodore 64 through telephone lines. They provide a wide variety of information and services, from stock quotes to games, electronic mail, shop-at-home services, and even magazines before they are published.

But connecting a computer to a telephone line is not as simple as just plugging it in. Ages ago, telephone lines were designed to carry voice signals, which are completely different from the data signals your computer is comfortable with. For computer signals to be sent over phone wires, the elec-

tronic bits of data must first be converted to voicelike frequencies and adapted to the special requirements of the telephone line. The computer on the receiving end must decode the voice signals back into computer language. The voice encoding process is called *modulating* and the decoding process is called *demodulating*. A device that can convert the signals in either direction, then, would logically be called a "modulator/demodulator," or *modem* for short. It is!

Most modems are expensive gadgets that sit under telephones and send information from one office of a company to another or let the user hook up to popular database services. They range in price from two hundred to thousands of dollars. More money usually buys more speed; premium-priced modems can send and receive data faster.

On the other hand, Commodore sells its VICModem (aka VIC-1600) at an amazingly low price—well under a hundred dollars. It is inexpensive because it sends and receives data relatively slowly and because it relies on your telephone to provide the most expensive part of the modem circuitry. Rather than replacing your telephone when it's working (as other modems do), the VICModem takes over your telephone and uses some of the circuits built into it. You save money because you don't have to pay for what's already in your telephone!

That's a clever idea, but it means the VICModem will only work with a standard-issue *modular* telephone—the kind with a handset (or receiver) that lifts off a cradle and is connected to the telephone base by a modular cord. The handset *must* have little plastic modular connectors so you can easily unplug it and plug in the VICModem.

The programming that makes the VICModem work is supplied with it on cassette. That means you'll either need to have a Datassette or else buy special (and extra-priced) disk software to make use of it. But you won't need any cables. The VICModem plugs directly into the expansion connector on the Commodore's back panel and plugs into the phone line through the coiled wire now tethering your telephone's handset.

As modems go, the VICModem is about as primitive as a stone wheel. It's inconvenient to use, since you switch cables around so much you'll feel like a snake charmer. And it was primarily designed to call distant computers, believing it's better to receive than send. But Commodore has introduced a new alternative, the AutoModem (the C-1650), which can automatically dial your phone for you or answer it, just like fancier models. It also allows several forms of data interchange,

including both *uploading* (sending data to another computer) and *downloading* (receiving information from another computer), and it includes a software disk or cassette with the program EasyCom 64. Included with the AutoModem are one free hour each on the CompuServe, Dow Jones, and Delphi database services.

Although the AutoModem does not require a modular telephone, it must be plugged into a modular jack—the little squarish hole in the wall that most standard telephones plug into. While prices vary ($80–$150), the AutoModem is often priced at only $30–$40 more than the VICModem. It's well worth the difference.

If you need to talk faster than either the VICModem or AutoModem allows (300 bits per second; just a little faster than the Commodore 1525 printer can type) or if you don't have a modular telephone, you can connect nearly any other modem to the Commodore 64 by using the Commodore RS-232 adapter. Getting the combination to work, however, may prove to be a genuine challenge. A Commodore design bug (or error) and other problems make running an RS-232 adapter at the high 1,200-bit-per-second speed used by some more expensive modems a chancy phenomenon. You may also have a difficult time finding or writing the necessary software to make it work.

Paddles and Joysticks

If you're serious about playing games, you'll need at least one *joystick* or *paddle*. Either device is a way of sending information to and communicating with your computer, which is exactly the same function performed by your computer keyboard. The difference between the keyboard and a joystick or paddle is what you do to communicate with the machine. The keyboard is designed for words and thoughts. The joystick or paddle communicates reactions. The motions you make with a joystick are more natural and quicker than typing your responses.

Paddles are special because they can communicate more than just on-and-off computer commands. Inside each paddle is something called a *potentiometer* that relays to your Commodore which way and how fast you spin its knob. Each paddle also has a "Fire!" button connected to an ordinary on-and-off switch. Not many games for the Commodore require that you use a paddle to play them.

The joystick has much in common with a keyboard. Both communicate with the computer by closing the contacts of switches. When you press down a key on the keyboard, you signal the computer by closing a switch. Instead of having over sixty switches, the joystick has only five—four to indicate direction and one for the "Fire!" button. Pushing the joystick forward, backward, left, or right closes the contact of a switch to signal the computer what you want to do, like which way you want to move *Pac-Man* on the screen.

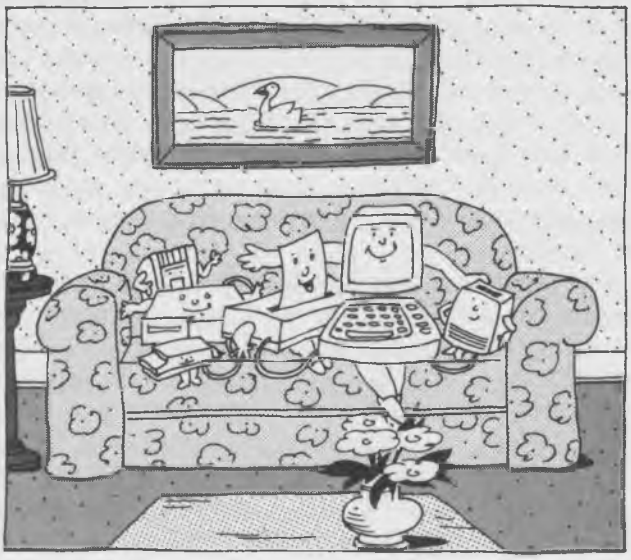
The joystick can also indicate in-between directions. When you push the stick in the direction between forward and right, you close *two* switches. That particular combination of switches may signal your computer that you've moved the joystick "northeast."

The goal of any joystick is to feel as natural and responsive as possible so that you can react and communicate your reactions to your computer as quickly as possible. The shape and feel of the joystick are very important to how well you can play games—how many space monsters you can zap or how many distant worlds you can save from total destruction. Commodore offers a modestly priced joystick that is easy to operate and effective, but it just might not match you or your hands properly for top scores. In fact, the Commodore joystick feels insubstantial to connoisseurs of such things.

Fortunately, a wide variety of outside suppliers' joysticks are completely compatible with the Commodore 64. There are dozens of joysticks designed to fit Atari (or Sears) video games, and any of them will plug directly into your computer. Wrap your hands around a few and find the one that feels best.

3

GETTING TO KNOW THE COMMODORE



Although the Commodore 64 does come to you completely assembled and safely wrapped in a plastic cocoon, Commodore doesn't tell you that a complete Commodore 64 computer *system* does require a bit of assembly (and the requisite amount of head scratching) on your part. Your job is hardly so delicate as welding microchips together under a microscope. Your mission is to get a cable or two correctly plugged in. Doing it right is just as critical to making your system work properly as the factory's assembly of the Commodore 64 central processing unit.

The job isn't tough. You can probably do it without so much as a glance at the instructions. But with some background, learning the right way to do it will give you a greater understanding of your Commodore 64 computer system and how it works.

Coping with Cables

A computer must be able to talk to and control its peripherals, and often it must be able to find out from the peripheral

exactly what is going on. Just as human beings have nerves that connect their brains to their muscles and senses, your Commodore 64 uses the current electronic equivalent of nerves. They're called wires.

Connecting peripherals to your Commodore 64 is a simple, painless procedure that's nearly impossible to do wrong—*if you follow two elementary rules*:

NEVER, NEVER . . .

1. Never, *never* plug or unplug anything when the power is on! This includes any cable or cartridge. Disobeying this rule can result in a dead computer. Accidentally crossed connections or surges of power can damage the sensitive internal circuitry of your Commodore 64.

2. Never force a connector into a jack. If it doesn't fit, it was probably designed not to fit. Many of the connectors for the 64 look the same, but the spacing of the pins or their arrangement might be minutely different. Forcing a cable can result in a broken connector or a broken jack on the computer or peripheral.

If you don't want to depend on your native instincts as to which cables go where or on your abilities to unfailingly put

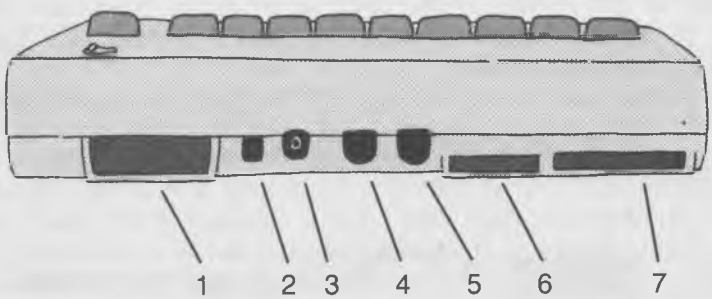


FIG. 1: The back of your Commodore. Each jack and connector has its own important function. Be sure you plug the right cable into the right slot!

1. Cartridge slot
2. Channel selector
3. RF connector (for your television set)
4. Audio/visual connector (for your monitor)
5. Serial port (for disk drive, printer, etc.)
6. Cassette interface (for Commodore Datassette)
7. User port (for RS-232 adapters, etc.)

the square peg in the round hole, let's run through more detailed instructions for connecting your Commodore 64 to various peripherals (see Fig. 1).

Connecting Television to Computer

If you've ever attached an antenna wire to your television set, you can hook up a Commodore 64. The cable you run from the 64 to your television set is a video cable, and pictures from the Commodore 64 work exactly the same way as any other television picture. You don't have to turn off the television to connect it up, but to avoid loud bursts of static, you may want to switch it off before you begin the operation. Although making this connection when your 64 is turned on won't hurt the computer, it's a good idea to turn the Commodore off, too.

The Commodore 64 is shipped with the proper cable to run between your computer and your television set. Of the two cables packed in the box, it's the one with identical connectors on either end. Each connector has a large central pin surrounded by a thin outer metal wall. They're called RCA plugs, and they're identical to the connectors used to hook up most home stereo systems.

Plugging this cable into the Commodore 64 is an easy chore. It only fits in one hole, the small, round silver jack almost in the middle of the back panel of the computer (see Fig. 2). Push it in, and you're one-third to one-half done hooking up the computer.

The other end is not as easy. If you have a recent-model

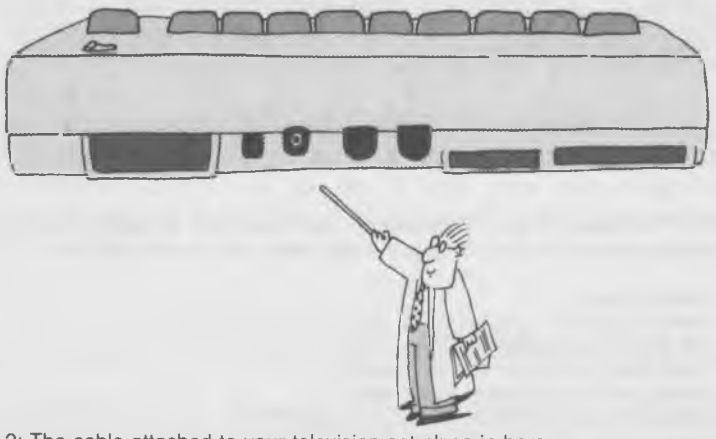


FIG. 2: The cable attached to your television set plugs in here.

television set designed to have a videocassette recorder or videodisc player plugged directly into it, merely plug the computer cable where you'd ordinarily plug the other video accessory. The Commodore 64 will now show its picture exactly as the previously connected video equipment did. Of course, you'll have to play pull and plug every time your interest in computing or video wanes and you want to change the source of the television picture.

Older televisions usually have screw terminals on the back for attaching antenna wires. Find the terminals marked "VHF" and unscrew the antenna wires. Find the little silvery switch box that came with your new computer, and screw the now-loose antenna wires to the screw terminals on the switch box. Use one wire per terminal; which one goes where doesn't matter a bit. Plug the free end of the computer cable into the jack on the box. Now attach the two spade lugs, the metal connectors at the end of the wire dangling from the switch box, to the VHF antenna terminals of your television, exactly where the antenna wires came from. Again, either wire can go to either terminal. If you want, remove the protective paper cover from the sticky tab on the switch box and firmly attach it to the back of your television set.

The function of this little box should be obvious. It's a special switch to change the input to your television from your antenna to your computer. You will have to switch between antenna and computer, depending on which you'd like to use. For obvious reasons, you cannot use both computer and antenna simultaneously.

If you have both imagination and logic, you can hook your computer to your television or video system in any of a variety of ways. Because the Commodore 64's video signals from the jack in the back are just television signals, they can be fed directly into your video recorder or routed through any other standard video accessory with little degradation. You can send the Commodore 64's picture and sound to multiple television sets by using the same kinds of "splitters" used to divide a signal from one antenna or cable among different sets in a house or apartment. Any standard antenna accessory will work with the 64's signals, including adapters to change the Commodore 64's round phone plug connections into standard cable television-type connectors (called "F" connectors by those in the know) or into flat twin-lead wire connections. With the right adapters, you can plug the 64 into any American television set. Suitable cable adapters can be bought in electronics, television, or video stores.

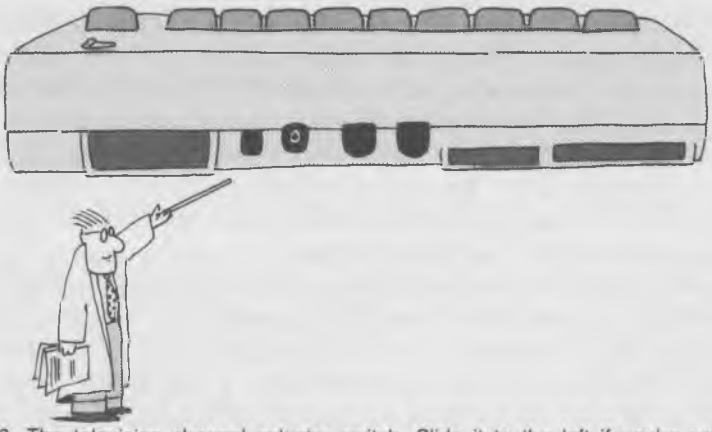


FIG. 3: The television channel selector switch. Slide it to the left if you're using channel 3, or to the right if you're using channel 4.

The Commodore 64 gives you a choice of two channels to "broadcast" over your television set: channel 3 or channel 4. To change the channel that your Commodore 64 uses, merely slide the switch next to the video cable jack. (When facing the back of the Commodore 64, channel 3 is to the left, and channel 4 is to the right; see Fig. 3.) To avoid possible interference, it's best to use whichever channel is not used by a television station in your area. If both channels 3 and 4 are free in your area, you can use either. But the best selection should be the channel furthest removed from any stations transmitting in your area.

Connecting a Monitor

The other cable supplied with the Commodore 64 is designed to connect your computer to a video monitor. It has one large black multipin connector on one end and either two or three smaller and more colorful connectors on the other.

No matter whether you have a two- or three-connector monitor cable, the big plug on the other end goes in exactly the same jack on the Commodore 64. If you look at the back of the computer, you'll note that there are two jacks side by side that look very similar and that the plug looks as if it could fit either one. The correct one is on the left (when looking at the back of the computer; see Fig. 4).

There's a dent in the silver metal ring at the end of this connector. The dent marks the top of the connector. The jack has a corresponding ridge in it to assure that all the pins line

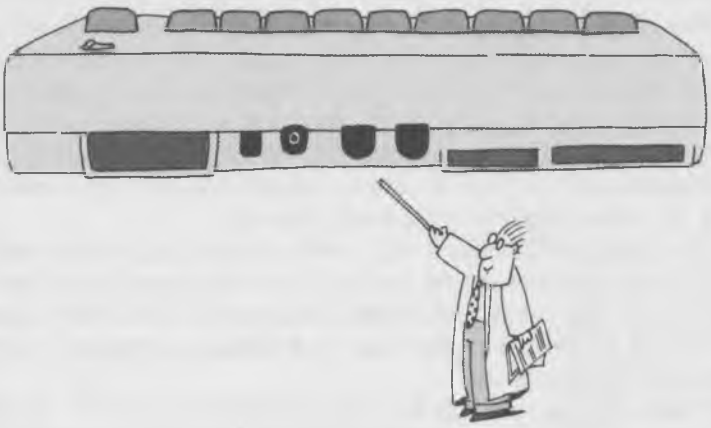


FIG. 4: The cable attached to your video monitor plugs in here.

up properly so that you don't end up playing a costly game of Kamikaze connectors. If the connector does not slide smoothly into the jack, try rotating it a little in either direction until the dent and the ridge line up.

The cables with two connectors at the other end use one for sound and the other for picture. The white one carries the sound (or audio); the red or yellow one (Commodore, in a fit of color blindness, has packed monitor cables of either color with the 64) carries the picture (video). If you use a Commodore monitor, check to make sure the switch on the back that selects between the front or rear jacks is in the "front" position. Then plug the two connectors into the jacks on the front of the set. The white connector plugs to the jack with the white center (labeled audio), and the colored connector plugs into the jack with the colored center (labeled video). In the unlikely case that you get your wires crossed, you'll know about it as soon as you turn the computer and monitor on. After the monitor has had time to warm up, you'll still have no picture, and you'll hear a buzzing sound from its speaker. Just uncross the cables, and everything will be fine.

If you use another brand of monitor, the red or yellow video connector should be plugged into the monitor's "video" or "composite" jack. Since most monitors have only one jack, you won't have too hard a time getting it right.

Most outside manufacturers' monitors do not have sound capabilities. You can, however, plug the "audio" plug from the Commodore audio/video cable into your stereo system's "aux" or "tuner" input. With the single cable provided by Commodore, that's a neat trick, because you've got all of 3

inches of wire to play with. You can either push your amp or receiver back against your monitor or buy a plain vanilla *shielded* audio extension cable at your local hi-fi emporium. While you're at it, get a Y adapter, too, so you can get sound from both sides of your stereo system. The adapter should be shielded and have one female connector and two male connectors to mate properly with most stereos.

The latest of Commodore's video connecting cables comes with three connectors on one end: one for sound, one for the black-and-white picture signal, and one for the color signal. Breaking the video signal into two parts is supposed to give better picture quality.

It also forces you to use the Commodore 1701 or 1702 monitor to get that quality. (For all practical purposes the two are identical; wherever we refer to the 1701 in the text, assume our comments apply to the 1702 as well.) With an outside supplier's monitor the best you can do is black-and-white (or -green or -amber), even with a color monitor. Plug the Commodore's *yellow* "luminance" connector into the monitor's video or composite input jack.

If you have a 1701 monitor, which Commodore seems to assume, the three connectors plug into the back. Just match the colors of the connectors to the jacks—white for audio, yellow for luminance (black-and-white information), and red for "chroma" (color information). You have five different ways of connecting it up wrong, but none of the wrong combinations can hurt your computer or your monitor. Just swap things around until you get the right picture.

The Commodore 1701's three-wire connection gives the best picture you can get from the 64. The only kind of monitor that can make a sharper color image is called an RGB monitor because it uses a separate signal for each of the primary colors of light—red, green, and blue. Alas, the Commodore 64 doesn't make the proper signals, so RGB monitors will *not* work with it.

Attaching the Datassette

The Datassette may be the easiest peripheral to attach to the Commodore 64. One end of its umbilical cable is already permanently attached to the Datassette unit itself. The other end goes to the only jack it fits into, the second (smaller) slot from the right as you face the back of the computer (see Fig. 5).

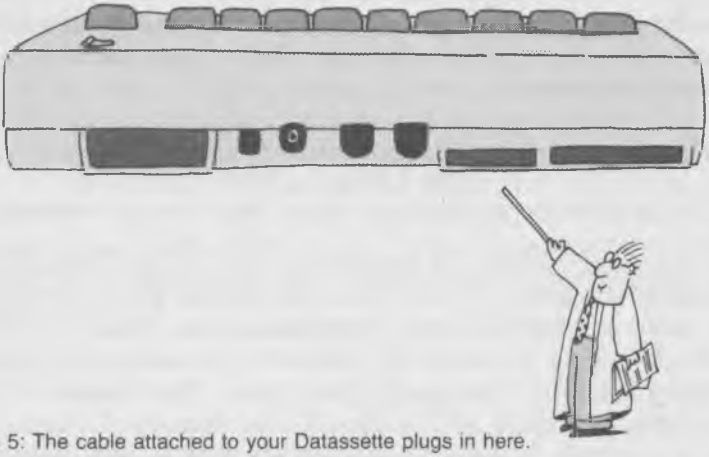


FIG. 5: The cable attached to your Datassette plugs in here.

Do not force this connector on! If you look closely at the cassette interface slot, you'll note a slot in the edge of the circuit board. You should also notice a vertical partition between the contacts in the connector. The partition must slide into the slot for the right fit. It is keyed so that the Datassette cable cannot be plugged in upside down and produce a fireworks show you probably would not enjoy. If the cable does not insert into the slot easily, merely turn the connector over.

Now you're left with an extra piece of braided wire with a lug on the end that doesn't seem to have anywhere to go. This wire is the shield of the cable running between the Datassette and the computer. It's an impenetrable wall for wandering electrical signals, preventing interference from creeping into the Datassette connections and, more importantly, preventing the signals going to and from your Datassette from being broadcast to nearby radios and televisions and causing interference. To work properly, this shield should be grounded by securely screwing it down to a computer's metal chassis. Look closely. You'll notice that the Commodore 64 has an all-plastic case with nary a grounding screw in sight.

Fortunately, the computer/Datassette system will work fine without this shield's being attached to anything. You're likely to receive interference only if you and your computer are sharing office space with a powerful X-ray machine or broadcast interference only if you try to watch television (on a separate set, naturally) and compute at the same time, with the television set sitting on top of the ungrounded cable! However, you probably ought to wrap electrical tape around

this dangling wire so that it doesn't inadvertently touch something and cause a short circuit that might damage your computer system.

The Disk Drive

The necessary cable for connecting the C-1541 Disk Drive comes in the box with the drive unit itself. The two ends of the cable are identical and interchangeable. This is a standard DIN cable (it stands for Deutsche Industrie Normen or German Industry Standard, if you care). That means if you accidentally bend a pin on one of the connectors when you use a hammer to plug it in securely (not a recommended procedure) or irreparably mash the cable in some way, you can buy a replacement where you bought your computer or at nearly any electronics or stereo store. Should fate require you to seek a replacement, make sure the arrangement of pins in the connectors of your old cable and the prospective new one is identical.

One end of this cable plugs into your Commodore 64's serial output jack, the right hand of the two similar-looking jacks on the computer's back panel (see Fig. 6). The other end plugs into *either* of the two jacks in the upper left-hand corner of the back of the disk drive.

The extra jack on the back of the disk drive allows you to connect multiple peripherals to the serial output (or "port") of your Commodore 64. You can attach up to four disk drives and two printers by *daisy-chaining* them together. The cable to the first disk drive plugs directly into the computer. Each

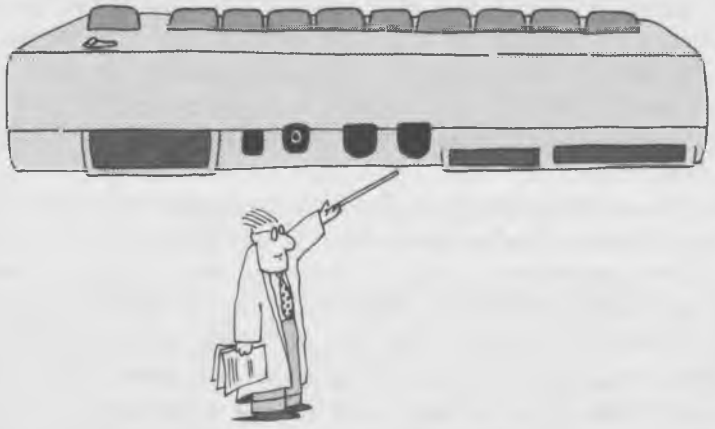


FIG. 6: The cable attached to your disk drive plugs in here.

additional device, disk drive or printer, plugs into the vacant jack on the back of the previous device.

"Daisy chain," by the way, is a term dreamed up by engineers who thought the various devices are linked together much like the nearly endless garlands or chains of wild daisies woven together by children. Alas, the engineers who frequently use the term probably have never even seen how fragile a real daisy chain can be. Perhaps a better term would simply be "chain," but flowery speech seems to be the engineers' one pleasure.

The Printer

Any official Commodore printer is connected to the 64's serial port in exactly the same way as the disk drive. If you don't plan on using a disk drive, the printer plugs directly into the Commodore 64's serial port. If you use it with one or more disk drives, connect the printer as the last device in the daisy chain, that is, to the last disk drive you've connected. The reason for making the printer the last in the chain will be obvious when you discover that it has only one connector on the back and that more devices cannot be chained from it. The new version of the 1525, the MPS-801, can be connected anywhere in the daisy chain, because it has two connectors on its rear panel, like the Commodore disk drive.

Connecting non-Commodore printers requires a little magic and exactly the right interface adapter. An interface is the connection where different parts of a computer system meet face-to-face. A wide variety of incompatible interface schemes have been devised by engineers who stay up late at night trying to find ways of making their pet systems impossible for normal human beings to understand and get working. Most non-Commodore printers use either a Centronics parallel or an RS-232 serial (or asynchronous) interface. Outside suppliers (other than Commodore) offer adapters to convert the Commodore 64's output into a standard Centronics parallel or RS-232C interface for printers that require them.

A word of caution is in order. Although any printer with one of these standard interfaces can theoretically be made or forced (at gunpoint, if necessary) to operate in conjunction with your Commodore 64, getting the machine to work in reality may require ingenuity, expertise, and the help of a professional guru. You should assure yourself that the printer/computer combination you hope to create will live in

harmony by seeing the twosome perform their duet together in your dealer's showroom. If he or she can't get them to work together, you need to find either a different dealer or a different printer.

Game Cartridges

Game cartridges (and canned application programs that come in cartridge form) are perhaps the easiest accessory to connect to the Commodore 64. They have no dangling wires and only fit one place: the cartridge slot on the far left of the back of the computer (see Fig. 7).

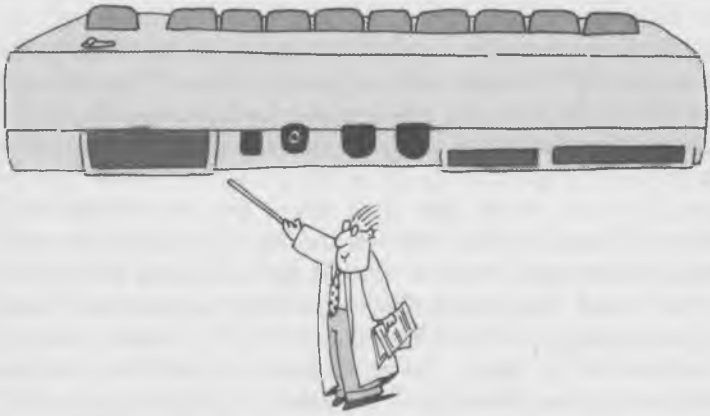


FIG. 7: Game cartridges plug into this slot.

Just slide the cartridge in firmly. The cartridge goes in right side up. Looking down at the computer, you should be able to read the label on the cartridge when you insert it. The most important thing to remember is to slide in the cartridge *before* you turn on the computer.

Modem

Either Commodore modem will slide into place as easily as a game cartridge. But its connector is in the rightmost slot (when the computer is viewed from the rear; see Fig. 8). A modem, too, should only be inserted when the power is off and should go in so that you can read its name as you look down at the computer.

The VICModem's other connection is the only one that breaks the "power must be off" rule. The wire that normally

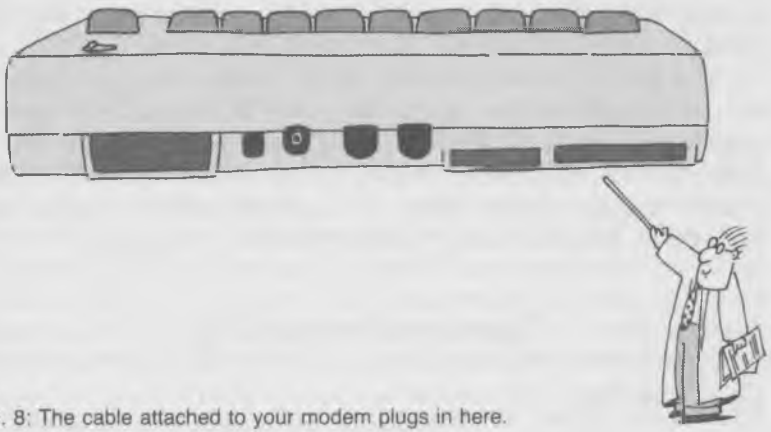


FIG. 8: The cable attached to your modem plugs in here.

leads to your telephone's handset gets plugged into the modem only *after* you dial the number you want and hear the connection being made. Plugging this wire into the modem just to see how it goes in won't hurt anything, but if you forget to plug it back into your telephone, the next call you make may be surprisingly quiet.

The AutoModem will plug into any phone jack that accepts a standard modular telephone. If, like most people, you want to use your AutoModem in conjunction with your telephone rather than in lieu of it, get a double-jack adapter anywhere telephones are sold. It plugs into a single-phone jack and gives you two to use as you please.

Paddles, Joysticks, and Light Pens

The 64 provides two jacks for plugging in either paddles, joysticks, or a light pen. You won't find them on the back of the machine but on the right side, just in front of the power switch. They are trapezoidal in shape, so the plugs can only be inserted in one way—the right way.

If you want to use only one joystick, set of paddles, or a light pen, use the jack toward the front unless the game or program you want to use advises otherwise on its package or instruction sheet. The front jack has the priority position, and a light pen will only work in this jack.

The Power Supply

The last thing you should connect is the power supply. Be sure to use only the Commodore 64 power supply that came

in the box with your computer. Other power supplies may produce different voltages that can damage your 64.

First plug its DIN-style connector into the rearmost jack on the right side of the Commodore 64. Then plug the power cord into a wall outlet. Inspect all the other cables you've installed to make sure the connections are secure. When you're sure that everything is connected properly, then, and only then, are you ready to begin turning things on.

Turn-on Procedure

Unlike the other electronic miracles you may be used to working and playing with, your computer is very particular about the way you supply it and its various peripherals with electricity. Computer equipment is apt to create very strange signals when it is turned on—signals that can wreak havoc with other peripherals connected to the system. Some computers and peripherals are designed to explore the circuitry they're connected to at the moment of turn-on. They memorize what they find, and if something else is added later, they may not know it's there.

The unit with the most elaborate turn-on procedure is the Commodore 64 itself. Consequently, it should always be the *last* unit of the system you turn on and the *first* to be switched off.

The order of turn-on for the rest of the peripherals is not as critical. You should establish a regular procedure: probably turn the monitor on first so that it has time to warm up, then switch on the disk drive(s), then the printer, and finally the computer itself.

If you violate your own procedure, you'll probably discover that the various peripherals are hardier than you think. You can switch your monitor on and off if you feel so moved, and you won't lose a bit of data. But you'll avoid subtle surprises if you stick to a regular procedure.

Before you turn on the printer, be sure you have it loaded with paper, and be sure to have the perforations between the individual sheets of continuous paper lined up with the top of the printhead. Most printers, including the Commodore machines, memorize the position of the paper at the time power is turned on and assume it's lined up properly for purposes of "form feed" commands, by which the machine advances to the top of the next page.

Similarly, be certain that no disk is in your disk drive

when you turn the power either to it or to the computer on or off. When you turn on the power for either the disk drive or the computer, it takes a while for the electronic brain inside to "wake up" and gain its composure. Likewise, when you switch either the computer or the disk drive off, it takes a while for the electrical signals inside to die down even though the microprocessor brain inside immediately stops working. The net effect in either case is that uncontrolled signals or commands might damage the information you have stored on a disk inside the machine. The best preventive measure is to remove disks from the drives *before turning any peripheral or the computer off.* By the same token, don't insert a disk into the drive until after the last component, the computer itself, is turned on and ready to operate.

TURN-ON CHECKLIST:

1. Make all connections; insert game cartridge, modem, etc., if desired.
2. Load and line up paper in printer.
3. Make sure no disk is in disk drive and no cassette is in Datassette.
4. Turn on monitor.
5. Turn on other peripherals.
6. Turn on computer.
7. Put disk in disk drive or cassette in Datassette.

TURN-OFF CHECKLIST:

1. Save program or text in computer's memory to disk or cassette (if desired).
2. Remove disk from disk drive and cassette from Datassette.
3. Turn off all peripherals, including monitor.
4. Turn off computer.
5. Unplug computer power supply from outlet (if you don't plan to return to your computer right away).

4

TAMING THE DREADED READY

You've got your Commodore 64 computer hooked up with nary a glance at the instruction book. You're already feeling like a pro! Just a quick flick of the switch, and in a few seconds the monitor screen lights up (see Box 3). Then up "it" pops like some evil spirit in the night. Staring at you is the *Dreaded Ready*!

Sure, it looks innocent enough. There's nothing on the screen but a banner announcing the name of the machine, a hopeful note about the amount of memory at your beck and call, and that helpful-sounding but ominous READY glaring at you with a square of light flashing impatiently below it. If you are a normal human being working with a computer for the first time, your emotions will race through the following sequence: pride, relief, then helplessness.

Pride first. You've made your first venture into the unknown world of computing, and you've met success. You've put together a working computer system with your own two hands.

Then relief. This first on-screen message reassures you that you have a completely functional computer. Every time you flick the power switch on, the Commodore 64 runs through

a repertoire of internal examinations to make sure everything within it is working well. The message confirms that it has passed its preliminary examinations and that your computer is going to make an effort to work properly. You can't yet give it an absolutely clean bill of health, because there might be some subtle manufacturing defects hiding within its circuitry. But if you get this screen message, odds are that everything is in good working order.

If the programs you encounter most often are television situation comedies, your relief will quickly lapse into helplessness. You've got the computer turned on and running. Now the big question is *What Are You Going to Do with It?* The Dreaded Ready hints that your machine is ready to do nearly anything you want it to do. But first you have to figure out what to tell it to do and how to do it.

BOX 3: THE TURN-ON BLUES

Before you try anything with a fledgling Commodore 64, you should make a quick check that the factory didn't make any mistakes and that all is well with your new computer.

A few seconds after you switch your Commodore 64 system on, your television or monitor screen should turn dark blue in the center with a lighter blue border and several rows of light blue letters across the top.

If your screen remains gray or filled with static, check the red pilot light on the Commodore 64. If it's not lit, check and make sure that the black power supply transformer is plugged into the wall outlet and into the computer and that the Commodore's power switch is turned on.

If you're using your television set for a monitor and the Commodore's pilot light is on and your screen is still gray or filled with static, make sure your Commodore and television are both switched to the same station. Be sure the switch box attached to your television's antenna terminals is set for the computer. Adjust your television set's fine tuning control for the best colors and least static.

If the on-screen image is shimmery, shaky, or otherwise unacceptable and no adjustment of the controls on your television set or monitor can correct the problem, you might just have a defective computer. Try it with another television or return it to your dealer.

You've learned the first big lesson about computers: without software, hardware can do nothing. Both are needed to make a computer system work. More important, you've learned that a computer does nothing unless you tell it to do something. Learning the art of programming merely means learning to communicate with the computer—learning how to give it the instructions it needs to do the job you have in mind.

Let's approach the computer problem with the logic of a normal human being. You want to run a program, right? Obviously, what you need to do is press the button on the keyboard that makes the machine run a program. Search the keyboard and find the key marked RUN/STOP. If you want to run a program, the likely button to press would be RUN. So press it.

Nothing happens. The Dreaded Ready is still staring at you. No change. No hope.

You've run smack into lesson two. Computers can perform so many different functions and commands that giving each one a separate push button would produce a control panel that would make a NASA space center look simple. And even with every possible command on a separate key, you'd still need a typewriter keyboard to enter data. The first computer programmers got the brilliant idea of using the same keyboard for typing data as for controlling the computer. In fact, if information and commands are typed the same way, they can be stored the same way, simplifying the computer system. To a computer, everything is data.

Instead of the RUN/STOP key, try typing the three letters RUN. Something happened! You've got the word RUN on the screen. Well, that's a minor accomplishment (although you could have typed anything), but the computer has done nothing. That's because the computer doesn't know you're done typing.

You have to let the machine know you've finished typing and that you want it to do something. Most computers recognize one particular signal to indicate that you've finished typing a command: a press of the RETURN key. The computer will do nothing until you press RETURN.

That's actually useful. If you make a mistake, you can go back and correct it before sending it to the computer. For instance, if you make a mistake in typing RUN and type RUM instead, you can correct the error by using the INST/DEL key to erase the offending letter. When you have your RUN displayed on the screen properly, just press RETURN to signal the computer to start running.

What happened? You've got another Dreaded Ready! They're

multiplying! What can you do? Are they going to take over? Why didn't the machine run a program?

Possibly the Dreaded Readies *will* take over. The machine is still telling you that it's ready to run a program. And you've told it to run a program. But because you haven't yet given it a program, the machine has nothing to run. After you typed in RUN, it went looking for a program, found none, and got ready to receive your next instruction. Does the situation look hopeless? Will you spend the next twelve years learning to program your $200 toy?

No. You could slip a game cartridge in and start playing immediately. (Warning! Turn off the computer before inserting the cartridge or connecting any cables!) But that's not why you bought a computer, is it? If you wanted to play games, you could have just bought a video game machine. What you want to do is learn to master this computer.

Can you do anything with a computer without putting in your own program first? Yes—if someone else writes a program for you. A game cartridge works without your doing any programming at all, because the game program is already written and stored inside the cartridge. All you have to do to feed the program into your computer is plug the cartridge in.

The Commodore Command Center

There are many ways people can communicate with computers. Joysticks and light pens convey a limited amount of information. Someday your home computer may even recognize your voice and your handwriting. But for now the computer and you share one common group of symbols: letters, numbers, and punctuation. That's why the keyboard is your command center for the Commodore 64. It allows you to tell your computer exactly what you want it to do.

Even with a computer, communication must be a two-way street. You send commands and instructions through the keyboard. The computer replies by lighting up the monitor screen, showing it has recognized the symbols you've typed.

Computer keyboards resemble typewriter keyboards, but they're just different enough to make using them inconvenient if you're a good typist. And if you're not a good typist, any typewriter-style keyboard is inconvenient. The Commodore 64 keyboard continues this worthy computer tradition.

The most familiar part of the 64's keyboard is its arrange-

ment of the letters of the alphabet, which follow the familiar QWERTY sequence of most typewriters. That's good. If you don't know how to type, the Commodore 64 will help teach you where the letters are.

The chief difference between typing on a computer and typing on a typewriter is that the computer uses a video screen instead of paper to show you your work. Computers mark the place where you are typing with a special symbol on the screen called a *cursor*. On the Commodore 64 the cursor is usually a flashing rectangle. On other computers (and with some programs for the 64) the cursor might be a nonflashing rectangle or just a thin line that may or may not flash. The cursor appears exactly where the next character you type will appear on the screen.

A Critical Look at the Keyboard

Computers and typewriters are different in their keyboards, too. The most significant keyboard differences appear at the left and right sides of the rows of letters. On the right, instead of the mishmash of normal punctuation, the Commodore 64 gives you a plethora of strange symbols, like a key devoted only to the lowly asterisk (*). Although a mystery to the new owner, this one little key can become very important in helping you save keystrokes once you begin to use the BASIC programming language. Not only is it a footnote symbol, but it indicates multiplication and is used as a "wild card" in pulling files from the disk drive. You'll also discover the "at" symbol (@) a real help in working with the disk drive. And the separate plus and minus signs will make your life easier as well.

These keys are easy to use because they keep you from having to press two keys at once—the shift key and the appropriate number key. They will save you a great deal of time and even more frustration when you get into really serious programming. Compliments to Commodore for making life just a little bit easier.

And a curse hailed Commodore's way, too. Stuck above the number 4 you'll see the dollar sign. No big deal until you encounter BASIC and discover that the lowly $ is probably the most-used symbol in most programs. The quotation mark is equally popular in programs, and it's stuffed right up there next to the $. Why are they up there when the rare pounds (Sterling) sign is granted its own key? Soon you will join

the ranks and curse Commodore for its befuddling lack of insight.

Finally, the biggest difference between the Commodore keyboard and most typewriter keyboards is the computer's large collection of special function keys. The unique *COMMODORE* key at the lower left and *F1* at the upper right are two examples. More than merely saving keystrokes, these extra keys multiply the number of symbols you can put on your video screen, thereby giving you immense control over the computer.

The Shifty Keyboard

The familiar carryover from the world of typing is the *SHIFT* key. As you might expect, it allows you to shift between lowercase letters and capitals, but not in the way you might expect. Turn the Commodore on, and it will make only capitals. When you hold down SHIFT while you type, you'll fill the screen with strange symbols that resemble the inscriptions on Egyptian tombs. Surprise! The symbol made by each letter key when shifted is indicated on the right front side of the key.

Using the SHIFT key allows most of the other keys to give you two different characters—one shifted and one unshifted. You get double the power from the same-sized keyboard.

Tucked at the lower left of the Commodore keyboard is a *SHIFT LOCK* key. Press it, and it locks down. In the down position the SHIFT LOCK makes the Commodore think that you're holding down the SHIFT key all the time. But there's an important difference. SHIFT LOCK *does not* shift the number and special function keys.

The Commodore doesn't stop with just doubling the amount of symbols you can put on the screen. The shift key actually works in two "modes." One mode, the one we've already seen, shifts between capitals and hieroglyphics; the other shifts between capitals and lowercase just like a typewriter. To shift between modes—from caps and symbols to caps and lowercase—just hold down the SHIFT key and press the special COMMODORE key.

But that's not all the COMMODORE key does. If you use it by itself as another shift key, it allows you to use the symbols represented on the left side of each key front.

So far, that's four symbols from each key. But that's still not all! Another key in the same group, *CTRL* (for ConTRoL),

allows you to use almost every one of the other keys for another purpose—a purpose different from the other four shifts and something unlike anything you've ever run into with your typewriter. Instead of shifting between different symbols on the screen, CTRL lets you send commands directly to your computer. Holding down CTRL and pressing a number 1 through 8, for instance, will tell your Commodore to change the screen color. CTRL and letters of the alphabet create other codes with special meaning to your computer. That's five functions from each key and a lot of power at your fingertips!

But the variety of functions each Commodore key can handle doesn't stop there. Programs can cause keys to have altogether different functions. The commands some keys issue can be blocked or changed to anything you or the programmer pleases. Don't be surprised when you run a canned program and none of the keys work the way you think they will. It's just part of the Commodore's versatility.

Dedicating the Keys

A number of the 64's keys have particular functions to make your computing life easier once you've discovered the black secrets of programming. These are called *dedicated* because they're dedicated to doing a particular thing.

Try the *CLR/HOME* key, and you'll quickly discover the logic behind the dual names atop the Commodore 64's dedicated keys. Unshifted, the CLR/HOME key sends the cursor instantly to its so-called *home* position: the upper left corner of the screen. But holding down SHIFT and pressing CLR/HOME at the same time will erase (or CLeaR) everything off the screen before it takes the cursor back home.

RUN/STOP lets you stop a program from running at any time merely by pressing it down all by itself. After what you've learned about the CLR/HOME key, you might expect that to start a program running again, you'd just press SHIFT and the RUN/STOP at the same time. After all, isn't that why the word RUN is above STOP?

Wrong. The Commodore masters of illogic have struck again. Shift the RUN/STOP key, and the result is a command to the computer, all right, but one that instructs the machine to "load" a program from cassette tape. If you can't figure out why the key wasn't simply labeled LOAD/STOP, you're not alone.

The other dedicated function keys are refreshingly more logical. Two of them let you move the cursor around the screen. They're labeled *CRSR*, an abbreviation for CuRSoR, and marked with arrows showing which way they send the cursor along. Unshifted, the keys move the cursor in the direction of the lower arrow. Shifting each key sends the cursor in the direction of the upper arrow. Holding either key down makes the cursor-moving command automatically repeat itself.

Two keys have nothing on them but a single arrow apiece: the "back arrow" on the left and the "up arrow" on the right. What they do may surprise you. Pressing the back arrow doesn't move the cursor at all. It simply puts a little arrow on the screen. The up arrow not only puts a small upward-pointing arrow on the screen but also warns the Commodore that any number following it is an exponent. That is, typing "3," "up arrow," and "2" tells the Commodore you want to enter the formula "3^2." Shifting the up arrow is entirely different; that lets the Commodore reach into its memory banks and pull out the value of pi, 3.1416.

INST/DEL stands for INSerT and DELete. Pressing down this key unshifted will delete or erase any character appearing immediately to the left of the cursor. Pressing this key and SHIFT simultaneously will insert a blank space immediately to the left of the cursor.

RESTORE by itself doesn't do anything. But when pressed in combination with RUN/STOP, it's very powerful—and dangerous, if pressed unintentionally. It immediately eradicates any and all commands you've given your computer and restores (or resets) the machine to the settings it would have immediately after being turned on. But (lucky for you) it does not erase the program in the Commodore's memory.

On other computers, keys that perform similar functions are called *reset* keys or *warm boot* keys. In fact, to reset some machines, you actually have to turn them off. The RESTORE function on the Commodore 64, however, is much more advanced, because it keeps intact whatever program you may have been working on. That means when you reset your Commodore 64, you will *not* destroy a program you've been typing into the machine for hours. That's very, very good and something that you will praise to the stars when you are a grizzled old programmer. Remember: if you run into major problems, just press the combination of RUN/STOP and RESTORE. You'll instantly return to a clean blue screen with your program all safe and sound.

To the far right of the keyboard are those four function keys. Since they can be used in their shifted or unshifted states, they can control a grand total of eight different functions. Each can be programmed to represent any combination of keystrokes that you want. You could even program one to type out your name every time you strike it or to change your screen to blue with pink stripes. Once you've mastered the fine art of 64 programming, you'll be in command of what each one does.

The Colorful Commodore

Every computer, including the Commodore 64, has programming instructions built into it. The machine couldn't even put the Dreaded Ready on the screen without a simple program to do so tucked away in its permanent read-only memory (ROM). Just typing in letters and having them appear on the screen requires a program built into the computer's ROM. These internal programs run so smoothly all by themselves that you never even notice they're there.

Some of the instructions stored in ROM let you make the computer do things without writing programs. You can give the machine simple instructions that cause the machine to run one of its own built-in programs.

Probably the simplest thing you can get the Commodore 64 to do is change the color of the letters on the screen. Perhaps the blue-on-blue of your color monitor's screen just doesn't go with your wallpaper. Maybe the characters on your monochrome display don't stand out enough. In any case, just by pressing two buttons, you can change the character of the on-screen characters.

Put your finger on the CTRL (or Control) button and hold it down. While still holding it down, press the number *1* key. Note that the cursor has changed color. Type in RUN and press RETURN and observe the difference in the new Dreaded Ready that appears. You've already taught it a trick!

You don't have to be particularly clever to note that the 1 key that turned the characters black has the legend BLK on its front and that the rest of the numbers up to 8 also have colors there. Hold CTRL and press them, and you'll wander through half the colors available for your on-screen characters.

You can get the other half by holding down the COMMODORE key (the leftmost key nearest the front of the keyboard) and the same numbers, 1–8. Note that anything that you or

the computer types *after* you press the key comes up in the new color, but the previously typed characters remain the same. Note that with a color display, one color of characters becomes invisible. This happens because the character color and the background color are identical, producing an effect something like the proverbial polar bear in a blizzard. On monochrome displays, many character and background color combinations may yield invisibility.

Computer as Character Generator

With the little you've already learned about your Commodore 64, you can do something useful—identify videotapes and make titles for your own vîdeo creations using the 64 as a character generator. (You can also use the computer to leave messages and notes. This is clever and an attention getter, but paper is cheaper!)

All you need to do is connect the output of the computer to the antenna terminals of your videocassette recorder and type away on the screen. You can change colors to suit your desires. You can even add graphics by using the SHIFT key. Here are a few hints to help you out:

1. The graphic symbols shown on the right side of the front of each key can be typed by holding down SHIFT and pressing the appropriate character key at the same time.

2. The graphic symbols shown on the left side of the front of each key can be typed by holding down the COMMODORE key and pressing the appropriate character key at the same time.

3. If you want capital and lowercase letters rather than graphics, shift modes by pressing the COMMODORE key and the SHIFT keys simultaneously. The SHIFT graphics will change into letters when you do this, but the COMMODORE graphics will not.

4. Move the cursor around with the CRSR keys on the left front of the keyboard or press CLR/HOME to put the cursor in the upper left corner of the screen. Do *not* press the RETURN key—unless you want to learn about error messages.

5. If you make a mistake, move the cursor over the incorrect characters and just type over them. The new characters automatically erase the old ones. Or you can eliminate the character to the left of the cursor by typing INST/DEL. Holding down the SHIFT key while typing INST/DEL will push in an extra blank space that you can later type over.

6. To erase all of what you've typed without changing your color settings, press SHIFT and CLR/HOME at the same time.

7. Once you've got everything just right, record it for posterity exactly the same way you would record a television show.

Making Memories

The characters aren't the only colored parts of your monitor screen. The area behind the characters is called the *background*; the area surrounding the background is known as the *border*. Their colors are also under your direct control, but only if you know the secret of how to tell your computer what color to make them.

The Commodore 64 uses its memory to remember what colors it should put on the screen. As you know, that memory is divided into *bytes*. In fact, each byte is assigned a number so that the computer can tell it apart from the others. Each of these numbers assigned to each byte is called an *address*. The number or address actually is a representation of a "memory location." The Commodore 64 can keep track of 64K addresses at a time.

The Commodore 64 uses one memory location for the background screen color and another for the border screen color: address numbers 53281 and 53280, respectively. (Note: Many computers are confused when you put commas in numbers, so don't.) These addresses are *machine specific,* meaning they apply *only* to one specific machine. Such addresses can be different even for closely related computers. The screen memory locations for the 64's cousin, the VIC-20, are completely different.

Stored in each screen memory location (at each of the two addresses) is a number from 0 through 15 that represents one of the sixteen colors the Commodore 64 can display. To change a color displayed on the screen, all you need to do is change the number at the appropriate address.

How? The command to change the contents of a memory location is POKE. Try typing "POKE" (the word, not the quotation marks) on your Commodore 64. Remember, when you turn your machine on, it automatically sets itself up to make the letter keys produce capital letters without your pressing the SHIFT key. If you do press SHIFT when you type, you'll put characters on the screen that neither you nor your computer will understand! Shifting a number key,

however, will still display the symbol printed above the number on top of the key.

Okay, now it says POKE on the screen. Next you must tell the computer what memory address you want it to change. So type "53280" (again, *don't* type the quotation marks), the address for the border color. If you want to, you can leave a space between "POKE" and "53280," but you don't have to. The Commodore can sort things out either way.

Finally, you must indicate what color you want the border to change to. Red will give the most dramatic change. Because the code number for red is 2 (see Table 1), 2 is the appropriate number to type.

But wait. If you type the 2 immediately after the address 53280, you'd get 532802—a number the Commodore would mistake for a different address, with undesirable consequences. To separate the contents from the address, the Commodore uses a comma (,), and so should you. That's why you shouldn't put commas in numbers. The Commodore uses commas to

TABLE 1: POKE VALUES FOR CHANGING SCREEN COLORS

For Color	*Make X =*
Black	0
White	1
Red	2
Cyan (blue-green)	3
Purple	4
Green	5
Blue	6
Yellow	7
Orange	8
Brown	9
Light red	10
Dark gray	11
Medium gray	12
Light green	13
Light blue	14
Pale gray	15

Command form: To change background, type POKE 53281,X
To change the border, type POKE 53280,X

separate *two different* numbers. The proper instruction typed on your screen should look like this:

```
POKE 53280,2
```

Now press RETURN so your Commodore knows you're finished with your command.

Presto! A red border. With the same simple instruction you can make the background and border any color combination you want. The code numbers to be POKEd into the different memory locations for the various available colors are given in Table 1.

With some color combinations, the letters you type can be hard to read, if not unintelligible. The problem is in your monitor (and everyone else's). The unreadable letters are a result of the way inexpensive color monitors are made. What can be done about the nonworking color combinations? Don't use them.

Using the POKE instruction, you can change the contents of any memory location inside the Commodore 64. Most of the time you won't notice a change, at least not right away. But if you put a number somewhere the machine expects something else, you can cause undesirable or disastrous consequences, like making the computer jam and refuse to do anything.

If you do that, you have two escape routes. Pressing the RUN/STOP key while you hold down the RESTORE key will wipe out any changes you've POKEd into your Commodore's memory and take you back to the very first screen that you saw when you turned on the machine. But if that doesn't work, flicking off the power to the computer for a couple of seconds, then turning it back on, will wipe everything out of the computer's memory, including the evil POKE. You cannot do permanent damage to your computer hardware with software commands, but mistakes can cause temporary and painful program problems (see Box 4).

Examining Errors

Before you go any further, you should be introduced to someone who will become an old friend (if you haven't already met him by accident). To introduce yourself and rouse him up, merely type your name followed by a RETURN. Immediately you should see

```
? SYNTAX ERROR
```

emblazoned on the screen.

Any time you type in an instruction that the computer doesn't recognize, it will taunt you with those same infamous words. "SYNTAX ERROR" is formally called an *error message*. It's your computer's way of saying, "Huh? I don't understand what you want. Are you sure that's what you mean to say? Until you talk to me the way I want you to, I'm not going to do anything."

It would be easy to blame this obscure verbiage on some electrical engineer or programmer who found it difficult to communicate in English and so came up with this term. Actually, syntax is the structural form of grammar, and this message warns you that the form of your command is in error.

When you rouse this error message, just look back up the screen and try to see what you did wrong. When you typed your name, the reason you got the message was that the BASIC language in your computer expects the first word of any line to be either a number or one of the few dozen words it recognizes as a command (like POKE). Your name, in all

BOX 4: TRICKS WITH POKES

Okay, so you don't believe me. You don't think that POKEs are too dangerous to type indiscriminately. If you need proof, try this:

```
POKE 56322,1
```

Guess what! Nothing works. You've told the Commodore to stop listening to the keyboard and turn its ear to the joystick. Nothing you do with the keyboard will change anything. However, if you turn the Commodore off and back on, everything will revert back to normal.

Useless? No. It's actually a neat command to sneak into a program. You can make your program run forever and keep its inner workings secret. No one will be able to see how you wrote it, because to do that, they would have to stop the program from running. And the only way to stop the program is to turn off the computer, which, of course, destroys the program!

likelihood, is not one of the authorized BASIC command words.

Incorrect syntax is not the only error you can make, but it is probably the most common. Most error messages are preceded by a question mark (?), equivalent to a human "Huh?" If you run into one before you're properly introduced to it, just remember to check and make sure that you typed in precisely what you intended. Even the slightest difference can result in a strange, obscure, and probably incomprehensible error message. Try typing POAK instead of POKE if you're a doubter.

Poking Around in Memory

There are a total of 65,536 different addresses you can "poke" into inside the Commodore 64's memory. Trying each, one by one, would not only take a long time but would be generally fruitless. Monkeying with most of them will give few discernible results. However, there are two large blocks of memory where you can see effects immediately—the addresses where the Commodore 64 stores the characters that are to be displayed on the screen. One block of memory holds a code number identifying the shape of each character to be displayed, and the other block stores the color of that character.

The Commodore 64 can display 1,000 characters on its screen—40 columns across by 25 rows down. Hence, two blocks of 1,000 addresses each are necessary to store the shape and color information. For reasons unknown but to them, the engineers at Commodore chose memory addresses from 1024 to 2023 to store character shapes and from 55296 to 56295 to store the color information. These memory locations are also *machine specific* and apply only to the Commodore 64.

Starting with the first address in the upper left (or home) position on the screen, each character position from left to right is assigned the next higher address. Each subsequent row is also numbered from left to right. Thus, the address to change the color of the upper left character is 55296; the top right position, 39 further along, would be 55335; the leftmost position on the next row would be 55336; and so on. A chart of the address assigned to each location is called a *memory map*.

Colors are coded at each address with the same numbers you used for POKEing the background and border hues. You can change the color of a single character that has already

been printed by POKEing a new value into its color address. For instance, to change the first asterisk of the first display you get upon turning on your computer, you could type:

```
POKE 55340,0
```

because the asterisk is the fifth character over in the second row (remember, start counting at the first address: 55296). To try it out, turn your Commodore 64 off and on again to get the initial message. Then type in the command in response to the Dreaded Ready.

It's easy to figure any position on the screen from a row and column number. First you have to think like computer folks and start counting at zero. Think of the first row on the screen as number 0 and the others as being numbered up to 24. Start at zero for the columns, too, which go from 0 to 39. Multiply the *row number* by 40, then add the *column number,* and finally add the *starting address,* the first number of the 1,000 addresses covered by the memory map. Often in programs you'll find this formula written out as X + (Y * 40) + Z, where *X* is the column number (or the x-axis coordinate), *Y* is the row number (or the y-axis coordinate), and *Z* is the starting address.

If you want to change the color of every character that is already displayed, you could try POKEing each address individually. Or you could try typing:

```
FOR I = 55296 TO 56295: POKE I,0: NEXT
```

You'll see each character that is displayed turn black in turn as the computer POKEs the zero (0) code into each character color memory location. If characters don't magically appear, see Box 5.

The Commodore uses a number called a *screen code* for each shape that can be displayed. The code for a diamond is 90. To put a diamond in the middle of the screen, try this:

```
POKE 1500,90
```

To fill the screen with diamonds, type:

```
FOR J = 1024 TO 2023: POKE J,90: NEXT
```

Note: some Commodore 64s are different, as Box 5 details. If neither of the diamond-making commands worked, it was probably because you have a new Commodore 64 and you

BOX 5: HOW NEW IS YOUR COMMODORE 64?

The examples to change the character colors at each address and put characters at each address were given in that specific order in the text because some Commodore 64s work differently from others. If you poked a character into an older 64, odds (15 in 16) are it would immediately appear on the screen in white. With newer machines, just POKEing a character may give no visible results. Although the character will be tucked happily away in its new home, it won't show. The reason is that each time a new Commodore clears the screen, it automatically sets the numbers stored in the color memory locations to be the same as the number stored in the background color address (6, at the time of turn-on). Because your POKEd characters are exactly the same color as the background, you can't see them!

To make POKEd characters visible on the latest Commodore 64s, you have to POKE in both the character and a color for it. The examples in the text had you POKE in a new color at every address before you POKEd any characters.

To see if you have an "old" or "new" Commodore 64, turn it off and back on. Then type:

```
POKE 1523,83
```

If a small heart does not appear near the middle of your screen, you've got a new one. If you do have a new Commodore and you want to make the heart visible, you'll have to type POKE 55815,10 to change its color.

didn't type the previous command (FOR I = 55296 TO 56295: POKE I,0: NEXT) that made all the on-screen characters black. You can make the diamonds visible by typing that command now.

Now to reveal the secret. The magical, mysterious command that caused a thousand POKEs is called a *FOR/NEXT loop* and is one of the most powerful and useful commands in the BASIC language. Here's how it works.

The computer treats the letter J as a *variable*—a number that can change. To start, J equals the number on the other

side of the equals sign. From then on, whenever the computer sees the J, it automatically substitutes J's current numerical value. If J equals 1024, the statement POKE J,90 is exactly the same as the statement POKE 1024,90.

When the computer finds the NEXT, it "loops" back to the FOR and increases the value of J by 1. Then it continues forward again, substituting the new value of J wherever appropriate.

The computer keeps looping back until J is greater than the number that appears after the TO. Then it jumps to whatever command follows the NEXT. In this case, there are no more commands.

In the Commodore version of BASIC, a colon (:) lets you put more than one command on a single line. The FOR/NEXT loop you typed is actually three separate commands. If you use any other character to break the command line into three (say, a period or comma), you'll say hello to the ? SYNTAX ERROR message again.

The variable you typed, the J, was named entirely arbitrarily. It could have been, L, M, N, O, or MUMBLE and the result would have been the same. In the version of BASIC used by the Commodore 64, a variable can be almost any letter, two-letter combination of letters, or a letter followed by a number. You can even use longer words or your name, but the Commodore will only recognize the first two letters of each. It would think BILL and BITSY were the same variable. Watch out.

There are several letter combinations and words that cannot be used for naming variables, however. Called "reserved" words, they are the names and abbreviations for commands and functions in the BASIC language. If you use a BASIC command (or just the first two letters of a few of them, like GO), you will get an error message instead of a working program. Although Commodore has minimized the problem, conflicts can pop up and bring programs crashing down. By the time you finish this chapter and the next one, you'll be familiar with the reserved words. But be careful.

Try typing another fill-the-screen command, but use MOM, instead of J and a different character instead of the diamond. For instance, try:

```
FOR MOM = 1024 TO 2023: POKE MOM,83: NEXT
```

You can even use the FOR/NEXT loop to clear all the characters off the screen. Just type in:

```
FOR B = 1024 TO 2023: POKE B,32: NEXT
```

The screen blanks out (at least until the Dreaded Ready pops back into view) because 32 is the screen code for a blank space. But there are faster ways of drawing a blank. You can also clear the screen by holding down SHIFT and pressing CLR/HOME at the same time. Try it, and you'll finally be rid of the Dreaded Ready!

Actually, the FOR/NEXT loop you used was a special case of that command, because it counted or stepped up by one each time the program looped through it. By adding another word to the FOR command, you can make the program step up (or down if you use negative number steps) by any amount or in any increment that you wish. For instance, to print something in every other character position, you would make the loop step by 2.

The screen code for an asterisk is 42. Try this:

```
FOR Q = 1024 TO 2023 STEP 2: POKE Q,42: NEXT
```

By changing the STEP again, you can make two vertical columns of asterisks. Just to be different, you can shift the starting address five positions so that the first column of asterisks is five positions from the left. First clear the screen (by pressing SHIFT and CLR/HOME) to get rid of the old asterisks. Then type:

```
FOR Q = 1029 TO 2024 STEP 20: POKE Q,42:
  NEXT
```

Because there are 40 character positions in a row, by putting an asterisk every 20 positions, you get two per row.

By using different numbers for the addresses, steps, and screen codes, you can waste half an evening putting things all over your monitor screen—and you haven't even started programming yet!

Remember, if you have a newer Commodore 64 and your characters remain invisible, POKE a color into each color address (55296 to 56295) using the first FOR/NEXT loop you typed.

You may notice that when the Dreaded Ready pops back on the screen, it's in a color different from the one you POKEd in. Furthermore, characters that later appear at the location of that message (or anything else you've typed in since POKEing in individual character colors) come up not the

color that you POKEd but the character color the Commodore used *before* you did your POKEing. That's because the Commodore 64 uses the same memory addresses that you do for storing the color of characters. Every time the computer puts a character on the screen, it POKEs a color into its color memory. When you POKE colors into individual addresses, you don't alter the Commodore's idea of what color it should POKE into them when it controls the typing. The Commodore will POKE in either the color you previously chose using the CTRL or COMMODORE keys or the color that's automatically set when the computer is turned on, light blue.

Note, too, that what you POKE into memory is not anchored to the screen position you POKE it to. By using the CRSR [↓] key repeatedly, you can *scroll* the text—move it up the screen as if you were reading from a scroll of parchment. (Using the CRSR ↑ key won't bring it back once it disappears, though.) The color of the characters doesn't change, because scrolling just makes the computer take what's in each screen memory location and pop it into a location 40 addresses away. The computer accepts what you've POKEd into its screen memory and moves it around as if it had put the data there itself.

Computing with Your Computer

One of the reasons you bought a computer was probably to do some computing. It's about time that your Commodore 64 shows off some of its mathematical prowess and shows you how easy it is to get an answer to even the most difficult problem. All you have to do is type in the problem you want an answer to and the Commodore 64 will figure it out. The only hard part is that all arithmetic is done horizontally, from right to left, using + for addition, – for subtraction, * for multiplication, / for division, and " ↑ " to indicate that the number following it is an exponent. Each one of these *operators* has its own separate key to make things easier for you.

You don't need an equals sign, because when you type the problem and finish with a RETURN, the Commodore knows you want the answer. Now go ahead. Type "2+2," and hit RETURN. In an instant, the Commodore has calculated the answer and is proud to announce it's ready to figure out another problem.

Something wrong? You don't see the answer? That's because computers are very lazy. They won't do anything they're

not told to do. You asked the machine to figure out a problem, and it did. But you forgot to command the machine to *tell you the answer!* The computer knows the answer, but it won't tell you what it is unless you ask.

The command to get the computer to communicate with you through the monitor is PRINT. If you command your Commodore to PRINT 2+2 and tap RETURN, you'll get your answer. Or you can be more complex, using parentheses for calculations like PRINT ((2345/12)*(53−98))/2.

The computer reads the parentheses and works first on the part of the problem enclosed in the most parentheses. Without the guidance of parentheses, the Commodore doesn't simply work a problem from left to right but follows "rules of precedence." Certain operations are done before others, in the order shown in Table 2.

TABLE 2: COMMODORE 64 RULES OF PRECEDENCE

1. The Commodore affixes values to NEGATIVE NUMBERS.
2. The Commodore multiplies out all EXPONENTS.
3. The Commodore MULTIPLIES and DIVIDES.
4. The Commodore ADDS and SUBTRACTS.

The Commodore 64 performs all operations in each of the four above classes from left to right before going on to the next class and works outward from the innermost pair of parentheses (when you use them).

The Commodore is particularly adept at calculating and doesn't mind doing it over and over again. Try this:

```
FOR I = 1 TO 100: PRINT 1+X: NEXT
```

You can substitute a formula for a number in a command and the Commodore 64 will calculate the number every time it needs it. For instance, you can POKE a screen filled with both colors and shapes by typing in this command:

```
FOR K = 1 TO 1000: POKE(K+1024),87:
  POKE(K+55296),7: NEXT
```

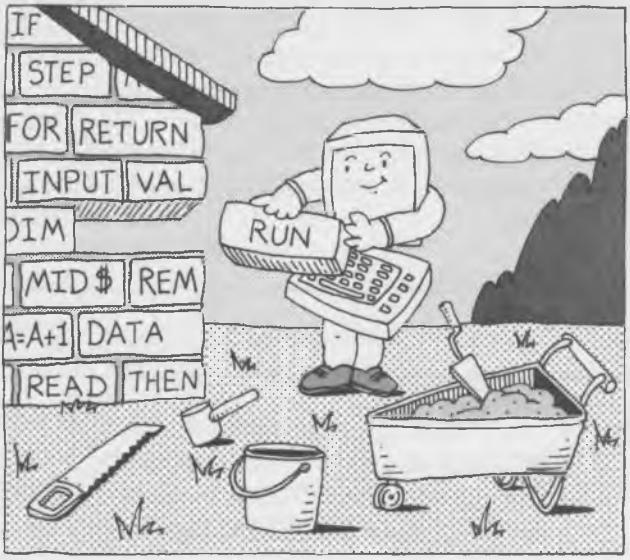
Each time the FOR/NEXT loop steps, it POKEs both a charac-

ter (87, a circle) and a color (7, yellow) into a screen memory address. As the loop counts upward, it adds one to the last address and, in so doing, fills each screen memory address. Using exactly the same command, you can fill the screen with any symbol of any color. Try some combinations.

Once you feel secure with giving your Commodore one-line commands, you're ready for your next step toward mastering the machine—programming.

5

STEP BY STEP: PROGRAMMING THE COMMODORE



To the uninitiated, computer programming is a lot like voodoo or black magic, full of strange rituals and unintelligible incantations. The strange formulas computer programmers type on their computer screens have a mystical element, a mixture of English (well, they look like words), numbers, and symbols thrown together like yesterday's beef stew.

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

You can actually make use of BASIC programs with abso-

lutely no knowledge of BASIC at all. After you search through books and magazines full of program listings and find a program that you think looks like fun, all you need to do is to type it into your Commodore, character by character, exactly as it is printed on paper. Simple. Just remember that there are subtle differences between different "dialects" of the BASIC language. Your Commodore 64 will only run programs that have been written specifically for it.

Nevertheless, an understanding of the underlying structure and workings of BASIC will unlock the mystery of such prefabricated programs and help you customize them for your own purposes. You'll have to know some of the rudiments of BASIC just to make canned programs work with your 64. Using the disk drive or Datassette requires you to type in simple BASIC instructions.

Fortunately, BASIC is designed for beginners. That's what its name stands for: Beginners' All-purpose Symbolic Instruction Code. In fact, even if you're a complete computer novice, all you need is a little experience with BASIC to begin to write your own programs.

This chapter will introduce you to some elementary concepts of BASIC programming. If you try the examples on your Commodore 64 as you read through the book, you'll become familiar with the basic BASIC commands and learn the rudiments of organizing and putting together a program. But this one chapter won't teach you all there is to know about BASIC. BASIC is so versatile that covering all of its intricacies takes whole books and more. But this chapter will show you that BASIC programming is nothing to fear. At its worst, it's relatively easy. It can actually be fun.

The rules for programming the Commodore 64 in BASIC are simple. For now, assume that each command step must be written on a separate line. Later, as you become a more proficient programmer, you can put multiple commands on a single line by separating them with colons. But when you're beginning, it's a lot easier to follow program logic if you put each command on its own line.

Each command line must begin with a *line number* that tells the computer the order in which to work the problems. Unless you tell it otherwise, the Commodore will execute the command on line 1 before going on to line 2. You indicate the end of each line of the program by hitting the RETURN key.

The BASIC language used by your Commodore 64 has two modes. The mode you've been using up till now is the *Immediate* or *Direct* Mode, which reacts to your commands as

soon as you press RETURN at the end of a line. The *Program* Mode is far more powerful. It allows you to type almost any number of program lines (until you run out of those BYTES FREE your Commodore mentions to you when you turn it on) and store them in memory to be executed one after another when you command the computer to RUN the program. The only outward difference between the commands in the two modes is that Program Mode commands *must* be preceded by a line number and Immediate Mode commands must *not* be preceded by a number.

Creating and Destroying Your First Program

Taking the first step from Direct Mode to Program Mode isn't hard. Just type the following lines (and don't forget to press RETURN after each one):

```
1 Y = 2 * 3
2 PRINT Y
```

Once you've typed in the program, tell your computer to RUN it by typing "RUN" (and pressing RETURN). Instantly, you should have your answer.

Big deal. You could have typed "PRINT 2 * 3" and gotten the same results. Yes, but there is a *big* difference. The computer has memorized what Y stands for, and it remembers that value until the next line. In fact, it will remember the value you assigned to that Y until you decide on a new and different value for it! Whenever you call for Y later in your program, it will use the value you've assigned. Try this:

```
3 PRINT Y
```

Now if you RUN your program, you'll get your answer twice. You didn't have to type lines 1 and 2 again. Your Commodore will remember them (or any program lines) until either you type new lines with those numbers or until you tell it to erase its memory.

In mathematics (and computer programming), a letter that can be assigned any numerical value is called a *variable* because its value can vary. Although Y is equal to 6 in your program, you can change its value by changing the numbers in the first line of the program. The opposite of a variable is a

constant, a number the value of which does not change—which just means a number like 1, 15, or 55000000.

Variables and how they work are extremely important to computer programming. An example will show you part of the reason why.

Try typing this:

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

Be sure you have checked that all your parentheses are all properly "paired." Then RUN the program (type RUN and press the RETURN key). If you work the problem longhand, you should come up with the same answer as the computer. If your answers don't match, you'd be smarter to check your work than to take your Commodore 64 in for repairs!

You've only had to type the values X and Y once, but the computer has used each number four times in the formula. When both X and Y are only one digit long, that doesn't make much difference. But to see how assigning a value to a variable can save you some time, try typing this:

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

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

As great as saving keystrokes may be for preventing blisters on your fingertips, it's not the only or the best use for variables. Changing the values of variables automatically and under program control is one of the most powerful abilities of your Commodore 64 and its BASIC language.

You should have noticed by now that whenever you retype a program line, using the same line number as an old one, the Commodore 64 automatically replaces the entire old line with the new one, making changes in programs easy. It does not matter when or where you type replacement program lines; the latest version of each number is the one the computer remembers. The order in which you type the original lines doesn't matter, either. Before the Commodore 64 runs a

program, it sorts through all the lines and arranges them in proper numerical order, starting with the lowest one.

If you think about it, you can see a potential problem. Old program lines can get accidentally mixed with new program lines. If there were no way of weeding out unwanted old lines, pretty soon there'd be a goulash of commands not even the computer could sort out. If the havoc went on too long, the entire memory of the computer could be swamped with old programs. One way to erase an unneeded program line is just to enter the number of the unwanted line and press RETURN. The blank line replaces the old one. The Commodore 64, not wanting to be bothered by nonentities, just lets line numbers followed by big blanks slip its mind.

But deleting a hundred-line program a line at a time would be more than a bit tedious. Obviously, there should be some way to tell the computer that you're done with the whole of an old program and that it should be entirely thrown out to make room for a new program. That command is NEW. Type in:

```
NEW
```

and press the RETURN button. Nothing happened? It only looks that way. Type RUN (and, obviously, hit RETURN) again. You're back to the multiplying Dreaded Readies. Your program is gone, but it didn't die in vain. You've learned that you should always type NEW before you start entering any program.

Learning Loops

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

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

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

When you RUN the program, you should get a column of numbers, 1–20. You've just taken our old familiar one-line FOR/NEXT command and written it on three lines, using separate lines instead of colons (:) to separate the parts of the program. By doing so, you've gained some versatility.

After you RUN the program the way it's typed above, *READY* will pop back into view. Retype line 20 as shown in the line below, making certain all of your punctuation is as shown:

```
20 PRINT A;
```

Ending the line with a semicolon told the computer not to advance a line or skip a space before PRINTing the next A. You can also end lines with a comma. Try:

```
20 PRINT A,
```

The comma makes the computer *TAB* between characters, much like a typewriter's TAB key. It puts the next character at specific screen positions: the first, tenth, twentieth, and thirtieth columns.

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

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

Note that the semicolon must be outside the last quote mark. Now RUN the program.

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

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

The quotation marks tell your Commodore 64 that the material typed between them is meant to stay together as a string, and that the individual numbers, letters, and symbols in the string should not be treated as a lot of different individual variables. A PRINT command followed by characters in quotes causes your computer to type out everything between

the quotes. In fact, the Commodore 64 will type *anything* that you have entered between quote marks, even simple commands for cursor movement or changing character color. The two exceptions are DELETE and RETURN. The RETURN still makes the computer think you're finished with typing the program line.

Try adding these new lines to your program. Don't type all the letters between the angle brackets (<>); just press the combination of keys listed there.

```
17 PRINT "<CTRL and 0 pressed
         simultaneously>"
22 PRINT "<CTRL and 8 pressed
         simultaneously>"
```

You may be surprised to discover that the computer displays graphic symbols for the commands you've typed. But when you RUN the program, you'll see that it changes the character colors between the time it PRINTs the words and the numbers just as if you had typed in the commands while the program was running. When the Commodore prints the string, it duplicates the keystrokes you made in entering the program, even to the extent of issuing commands to itself automatically. The Commodore 64 actually enters a new mode of operation, called *quote mode,* between the quotation marks. The quote mode abilities of your Commodore allow you to format screens for your programs. That means you can make your monitor screen look any way that you want. See Box 6 for another nifty trick.

Another of the powerful features of the Commodore 64's BASIC language is its ability to change automatically the

BOX 6: TIRED OF TYPING?

As you do more programming, you'll quickly tire of typing out the same words over and over again. The engineers at Commodore figured that PRINT would probably be one of those words, so they've provided an abbreviation that works exactly like typing out the whole word. It's simply a question mark (?). Not only will the computer understand what you mean by "?," but whenever you LIST the program, it will spell out the whole word PRINT for you. Try it. Type in "? 2+2".

values of variables within a program as it is running. Type NEW, then enter this program:

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

Line 40 might not make sense to you, but if you RUN the program, the results won't seem unusual. If you wrote an equation like this on a math examination, you might earn a place in a remedial arithmetic class. In BASIC, however, it's perfectly acceptable. What it means is that the value of J should be changed to one greater than its old value.

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

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

Now the computer prints pairs of numbers. When one of the pair increases by 1, the other decreases by 2.

The Amazing Jumping Program

Now that you have a good grasp of FOR/NEXT loops, you should try another way of looping around in BASIC. The Commodore 64 can make loops with another command, GOTO. GOTO is sometimes called a *jump,* because instead of letting the program progress step-by-step, it makes the program jump ahead or back to a specific place. In effect, a GOTO tells the program to "go to" a given line number without any question. Therefore, a line number must always follow the word GOTO.

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

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

Your computer will continuously PRINT the string in line 10 until you press the RUN/STOP key. Line 20 makes the computer go back and execute line 10 again, after which it

progresses to line 20, which makes it execute line 10 again, after which it progresses to line 20, and—you get the idea.

After a program has been STOPped by pressing the RUN/STOP key, you have your choice of two ways of restarting it again. Entering RUN (typing it in and pressing RETURN; pressing the RUN key does absolutely nothing) starts the program all over again from the beginning. Entering CONT (for CONTinue) causes the program to continue RUNning where it left off. The difference between the commands becomes very important in long programs. With CONT you don't have to wade through all the preliminary steps of a program over and over again.

To see what's really happening in an endless loop, type NEW and try this simple program:

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

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

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

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

Closely related to the GOTO jump is the GOSUB command, which lets you make a two-way jump. Like GOTO, GOSUB is always followed by a line number and sends the program looking for a new line number out of the step-by-step sequence. Then the program jumps to the new line and starts executing a subsidiary part of the main program called a *subprogram* or *subroutine* there.

What makes GOSUB different is that the subroutine it "calls" must end with a RETURN command—the word "RETURN" on a program line, not to be confused with the RETURN key on the keyboard. The RETURN command instantly bounces the program back to *the line after* the GOSUB command. If it went back to the GOSUB command, the program would be caught in an endless loop.

Try this:

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

Line 40 never does print because it's outside the GOTO loop and the program never gets that far. But the GOSUB command forces the program down to line 50, and the RETURN command in line 60 sends the program back to line 30.

Subroutines are programs that appear within larger programs. They are extremely handy because you can reuse them without having to retype the lines. The program in Box 8 uses several subroutines. They make the program more compact and easy to understand.

The Color Checker

You're now ready for your first full-length program that does something almost useful.

You know how to change the background and border colors, but just POKEing around to find the right combination is time-consuming and tedious. Things that are time-consuming and tedious are the easiest to computerize. Here is a simple program that uses nothing but familiar old POKEs, PRINTs, and FOR/NEXT loops.

One part of the program you haven't seen before is the memory location 646 that is POKEd into. Address 646 holds the color code for the colors being typed on the screen. When the Commodore displays a character, it puts the value from 646 into the appropriate address in the screen color memory map.

Also note the strange indication between angle brackets (<>) in the first PRINT command. All it means is that you should hold down the SHIFT key and press CLR/HOME. What you'll see for your efforts is a heart. When your computer PRINTs this line, it automatically clears the screen so the program can work with a big blank screen. You might want to include this command at the beginning of every program you write.

The <5 CRSR-DWN> commands simply indicate that

you should press the CRSR-DOWN ARROW key the number of times noted right after the first angle bracket. A reversed Q should appear on your screen each time you do, since the computer will be in quote mode following the quote marks. The <SHIFT CRSR-DWN> commands merely mean you should hold down SHIFT and press the CRSR-DOWN ARROW key. That will get you a reversed dot on the screen. Although the up and down CuRSoR commands may seem to cancel one another out, the combination actually produces a nice on-screen arrangement.

This program will PRINT one line in each of the sixteen available Commodore colors, then cycle through every available background and border combination. If you see one you like, just press the RUN/STOP button. Not only will your favorite colors stay on the screen and in memory for future use, but the proper numerical values for future POKEing will also be displayed!

```
10 PRINT "<SHIFT CLR/HOME>"
20 PRINT " *** THE COMMODORE COLOR CHECKER
   *** <CRSR DWN>"
30 FOR I = 0 TO 15
40 POKE 646,I
50 PRINT "THIS IS COLOR NUMBER "; I
60 NEXT I
70 FOR J = 0 TO 15
80 POKE 53281,J
90 POKE 646,J+1
100 PRINT "<2 CRSR-DWNS> THIS IS BACKGROUND
    COLOR NUMBER ";
110 PRINT J; "<3 SHIFT CRSR-DWNS>"
120 FOR L = 0 TO 15
130 POKE 53280, L
140 PRINT "<5 CRSR-DWNS> THE BORDER IS COLOR
    NUMBER ";
150 PRINT L; "<6 SHIFT CRSR-DWNS>"
160 FOR M = 1 TO 500
170 NEXT M
180 NEXT L
190 NEXT J
200 END
```

Making a LIST

After a while, you can pile up an awful lot of commands in an order that's very hard to follow. It may seem difficult to keep track of them all. But fortunately you can look at the pro-

gram you've entered. All it takes is one command: LIST. This command lets you peer inside the computer's memory and see how it has rearranged the program. Type LIST and press RETURN. The whole program will roll forth from the computer's memory, with all of the program lines arranged in order.

If the program listing is longer than the screen, lines will scroll off the top of the screen as new ones are added at the bottom. If this scrolling goes too rapidly for your eyes to follow, you can slow it by holding down the CTRL key as the program is LISTed. You can stop the LIST at any time by pressing the RUN/STOP key.

To look at a single program line, simply type LIST followed by the line number. LIST 40 would show you line 40 in all its glory. To see a group of lines, type LIST followed by the starting number, a hyphen, and the ending number. For instance, LIST 50-100 would list all the lines from 50 to 100, inclusive.

To LIST a given line number and all the lines thereafter, type LIST and the line number followed by a hyphen, like LIST 50-. To see a program line and all the lines that come before it, type LIST, a hyphen, and then the line number, like LIST -50.

Saving Your Fingertips

There's another important way to use the LIST command. Commodore's version of BASIC has a powerful feature called a "screen editor" that often costs extra or simply isn't available even on more expensive computers.

Commodore's screen editor allows you to change lines of BASIC programs without having to retype each whole line. Any line of BASIC can be changed on the screen using the CRSR and INST/DEL keys. When you press the RETURN key with the cursor located anywhere in a program line, that line is added to the Commodore's memory. To edit a program, just LIST it, move the cursor around and change the lines as necessary on the screen, and press RETURN after each change.

One warning: if you change a line number using your Commodore's screen editor, remember that the old line with the original number will remain in memory until you erase it or replace it with a new one! Type NEW and then type in:

```
100 PRINT "THIS IS THE OLD LINE"
```

Now use the screen editor to change it to:

```
110 PRINT "THIS IS THE NEW LINE"
```

LIST the program. You should discover *both* lines on your screen.

Getting Data In

After you've used the color checker, you'll have found your favorite color combination. But running through every choice every time you turn your computer on wastes exactly the time you thought your computer could save you. What you need is an alternative that you can use simply and easily without resorting to a series of POKEs and mysterious, meaningless numbers. If you know the trick, you can make your computer ask what colors you want to put on the screen.

The key to getting data into your computer while a program is running is the INPUT statement. When a BASIC program encounters one, it stops and waits for something to be typed into the keyboard and the RETURN key to be struck. An INPUT statement can even ask the operator a question or suggest the sort of information that the program is looking for.

First let's try a program that changes the screen colors. Type this into your Commodore 64:

```
10 INPUT "COLOR NUMBER FOR CHARACTERS"; A
20 INPUT "COLOR NUMBER FOR BACKGROUND"; B
30 INPUT "COLOR NUMBER FOR BORDER"; C
40 POKE 646,A
50 POKE 53281,B
60 POKE 53280,C
```

RUN the program. It will ask for the first color number. You type in your answer and a RETURN, and the program will progress to the next question. When you're done, it will carry out your command.

You can type in any number you want in response to the question, and the computer will try to comply with your request. But often a number that's "out of bounds" can make a program die. If you want to be obstinate about it and have the computer only accept numbers less than 15 to comply

with the Commodore scheme for naming colors, you can add a "conditional test" to each reply. It's called an *IF . . . THEN statement.*

Conditioning Your Computer

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

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

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

The > sign means "is greater than" to BASIC, so it tests the variables A, B, and C as you enter them in response to the INPUT statements, checking each one to see whether it's larger or smaller than 15. RUN the program. Try to specify a color number 16 or higher. Just try! The Commodore will obstinately ask the question again until you come up with a reasonable value. See Box 7 for other comparison symbols used in Commodore BASIC.

Closely related to IF . . . THEN statements are ON . . . GOTO and ON . . . GOSUB. Rather than including a single

BOX 7: SYMBOLS USED IN BASIC TO COMPARE QUANTITIES AND STRINGS

=	equals
<	is less than
<=	is less than or equal to
>	is greater than
=>	is greater than or equal to
<>	does not equal

line number to GOTO or GOSUB, ON . . . GOTO (and GOSUB) statements include several line numbers. The formats are as follows:

```
2000 ON N GOTO 128,700,950
2500 ON J GOSUB 1000,2000,3000
```

If the value of the ON part of the statement equals 1, the program jumps to the first line number in the list. If the value equals 2, the program jumps to the second line number in the list. And so on.

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

What makes the program finicky is the variable that follows the INPUT statement. If the variable is merely a letter or character combination (like A, XY, or BERTHA), the program will expect—and demand—a number. But when the variable is followed by a dollar sign (like A$, XY$ or DOLLAR$), the computer treats the INPUT as a string of letters or other characters without any numerical value at all. The strings you type in response to INPUT commands should not be enclosed inside quotation marks. The Commodore 64 knows that your response will be a string because of the "$" after the variable name, so it doesn't require quotes.

A variable with a trailing dollar sign is called a *string variable,* because the computer remembers its value as a string of characters. If you try to make a string equal to a numerical variable or something not within quote marks equal to a string variable, BASIC balks and sends you a TYPE MISMATCH error. To rouse that error message, try these immediate commands:

```
A$ = 56345
B$ = BATTLESHIP
C = "HAPPY TOADSTOOLS"
```

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

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

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

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

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

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

Try it. Note that if you type the whole word YES in reply to the computer's query, it will think you said NO because Y is not exactly the same string as YES.

There are two ways around the problem. Line 80 could be rewritten to let the operator know what kind of answer the program expects:

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

Better still, you can make the last IF . . . THEN line only look at the leftmost character in the response string. Replace line 90 by typing in the following:

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

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

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

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

```
10 A$ = "HI THERE, EVERYONE"
20 PRINT LEFT$(A$,2)
30 B$ = "EARTHLANDER"
40 PRINT MID$(B$,7,3)
50 C$ = "GOOD-BYE"
60 PRINT RIGHT$(C$,3)
```

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

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

Actually, the Commodore 64 stores and manipulates all characters and symbols as numbers in its memory. It assigns each character a numerical code known as ASCII. The ASC command lets you see the code for any character, though if you try to get more than one character at a time, you'll get only the value of the first character. Try these immediate commands:

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

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

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

The program in Box 8 will give you practice with strings and subroutines—along with the time of day.

Screen Formatting—Making a Menu

The simple program we put into your computer's memory to let you change the screen colors has one minor problem. Someone who has never seen the color codes for the Commodore 64 will have no idea what color equals what

BOX 8: TIME FOR PRACTICE STRINGING

Here's a program that will help you practice manipulating the characters in strings and introduce you to the "real-time" clock built into your Commodore 64. When you RUN this program, it will POKE a digital clock in the bottom left-hand corner of your monitor screen.

The Commodore 64's real-time clock is displayed as a string. You can see it by typing PRINT TI$. To set the clock, you type in TI$ = "HHMMSS". HH stands for the hours expressed in 24-hour military clock fashion, MM stands for the number of minutes, and SS stands for the number of seconds. Because the clock is stored as a string, you must enter its value in quotation marks to avoid a TYPE MISMATCH error.

The line numbers of this program start high so that you can add it to the end of another program. When the first program is done, your computer will turn into a clock until you find something else to do.

```
5000 FOR I=56285 to 56295: POKE I,1:
     NEXT I
5010 POKE 2023,13
5020 A$=LEFT$(TI$,2):A=VAL(A$)
5030 IF A=> 12 THEN POKE 2022,16
5040 IF A<12 THEN POKE 2022,1
5050 IF A>12 THEN A=A-12
5060 E$=MID$(TI$,6,1): E=VAL(E$):
     POKE 2020,E+48
5070 F$=MID$(TI$,5,1): F=VAL(F$):
     POKE 2019,F+48
5080 POKE 2018,58
5090 G$=MID$(TI$,4,1): G=VAL(G$):
     POKE 2017,G+48
5100 H$=MID$(TI$,3,1): H=VAL(H$):
     POKE 2016,H+48
5110 POKE 2015,58
5120 IF A=0 THEN GOSUB 6000
5130 IF A>0 AND A<10 THEN GOSUB 7000
5140 IF A>9 THEN GOSUB 8000
5150 GOTO 5000
6000 REM 1200 TO 100 SUBROUTINE
6010 POKE 2014,50
6020 POKE 2013,49
6030 RETURN
7000 REM 100 TO 1000 SUBROUTINE
7010 POKE 2014,A+48
```

(*Continued*)

```
7020 POKE 2013,32
7030 RETURN
8000 REM 1000 TO 2000 SUBROUTINE
8010 POKE 2014,(A-10)+48
8020 POKE 2013,49
8030 RETURN
```

number. What we need is a *menu* telling what selections are available.

Using PRINT statements, we can paint a menu across the top of the screen. We can combine some of the symbols on the keyboard to make simple graphics. The listing below is a fine-tuned version of the color selector program with a beginning menu drawn from simple keyboard graphics. It sets the border, background, and character colors to take advantage of some of the Commodore 64's capabilities, but by now you should be able to change the colors to suit yourself.

Once again, the information between angle brackets tells you what key combinations to press and how many of them. Don't type the brackets or what's inside them literally. COM, as you may have guessed, refers to the Commodore key. Remember to type NEW before you start entering the program.

```
10 PRINT "<SHIFT CLR/HOME>":POKE 646,6:POKE
   53280,2: POKE 53281,12
20 PRINT " *** COMMODORE 64 COLOR SELECTOR
   *** ": POKE 646,1
30 PRINT "<SHIFT O> <38 COM Y> <SHIFT P>";
40 PRINT "<COM H>   0=BLACK       1=WHITE
   2=RED      <COM N>";
50 PRINT "<COM H>   3=CYAN        4=PURPLE
   5=GREEN    <COM N>";
60 PRINT "<COM H>   6=BLUE        7=YELLOW
   8=ORANGE   <COM N>";
70 PRINT "<COM H>   9=BROWN      10=LT RED
   11=DK GRAY <COM N>";
80 PRINT "<COM H>  12=MED GRAY   13=LT GREEN
   14=LT BLUE <COM N>";
90 PRINT "<COM H>                15=PALE GRAY
   <COM N>";
100 PRINT "<SHIFT L> <38 COM P> <SHIFT @>";
110 POKE 646,13: INPUT "COLOR NUMBER FOR
    CHARACTERS"; A
120 IF A>15 THEN GOTO 110
130 INPUT "COLOR NUMBER FOR BACKGROUND"; B
140 IF B>15 THEN GOTO 130
150 INPUT "COLOR NUMBER FOR BORDER    "; C
160 IF C>15 THEN GOTO 150
170 POKE 646,A
180 POKE 53281,B
```

```
190 POKE 53280,C
200 PRINT "<CRSR DWN>DO YOU WANT TO TRY
    ANOTHER COMBINATION?"
210 PRINT "<2 CRSR DWN>                ('Y' OR
    'N') <4 CRSR DWN>"
220 INPUT D$
230 IF LEFT$(D$,1)="Y" THEN GOTO 10
240 PRINT "<SHIFT CLR/HOME>"
```

This program demonstrates another important feature of BASIC: that there are always multiple ways of achieving the same result. Usually, there is no best way. Rather, each different path has its advantages and disadvantages. For instance, instead of typing all the necessary individual blank spaces to separate the selections on the menu, you could use the Commodore 64's time-savers: SPC and TAB.

SPC(n) tells the Commodore to insert n spaces before PRINTing the next character. TAB(n) tells the Commodore 64 to PRINT the next character in column n. Try these samples to see the difference:

```
PRINT "BANG!"; SPC(20) "OUCH!"
PRINT "BANG!"; TAB(20) "OUCH!"
```

Instead of including the prompt string as part of the INPUT statement, you can use one or more PRINT statements to put words on the screen before the INPUT is asked for. For instance, try this:

```
10 PRINT "TYPE IN YOUR NAME"
20 PRINT "LAST NAME FIRST, FOLLOWED BY YOUR
   FIRST NAME"
30 PRINT "DON'T FORGET THAT NEATNESS COUNTS"
40 INPUT A$
50 PRINT "OH, SO YOUR NAME IS"; A$
```

A command called GET can perform much of the same function as INPUT but in a slightly different way. GET merely matches what is typed on the keyboard with a variable. For instance, the command GET A$ will assign the string variable A$ the next character typed on the keyboard. GET X$, Y$, Z$ will read three keys from the keyboard.

One clever thing GET can do is make a program pause until you press a key on the keyboard. Try this:

```
10 PRINT "STRIKE ANY KEY WHEN READY"
20 GET A$
30 IF A$ = "" THEN 10
40 PRINT "THE LOOP IS \BROKEN"
```

The " " in line 30 is what's known as a *null string*—a string with nothing in it (not even blanks, which are *something*). A$ is null until a key is pressed, so the program loops until a key is struck.

READing DATA

So far we've assigned values to variables by using simple equations like A = 256. That can become cumbersome when a program must use lengthy lists of numbers.

Long lists of data can be put into a program with the help of a DATA statement. It takes the form of the DATA command itself, followed by a list of the data in which each element is separated by a comma. A typical DATA statement looks like this:

```
DATA 1,1,2,3,5,8,13,21,34,55,89,134,223
```

The DATA statement does not put the data into the program. Instead, a command called READ sends BASIC looking for DATA statements (which can appear either after or *before* the READ). The READ command is followed by variable names that BASIC matches one-for-one and in the same order with the data listed in the DATA statements in the program.

The following READ and DATA statements assign the value of 6 to A, 7 to B, and 16 to C.

```
10 READ A,C,B
20 DATA 6,16,7
```

If you try to read more variables than there are elements in DATA statements, you get the message "OUT OF DATA ERROR." The information in DATA statements can be used over again. The command RESTORE sends subsequent READs back to the beginning of the first DATA statement to gather up values.

All the Stuff That's Too Complicated for One Meager Chapter on BASIC Programming, So It Will Just Be Mentioned

Files

A file is the Commodore 64's way of bundling data. Information must be tucked into a file to be sent to a peripheral or stored on tape or disk.

The easiest way to understand how files work is to think of them as file folders that can be stuffed into mailing envelopes. Before anything can be put into a file folder, the folder must be opened. The BASIC command that prepares a file for data entry is therefore logically called OPEN. Every file folder must have a name to identify it. The Commodore 64 uses a number between 1 and 255 to name files, but for reasons we'll explain in a later chapter, it's best to stick with the numbers below 128.

Next comes the address. Like the address on the envelope of your file folder, the computer file address tells your Commodore where the data in the file will go or come from. This address is expressed as a "device number": usually 0 for the monitor screen, 1 for the cassette, 4 for the printer, and 8 for the disk drive.

Next comes a secondary address. Actually more of a command, the secondary address tells your Commodore 64 specific details about the file. For instance, the secondary address for a disk file is called the "channel number," which specifies a pathway between computer and disk drives.

After the secondary address is more information in the nature of a command and identification, and some of it is redundant. You'll find the details in this book's chapters about the cassette, disk drive, and printer.

Another group of commands that send and retrieve information from files are GET#, INPUT#, and PRINT#. Although the Commodore normally ignores extra spaces in command lines, you *cannot* add a space between the command name and the # sign. These commands work like their namesakes (without the terminal #) except they send data to the specified file instead of the monitor screen. Each command must be immediately followed by the number that identifies the file and a comma.

The Commodore 64 only allows ten files to be OPEN at one time. To keep things under control, another command CLOSEs files that are OPEN. A CLOSE command needs to be followed by the number of the file to be CLOSEd.

If you find it confusing, you're not alone. Using data files properly is one of the highest arts of BASIC programming. Yet mastery of file functions is necessary only when your programs must manipulate massive blocks of data that cannot all be kept in your Commodore 64's RAM at one time. If you're not planning on dedicating the next few months of your life to becoming a software hacker, probably the most important use you'll have for file commands will be sending text to the printer. You'll find more about this in the chapter about the printers.

Sprite Graphics and Sound Generation

Probably two of the major reasons you bought a Commodore 64 are its sound and graphics capabilities. Commodore sprite graphics and the Commodore SID sound chip make both functions easy and frustrating. Easy compared to other computers. Frustrating when you want to create more than primitive sounds and graphics with the Commodore. Getting all the details right takes a lot of patience.

Both sound and graphic functions can be created by using only the handful of BASIC commands that you have already learned. The key to handling them properly is knowing how to POKE the proper values into the proper memory locations.

Sprite graphics are fancy indeed. With the simple graphics that you've already learned, you know that you can POKE any character you want into any memory location. Sprite graphics allow you to make your own larger characters called "sprites" and place them anywhere on (and sometimes off) the monitor screen. You give your sprite a code number, so that you only have to refer to the code number (as opposed to a description of the sprite) when you want to move the sprite around.

The Commodore 64 can handle up to eight sprites at one time. Defining the shape and color of sprites and moving them around requires only that you POKE the right numbers in the right places in the right order. Certainly that's an oversimplification, but it must be. Using sprite graphics is an art in itself, and covering the subject completely could easily take a whole book.

Both sprites and sound can be deviously deceptive. Both functions are easy to bring to life but extremely complex to use artfully. Just playing a few seconds of a melody can take dozens of program lines, each with its share of frustration and error as you write it. If you plan to make extensive use of either function, a *development system* is an advisable addition to your program library.

A development system takes the simple BASIC graphics and sound commands and controls them with sophisticated higher-level commands. Functions that might take several program lines and dozens of POKEs are reduced to a single command. If you're serious about sound and graphics, a development system will make your life a lot easier.

Music Demonstration Program

"Pop Goes the Weasel"

```
10 DIM NH(15),NL(15)
20 GOSUB 1000
30 POKE 54276,33:POKE 54296,10
40 POKE 54277,50:POKE 54278,125
50 FOR I = 1 TO 31: READ N,D,E
60 POKE 54276,33
70 POKE 54272,NL(N)
80 POKE 54273,NH(N)
90 IF NOT E THEN POKE 54272,32
100 FOR J = 1 TO D:NEXT J
110 NEXT I
120 POKE 54296,0
130 END
1000 FOR I = 1 TO 15:READ NH(I),NL(I):NEXT
1010 RETURN
1020 DATA 16,195,17,195,18,209,19,239,21,31,
     22,96,23,181
1030 DATA 25,30,26,156,28,49
1040 DATA 29,223,31,165,33,135,35,134,37,162
2000 DATA 6,250,0,6,125,0,8,250,0,8,125,0
2010 DATA 10,125,-1,13,125,-1,10,125,-1,6,
     250,0,1,125,0
2020 DATA 6,250,0,6,125,0,8,250,0,8,125,0
2030 DATA 10,375,0,6,125,0,1,125,0
2040 DATA 6,250,-1,6,125,-1,8,250,-1,8,125,0
2050 DATA 10,125,0,13,125,0,10,125,0,6,250,
     0,0,125,0
2060 DATA 15,125,0,0,250,0,8,250,0,13,125,0,
     10,375,0,6,500,0
```

Sprite Demonstration

This draws a highway and animates three cars—press any key to stop.

Note: lines 30, 40, 200, 230, and 270 have each been printed on two lines here because of space requirements. Be sure you enter them as one line each. That is, don't press RETURN until you've entered the whole line. In lines 30 and 40, you must enter special characters by holding down the CTRL key and pressing a number. You must also hit the space bar forty times where the program indicates <40 SPACES>.

```
10 PRINT "<SHIFT CLR/HOME>"
20 POKE 53281,5
30 FOR I = 1 TO 6:PRINT "<CTRL 9> <CTRL 7>
   <40 SPACES> <CTRL 0>";: NEXT
40 FOR I = 1 TO 3: PRINT "<CTRL 9> <CTRL 2>
   <40 SPACES> <CTRL 0>"; NEXT
50 SL=832
60 FOR I = 0 TO 26:READ T:POKE SL+I,T:NEXT
70 DATA 0,0,0,1,248,0,7,196,0
80 DATA 7,199,128,63,255,192,127,255,192
90 DATA 49,241,128,10,10,0,4,4,0
100 FOR I = 27 TO 63:POKE SL+I,0:NEXT
120 POKE 53269,0:POKE 53264,0
130 POKE 53248,0:POKE 53251,0
140 POKE 2040,13:POKE 2041,13
150 POKE 53249,100: POKE 53251,100
160 POKE 53248,0:POKE 53251,0
170 POKE 53287,7:POKE 53288,5
180 POKE 53271,1:POKE 53277,1
190 POKE 53269,3
200 FOR I = 1 TO 255:POKE 53248,I:IF I/2=
    INT(I/2) THEN POKE 53250,I/2
210 NEXT
220 POKE 53248,1:POKE 53264,1
230 FOR I = 1 TO 100:POKE 53248,I:IF I/2=
    INT(I/2) THEN POKE 53250,128+INT(I/2)
240 NEXT
250 FOR I = 100 TO 255:IF I/2=INT(I/2) THEN
    POKE 53250,128+I/2
260 NEXT
270 POKE 53248,0:POKE 53264,2:POKE 53250,
    0:POKE 53287,0:POKE 53271,0
280 FOR I = 1 TO 127:IF I/2=INT(I/2) THEN
    POKE 53250,I
290 POKE 53248,I*2
300 NEXT
310 POKE 53248,0:POKE 53264,3
320 FOR I = 1 TO 100:POKE 53248,I:NEXT
330 GET A$:IF A$<> "" THEN 400
340 GOTO 120
400 POKE 53269,0
410 END
```

Mathematical Operators

The BASIC built into the Commodore 64 has a reasonably large repertory of mathematical functions. If you plan to use your Commodore for heavy calculating, the following choice of functions is at your fingertips:

1. ABS (for ABSolute) gives the absolute value of the following number. For instance, PRINT ABS(–3) will put the number 3 on the screen.

2. ATN (for ArcTaNgent) finds the arc tangent of the number following, that is the angle (in radians) corresponding to the tangent of the number following the command. Typing PRINT ATN(30) would produce the result 1.53747533.

3. COS (for COSine) finds the cosine of the number following (where that number is an angle in radians). Typing PRINT COS(4) would produce the result –.65364362.

4. EXP calculates the value of *e* (the base of natural logarithms) raised to the power of the number following. That number cannot exceed 88.0296919 without an OVERFLOW error occurring. PRINT EXP(2) would give 7.389051; PRINT EXP (100) would result in ? OVERFLOW ERROR.

5. INT (for INTeger) returns a numerical value equal to the integer value nearest and below the number following the command. In other words, it lops off all the decimal places of positive numbers or finds the integer just less than a given negative number. PRINT INT(8.245676) would return 8; PRINT INT (5.994321) would give a result of 5.

6. LOG (for LOGarithm) gives the value of the natural logarithm of the number following (which must be a nonzero, positive number). PRINT LOG(12) would result in 2.48490665; PRINT LOG (–12) would get a "? ILLEGAL QUANTITY ERROR."

7. RND (for RaNDom) gives a "pseudo" random number between zero and one. It requires a number (or numerical expression) to follow the command to serve as the "seed" from which the random numbers are calculated. For every repetition of the RND operation, a new "random" number is calculated in sequence. The same seed number always generates the same sequence if the seed is positive. If the seed is negative, the RND function is reseeded each time it is called, essentially giving the same number. Zero causes the built-in Commodore clock function (TI) to be used as the seed, giving you a different sequence of numbers each time. Your

guess is as good as mine as to what result you'd get from typing PRINT RND(8)!

8. SGN (for SiGN) returns a value based on whether the number following is positive, negative, or zero. Positive returns a 1, zero a 0, and negative a –1.

9. SIN (for SINe) gives the sine (in radians) of the number following. For instance, PRINT SIN(9) would produce .412118485.

10. SQR (for SQuare Root) gives the square root of the number following. The Commodore 64 has problems finding the imaginary SQRs of negative numbers and responds with an ILLEGAL QUANTITY error message should you try. PRINT SQR(65536) results in 256; PRINT SQR(–16) results in a ? ILLEGAL QUANTITY ERROR.

11. TAN (for TANgent) gives the tangent of the number following in radians. For instance, PRINT TAN(10) would produce .648360828. Some values of TAN can result in a DIVISION BY ZERO error message.

BASIC is very versatile. If the above list does not include a mathematical function you need in your work, you can create one using the DEF FN statement, which lets you give any function or formula a variable name. To see how it works, try this:

```
10 DEF FN A (X) = X + 10
20 PRINT FN A (2)
30 PRINT FN A (100)
```

Logical Operators

BASIC makes decisions in IF . . . THEN and ON . . . GOTO/GOSUB statements based on values determined by a system of formal logic called Boolean algebra. There are two essential operators in BASIC's Boolean system: AND and OR. (There's also NOT; its complexities will not be explained here.)

BASIC's Boolean algebra uses the operators AND and OR to compare statements for truth or falsehood and makes decisions based on the results. For example, in the statement

```
IF A = 1 AND B = 16 THEN 50
```

the program will jump to line 50 only when *both* sides of the AND are true. In the statement

```
IF A = 1 OR B = 16 THEN 50
```

the program will jump to line 50 when *either* side of the OR is true.

A Remark to End

One BASIC command seems to do almost nothing at all. REM advises the Commodore 64 to ignore everything else that follows it on a program line.

The REM command is of immense value in documenting your software. It allows you to add comments to program lines to tell others (and yourself, after your memories of the program become hazy) why you did what you did or what a program line is supposed to do. Documenting your programs properly helps others learn from your experiences and helps you share your programming knowledge with others.

REM statements do take up valuable room in the Commodore's memory, however. When you want to make a program as small as possible, the first thing to be eliminated will most likely be the REMs.

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

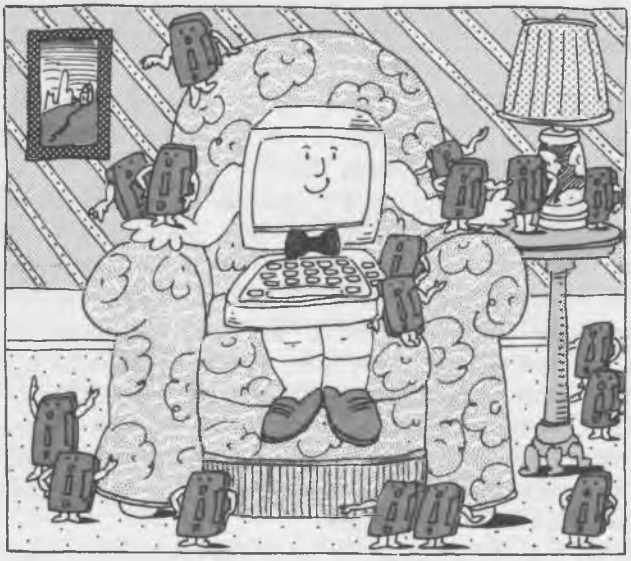
```
10 REM THIS LINE WILL NOT PRINT
20 PRINT "THIS WILL": REM "BUT THIS WON'T"
```

Sometimes you need to end a program before the last line. The command to do that is END. Try this program to see how END works:

```
10 PRINT "THIS IS THE END OF THE BASIC
   CHAPTER"
20 END
30 PRINT "THIS IS PAST THE END"
```

6

COPING WITH CASSETTES



Once you've created a program, you'll come face-to-face with a problem that has plagued mankind since the beginning of time—mortality. The life of a program is destined to be far shorter than even a human life. Programs are susceptible to the NEW command, your making (and then forgetting) subtle changes, and the whims of your computer's power switch and the power company. Flick the power off (or lose electric power), and the program evaporates forever from the volatile RAM memory of your Commodore 64. Accidentally type NEW and you have blank memory in a flash. Make a few innocent changes in a couple of program lines, and the program may stop working. If you forget which changes you made, you can be stuck for hours trying to work back to where you were before.

You can't easily keep an additional copy of a program in your Commodore 64's RAM so that you can tinker on one and still have the original copy left. Even if you did, the duplicate would be just as fragile and subject to the vagaries of the power line.

RAM is *volatile*. It forgets when the power goes, even for an instant. You can protect yourself, however, by making a

program duplicate in a *nonvolatile* storage medium. Although a printer can produce permanent hard copy, you'd still have to retype the printed listings into your Commodore's memory if you wanted to use them again. Fortunately, both cassettes and disks have nonvolatile magnetic memories that won't "forget" when the power is turned off, and both will automatically put your favorite programs back into the computer without tedious retyping. Moreover, files of the data used in programs can be stored separately on cassettes or disks using BASIC's file handling instructions.

The easiest way to save a program against almost all adversity is the cassette. Cassettes are the cheapest permanent mass storage you can get for your Commodore 64, and they're certainly not hard to use. In fact, whenever you ask to SAVE or LOAD a program, the cassette recorder is the *default* option. Give your computer no other instructions, and it automatically expects to put or find programs on cassette.

Cassettes are durable, nearly as childproof as a set of building blocks, and able to withstand all but the most severe tantrums and tempers. They're an excellent product for first-time computer users.

Cassettes are familiar and reliable, based on the same old mechanism first used for portable dictating machines two decades ago and promoted to a primary music medium ten years later. In fact, computer signals must be converted to music, or soundlike signals, to be recorded on tape. That seemingly trivial fact has great repercussions: Commodore computer signals can be recorded by nearly any cassette recorder.

The Commodore C2N Datassette unit is the easiest to install of all cassette recorders. Just plug it into the Commodore 64. One cable connects all signals to the cassette recorder and even supplies it with power. Except for its streamlined case and ominous cable, the Commodore Datassette is not much different from a normal cassette recorder. It records and plays the same signals using the same mechanism. If you wanted, you could use nearly any audio cassette recorder instead of the Datassette with your Commodore 64.

However, most machines other than those sold by Commodore will require adapters that can be as expensive as the recorders themselves. Such adapters sort out the special control signals that go to and from the cassette recorder and tell your computer what the cassette recorder is doing.

If you don't have a cassette recorder already, you might as well buy the Datassette. It will probably cost less than an equivalent cassette recorder and the necessary adapter, and you're assured it will work with your computer.

Putting a LOAD on Your Commodore's Mind

If you're capable of following simple on-screen instructions, you can probably use your Datassette without a problem. From RECORD to EJECT, all the controls on the machine itself are self-explanatory and work exactly like those on any other cassette recorder.

The only control your Commodore 64 has over the Datassette is the ability to start and stop its motor. It cannot press down on the control buttons. All it can do is ask you to do that trivial job.

Nor does the Commodore 64 know which button you press down. It can only sense when one of the controls has been pressed down to actuate the motor. The computer *assumes* that you pressed the right button (or combination of buttons), so you should always verify that you've done your job properly.

Getting a program from a cassette into your computer is called *loading* it. To LOAD a program from a tape, type LOAD and hit the RETURN key *or* press SHIFT and the RUN/STOP key simultaneously. Immediately, the message PRESS PLAY ON TAPE will pop up on the monitor screen. Just push down the Datassette's PLAY button, and the LOADing process will begin.

The monitor screen will then blank out. More accurately, the whole screen will change to whatever *border* color was in effect immediately before you gave the LOAD instruction. The blank-out is more practical than aesthetic: LOADing a program uses some of the same memory locations used in displaying information on your monitor screen. To prevent confusion, the screen is temporarily switched off during LOADing.

In a few seconds the monitor screen will revert to text and announce that it has found a program by showing the name of the program if it has a name. If you do nothing more, the Commodore 64 will wait about 30 seconds and actually LOAD the program, again blanking out the screen. But you don't have to wait 30 seconds if you don't want to wait. Pressing

the COMMODORE key, the space bar, the CTRL key, or the "back arrow" key will cause the program to LOAD immediately. If you change your mind, pressing RUN/STOP will abort cassette LOADing. Pressing other keys will have no effect. Be patient. LOADing from cassette can take several minutes.

You can also specify the name of a program you want to LOAD simply by putting the program name inside quotation marks or after a single quote mark (the closing quote is optional) after the LOAD command. For instance, to LOAD a program called MUMBO JUMBO, you would just type:

```
LOAD "MUMBO JUMBO"
```

The Commodore 64 will then cause the Datassette to scan through the entire tape, *starting at the current tape position,* until it either finds the program with that name, an "end-of-tape" software marker, or the actual end of the tape. If the program is not on the tape or you start the tape running at a place beyond the beginning of the program you want to LOAD, the program can't be LOADed.

When a program name is specified after a LOAD command, the Commodore 64 will search the entire tape for that program, if you let it. Every time it finds a program, it will note the fact on the monitor screen (as FOUND "PROGRAM NAME"), making a longer and longer list as it rolls through the tape. If the program is not found before the computer reaches the end of the tape or a special command recorded on the tape called an "end-of-tape marker," you'll get the error message FILE NOT FOUND.

If you suddenly realize that you're too far into your tape and you want to rewind it to the beginning, you can get it there by outsmarting your Commodore 64. When the monitor asks you to press PLAY, just press REWIND instead. When you reach the beginning of the tape, press PLAY. Your computer won't even notice your transgression against the rigid order of its instructions.

The Commodore 64 identifies programs on the tape by looking for a *header*—a special sequence of data codes that tells it that a program is recorded on the tape, the name of the program, and exactly where it starts. The computer will not recognize a program if it does not find the proper header (see Box 9).

Every time a program header is found on a tape, your

BOX 9: A LOAD OF TROUBLE

For safety's sake, the Commodore records two copies of every program that it SAVEs to cassette. When the program is LOADed, the two copies are compared. If they do not match (if there is an error on either one of them), you will get the error message LOAD ERROR. There's no way to retrieve either copy. Your only hope is to try LOADing again—and again—before cursing computers to the sky and putting your fist through the Datassette.

Commodore's microprocessor examines it, sends it to the monitor screen and compares it to the name that you requested the machine to LOAD. If the names match, the Commodore stops the cassette recorder and prepares to finish the LOADing process. If the names don't match, the Commodore will keep searching until it reaches an end marker or you tell it to stop by pressing the RUN/STOP key.

Putting Programs on Tape

Putting a copy of a program on tape is called *saving* it. Just type SAVE (and hit the RETURN key). Your Commodore 64 responds with the message PRESS PLAY AND RECORD ON TAPE. When you press those buttons on the recorder, your monitor screen will turn completely the color of the border. The Commodore will write (record) a copy of the program on the cassette in the recorder, provided of course you put a cassette into the recorder. You can't press down the RECORD button on the Commodore Datassette if no cassette is present in it!

After you've SAVEd a program, the original is left unscathed in your computer's RAM memory, and a duplicate is safely recorded on cassette. If you ever want to put a copy of the cassette version of the program back into memory, first NEW the Commodore's memory, rewind the tape to a position before the spot you started recording the program, and LOAD it back in.

The simple SAVE is hardly sufficient when you try to put several programs on the same tape. Once you put the cassette away and pull it back out later, you have no way of identify-

ing individual programs on the tape to LOAD them back into your Commodore 64's RAM. You could keep programs straight by putting each one on a separate cassette. The easier way is to identify each program on the tape by giving it a name. It's called a *file name.*

To specify a name for a program, just type the immediate instruction SAVE into your Commodore 64, followed by the name you want to give the program (inside quotation marks, of course, although the second quote is optional for most SAVEs).

On the Commodore 64, file names are limited in length to 15 characters, which includes all spaces and punctuation (except for the quote marks, which aren't really a part of the file name). Because program names are strings, a previously defined string variable can be used in place of the file name in the SAVE command (for instance, SAVE A$). SAVE can be used in either the Immediate or Program Mode of BASIC, meaning you can write a program that will SAVE itself!

If you want to SAVE a program called QUACKENBUSH 2, you just type:

```
SAVE "QUACKENBUSH 2"
```

then press RETURN. The computer responds with PRESS PLAY AND RECORD ON TAPE, and when you do, it *immediately* begins to record your program.

Immediately means that the Commodore 64 does not take the slightest glance at your tape to see what's there. If you have an important program already on tape, the Commodore 64 will happily record right over it without a second thought. On the other hand, if you just press PLAY and not RECORD or don't even have a tape in the Datassette at all (in which case you will only be able to press PLAY), your Commodore 64 will never know and will assume that it has SAVEd your program.

Obviously, it's important, then, to have your tape stopped and ready at the spot where you want to SAVE your program. One way to find the end of the last program on a tape is to LOAD the last program, after which the Datassette will automatically stop at the right place. Alas, you'll have found the perfect place to put your new program, but by LOADing another program into the Commodore's memory, you'll have destroyed the one you wanted to SAVE. Somehow that doesn't seem to be such a good idea.

A better solution is to use the BASIC command VERIFY,

which compares the next program on tape with the program in memory. If you just enter the word VERIFY, your Commodore 64 will tell you to PRESS PLAY ON TAPE. After you've done that, it will tell you the name of the next program on the tape (as FOUND WHATEVERITIS). Then the computer will compare the program on tape with the one in RAM and tell you that they don't match by saying ? VERIFY ERROR. (If they match, why bother SAVEing the program again?) The Datassette will then stop the proper distance after the program on the tape to preserve it safely when you record the next program.

For this command to be useful, you must have some idea of what programs are recorded on the tape so that you know the name of the very last one. VERIFY only checks one program at a time, and if you use it on any program but the last one on a tape, it will stop at the beginning of the next program. If you then SAVE, you'll erase that program, and maybe the next one, too. That's why it's a good idea to keep a record of what's on each and every cassette you use.

The SAVE command allows a few options to increase its versatility. You can optionally add a "device number" to the SAVE command. A device number unambiguously identifies a peripheral to the Commodore 64 so that you can be sure both you and your machine agree on what you're doing. The device number assigned by Commodore to the Datassette is 1.

Device numbers are most important when you are dealing with peripherals *other than* the Datassette. If you don't add a device number to your SAVE commands, the 64 will assume that "1" was what you had in mind. The only time you *have* to indicate the Datassette's device number is when you add a *secondary address* to your SAVE command. For instance, if you want to indicate to your computer that the program you are SAVEing is to be the last one on the tape, you add a secondary address of 2 *after* the device number. To make a program named KING CANUTE the last one on a tape, you would type:

```
SAVE "KING CANUTE",1,2
```

Note the position of the commas. The command would be invalid without them.

There are several valid secondary addresses. "1" indicates that the SAVEd program, when LOADed, should occupy the same location in memory as it did when it was SAVEd.

Otherwise, all programs are LOADed to start at memory address 2048. (Don't worry about it until you're a full-fledged assembly-language programmer.) The number 2 adds an "end-of-tape" marker to the cassette after the program is SAVEd, telling your computer there are no more programs on the tape no matter how much tape is left. A secondary address of 3 combines both functions.

If you have valuable programs (or files) that you want to preserve forever on cassette, you can keep them from getting accidentally erased by breaking off the *write protect* (or *record protect*) *tabs* on the back edge (the long, thin, tapeless side) of the cassette. There are two tabs on each cassette. With the front of the cassette facing you, the tab that controls writing on the side of the cassette in use is the one on the left.

When a tab is broken off (or when no cassette is in the cassette recorder) a mechanical interlock will prevent you from pressing down the RECORD button. Because it is a mechanical interlock, you will feel a great deal of resistance to your pressing down the button. Invariably, you are stronger than the mechanism, so if you press too hard, you're left with a broken cassette recorder. The moral is to think twice and look at the cassette you're using before severely mashing down the RECORD button.

If you've decided to preserve a program forever and then forever comes and passes you by, you can write on the cassette again. Simply cover the write-protect holes (where the tabs used to be) with a reasonably strong piece of tape or a self-adhesive label. In fact, the self-adhesive tabs used for write-protecting 5¼-inch diskettes do a great job of *un*-write-protecting cassettes!

Cassette Carbon Paper—Making Copies

The only difference between a computer cassette and the cassettes used in music recording is that (in general) the computer cassette is monaural and the music cassette is stereo. Owing to an amazing bit of prescience on the part of the Compact Cassette's inventors, mono and stereo cassette tapes are compatible and can be used interchangeably in either style of machine.

That interchangeability means you could play and record computer cassettes on your stereo system. They won't sound like much (they drone worse than bagpipes), but you don't

have to be particularly quick of mind to figure out that if you can play and record them on your stereo, you can copy them just like stereo tapes.

Computer types call making a copy of a tape or disk *backing it up*. To back up a computer cassette, all you need to do is have another cassette recorder. If you expect workable results, don't use a microphone. Hook the two together with a patch cord from the TAPE OUT jacks of one to the RECORD IN or TAPE IN jacks of the other. Play the computer tape you want to copy on the machine with the patch cord plugged into the TAPE OUT jacks. Record the copy on the other machine. For best results, do not use Dolby noise reduction on either the original or copy.

One warning: computers require that a very high level signal be recorded on the tape. You should make your computer tape copies with the recording level set at or just above the Dolby level marks (on both channels if you use a stereo machine). The automatic level control used on some machines should be defeated if possible.

Although copying prerecorded programs may be a violation of the copyright laws (meaning you shouldn't do it), you can and should back up all your important programs and all your important cassette data files. See Box 10 for another reason you might want to back up a cassette.

You can also copy programs that you write simply by LOADing them into your Commodore 64's RAM memory, then SAVEing the program on a different cassette tape.

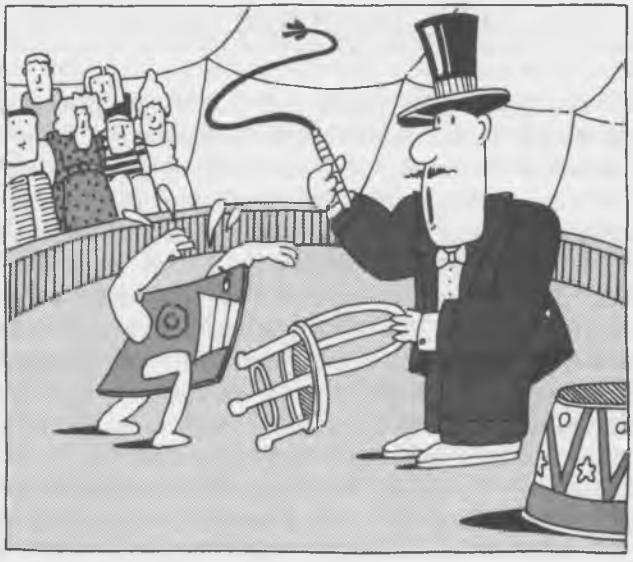
BOX 10: A SOLUTION FOR SPECIAL LOAD PROBLEMS

Many canned cassette programs are copy protected. Special signals are recorded on the tape that get mixed up with the normal program signals when the tape is copied. Sometimes copy protection works "too well," and when you try to LOAD the program, your Commodore gets confused.

In some cases, you *can* make a usable copy of these overly protected cassettes if you have a stereo cassette player. Many cassettes are copy protected by a wide track of confusing data recorded in the center of the normal width of the cassette tape. If you play back only one channel (usually the left) of the badly protected tape and record both channels on the copy using a Y adapter, you should be able to salvage the tape.

7

CONTROLLING THE MASSES: DISK DRIVES



A disk drive is a mass storage device. A receptacle for storing information, it's often compared to a filing cabinet and more often used as a wastebasket.

Mass storage is the place all data and programs are put while they are not immediately needed but still must be close at hand. It's both a reservoir for data that would overflow the RAM memory of the computer and a rest home for *files*—programs and collections of data.

But a disk drive is more than just a filing cabinet. It's more like a safe with a file drawer on the inside. Not only does it hold information, but it preserves and protects it as well. All the programs, formulas, and information that you enter into your Commodore's RAM memory are only there by the grace of the power company. If you haven't gotten around to saving a file and your lights wink at you due to a minor electrical interruption only a fraction of a second long, all the information that took you hours to type will wave good-bye and vanish.

Unlike the entirely electronic memory of your Commodore 64 itself, the disk drive relies on an entirely different storage principle: magnetism. Electricity is rather shiftless and pre-

fers to always be on the move, but magnetism likes to stick in one place. A magnetic field can be rather permanently anchored to metal particles; electricity usually drains off shockingly fast.

The Disk's Magnetic Personality

Disk drives work like most any other magnetic storage medium, which includes the lowly hi-fi cassette. The electrical impulses of the music in your hi-fi, or data in your computer system, are converted into pulses of magnetism in an electromagnet, often called a *magnetic head* or *read/write head*. This is the part of the system that does the actual recording. The magnetic field a head produces can induce permanent magnetic fields in a suitable storage medium—the magnetic-responding compounds that coat the surface of cassette tapes and computer disk drives. When the head's magnetic influence is removed, the field setup in the recording surface remains more or less permanently.

No matter how sophisticated magnetic storage devices may be electronically, they all (except magnetic bubble memories, which are not yet available for the Commodore 64) rely on some form of mechanical machine. The reason is simple. If the magnetic medium did not move in relation to the magnetic head, the head would try to record all of its data on the same single patch of material, and new information would be recorded or "written" over the old.

Which Way to Go

Once an engineer makes the obvious decision to move the storage medium so that different bits of information are recorded at different places, he faces a tougher choice: which way to move the marvelous magnetic medium and/or the magnetic heads. The possibilities are perplexing, more than just left to right or top to bottom, and include any and every direction that you can imagine—back and forth, in and out, and even around and around.

The cassette demonstrates one of the easiest solutions to the problem. The tape merely moves from left to right past the read/write head. More than simple (and hence cheap), the cassette mechanism is proven and has undergone twenty years of design refinement.

From a mass storage standpoint, the most important aspect of a cassette system is that data are stored one-dimensionally, in a straight line. If you opened a computer cassette and stretched out the tape, all the data would be arranged one bit after the other in a straight line for the length of the tape.

Certainly the shortest distance between two points is a straight line, but getting between two separated points on any line requires that you pass through every other point between the beginning and end points—the geometric equivalent of having to visit every bar you see along the way home from work. Both the geometric and inebriate journeys can take an agonizingly long time, especially if someone's waiting at the other end of the geographical region. Getting to one specific hunk of data from another requires wandering through every hunk of memory between the two.

Because cassettes must check each address one by one in sequence to find any particular byte of data, they are often called sequential storage devices. Information is stored in one continuous "stream," one byte after the other.

There's nothing wrong with sequential storage in itself. It can be very fast if you can find a way to move from one storage point to another very fast. All-electronic "shift registers" move data sequentially at nearly the speed of light. Cassettes, and most other practical mechanical mass storage systems, however, are not fast. Cassette tape meanders through the machine at a rate of less than 2 inches per second. In computer terms, that amounts to only (only!) hundreds of characters (or bytes of data) per second. In human terms, that means when you need a program at the end of a tape that's currently at the beginning (tens of thousands of bytes away), you'll have lots of time on your hands.

Disk drives organize data differently. Data are stored on the large flat area of a disk's surface in *two* dimensions, like the entire area encompassed by a circle. Although the shortest distance between two points (or two bytes) is still a straight line, shortcuts across the width of the circle can move you from one byte to another in a hurry. Because the recording head can jump from byte to byte at widely separated locations on the disk surface and because data can be read and retrieved in any order, such memory units are often called *random access,* just like RAM memory chips. Hence, disk drives are often called random-access devices.

In any two-dimensional storage system the read/write head must move in two dimensions in relation to the storage

medium. In reality, it's usually simpler to move the recording medium one way and the read/write head in a different direction (usually perpendicular). A disk drive rotates its disk in one dimension and moves its head back and forth across the surface of the disk in the other dimension.

Finding the Way

With random-access devices like disk drives, new problems arise. Not only do the head and medium move in different directions, but the machine must know where everything is. The disk drive must know where to put data and where to find it.

Disk drives use complex schemes to set up signposts and other markers to find their way around a disk. The disk surface is magnetically divided into short segments called "sectors," each of which is assigned a number, similar to the addresses assigned in RAM memory.

Some disks use holes punched in the surface to indicate where each sector begins. Because the location of each sector is determined by the hardware (the disk itself), such disks are called *hard sectored*. The Commodore 64's disks, on the other hand, are *soft sectored*. Only one reference hole is punched in the disk, and the locations of the various sectors are determined by markers magnetically written on the disk surface by the disk-drive system's software.

Information is written and retrieved one sector at a time. In the Commodore 64 disk subsystem, each sector holds 256 bytes. That means disk drives can randomly access *sectors* but cannot randomly access single bytes of data. Sectors are short enough, however, that looking through one sequentially to find any given byte takes just a tiny fraction of a second. Sectors are arranged one after another (sequentially, again) to form individual *tracks* on the surface of the disk. Unlike the most familiar rotating storage device, the phonograph record, which stores information in a spiral groove created by smooth movement in both the round-and-round and back-and-forth directions, the computer disk drive moves its head back and forth in individual steps. Its tracks of data are therefore concentric circles.

Different brands and models of disk drives use different arrangements of tracks and sectors called *disk formats* for soft-sectored disks. Although many popular disk formats put the same number of sectors in each track, the usual Commo-

dore 64 disk format puts different numbers of sectors in different tracks. The tracks nearer the outer rim of the disk have more sectors because the outer rim has more room for them.

Putting different numbers of sectors in different tracks offers a big advantage: more information can be recorded on a disk for a given recording "density." So-called *single-density* 5.25-inch disks normally cannot handle more than 100K bytes of data per side and usually only hold 70K–80K bytes. A Commodore 64 format single-density disk holds 170K, as much as many *double-density* disk drives, which require more costly double-density disks.

There is another limit to the storage capacity of a Commodore 64 disk. Each can hold no more than 144 different programs and/or files, no matter how tiny they are.

Every sector on a disk can be recorded, erased, and recorded separately. Even a single byte in a sector can be changed: the computer reads the sector containing the byte into RAM memory, changes the byte in RAM, erases the original sector, and rerecords the changed contents of the RAM memory. It all happens a lot faster than it takes to describe it.

When you start to use a disk drive, you'll discover you can change a program or file on a disk whenever you want. You can even take the first file you recorded on the disk and make it longer. Ordinarily, squeezing more information into the same memory area would be like trying to put a quart of milk into a pint bottle.

Disk drives solve this rather obvious problem by dividing large masses of data into smaller *blocks,* one of which can be written into each disk sector. The different blocks of one file or program can be recorded into sectors scattered almost randomly all over the surface of the disk, wherever there is blank space to write them. Ordinarily, the result of random scattering would be chaos, but disk drives keep track of which of their sectors are related (and how they fit together). The first 2 bytes of data of each block record in a sector code the location of the sector where the next related block of data is recorded. Each block tells where the next one begins.

That leaves the problem of finding the first sector and block of each record. A special file called the *directory* contains the names of all the records on the disk and the sector location of the first block of data.

One problem remains. To determine whether a sector is blank or used (so you don't record new information over old

information), the disk drive would have to read every block and see if it were linked to another as part of an existing record. A special file on the disk takes care of this. It's called the *Block Availability Map* or BAM, and it keeps track of used and unused sectors. Before the disk drive attempts to write in any sector, it checks the BAM to see if that sector is available.

Using Disk Drives

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

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

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

Once you understand this, using it is simple. Just add one additional number to the cassette commands, and your data goes to disk!

Ah, but won't you be surprised! Everything will seem to go all right when you save that monster program that took you two days to type in and get running. But the next day, when you turn the machine on, you may discover that your supposedly safe disk copy of the program does not exist! Lots of things can go wrong with the simple process of saving programs to disk, and your Commodore will do little to warn you about them.

Formatting Disks

Perhaps the most mysterious and necessary process to making your disk drive work is often called *formatting*—the func-

BOX 11: THE NEW, INITIAL FORMAT AND OTHER NONSENSE

Many terms are commonly, and confusingly, used to describe the format process. Some computer systems call it "initializing" the disk. That's fine, except that the Commodore 64 has a separate function called "initializing," and it's entirely different from formatting! Commodore calls the formatting process "newing" the disk. Besides sounding more like something your cat might do, it can easily be confused with the BASIC command NEW. To avoid any more confusion than has already been caused, future references to the process in this book will use the most common term: "formatting."

tion that puts the all-important format signposts on your disk. (The process also has other names: see Box 11.)

Blank disks as they come out of the box are exactly that. Blank. No programs. No noise. No nothing. That should be good, since you have all sorts of room for your programs.

To your Commodore disk drive, however, a blank disk is as good as no disk. In order to store information, it requires prerecorded format signals on the disk to help it keep its place and find its way around and between the disk's 683 recordable sectors. Blank disks do not have format control signals recorded on them and are often called "unformatted" disks. If your disk drive does not find the signposts it expects, it gets lost and frustrated and refuses to write or record anything.

The disk drives of different computers use different electronic formats for their disks even if the disks are physically the same. Commodore's 1541 Disk Drive format does happen to be compatible with Commodore 4040 and 2031 disk-drive media. But although some Commodore disk formats are compatible, the Commodore 64 may not be able to RUN programs written for a different-model Commodore computer. And disks formatted for a different brand and model of home computer will not work with your Commodore 64 unless your computer reformats them, thereby wiping out all the information they hold.

Formatting a disk *erases all information stored on that disk.* You cannot transfer programs or information between different brands of computers by reformatting. The programs you'd

want to transfer would be destroyed in the process. Unless you want to erase all the data you have stored on a disk, do *not* format it again once you've SAVEd programs or files on it.

Before you can use even the simplest SAVE "FILE NAME",8 command you *must* format the disk you want to use. Obviously, the first thing you'll want to do with any new disk is record these control signals onto it. You can also format previously used disks, or even disks that have been previously used with other machines, to recycle them for use on your Commodore 64, so long as they are soft sectored.

In the Commodore 64 scheme of things, when you format a disk, you also give it a name (up to 15 characters long) and a two-digit identification number. Your computer uses the ID number to verify that you're using the right disk when you shift disks around within a program.

Commodore, contrary to most of the computing world, calls the disk-formatting command NEW. To format a disk, simply type in the following easy-to-remember (!) command into your Commodore 64:

```
OPEN 15,8,15,"NEW0:[DISK NAME],[2-digit ID
     number]":CLOSE 15
```

As with all BASIC commands, it is mandatory that your punctuation, commas, colons, and quote marks be *exactly* as shown in the example!

Every command to the disk drive must be routed through the Commodore 64's serial port. Part of each command must tell the Commodore 64 to route the proper instruction to the serial port so that it can be sent to the disk drive.

The command, or any information, to be sent to the disk drive must be contained in a file—that electronic equivalent of an envelope to be delivered to the address of the disk drive. In the formatting command, the word OPEN instructs your Commodore to OPEN a file for the disk-drive command. The first number following the OPEN identifies the file so that information can be put in the right electronic envelope.

File numbers can be any number between 1 and 255. However, using file numbers greater than 127 will cause extra blank lines to appear between lines of programs or data retrieved from the disk. Most programmers use file number 15 for disk-drive commands, because that's the same number as the disk-drive command channel, and it makes one less number to remember.

The second number is the address on the envelope, telling the Commodore where the information or command is bound so that it is sent to the correct peripheral on the serial bus (which might host up to five disk drives and two printers). As shipped from the Commodore factory, disk drives are always assigned the device number 8. If you have one disk drive and do not alter its device number, whenever you OPEN a file bound for (or coming from) your disk drive, the second number must be 8. If you buy one or more additional drives, each one must be assigned a separate device number to avoid confusion. You may then route commands to the proper drive by substituting its device number for the number 8 given in the previous commands.

The final number identifies the channel to be used. One channel, number 15, is reserved for the sending of commands to peripherals and hence is called the "command channel." Commands to the disk drive must be on channel 15 for the drive to act on them. Information sent to be stored on disk, however, may use any channel from 2 to 14. Channels 0 and 1 are used by your computer to SAVE programs from its memory to the disk drive or to LOAD programs from the disk drive into the computer's memory.

The actual command to the disk drive is handled by your computer as a string. That can mean two things. First, the computer always keeps it intact and does not evaluate or manipulate it unless specially commanded to do so. Most importantly, the command itself must always be enclosed in quotation marks.

The first letter or word after the first quotation mark is the actual command. It is a special code that is recognized by the disk drive and causes the drive to take specific action. The command word or letter is always followed by a drive number.

What's that you're saying—haven't we already identified the disk drive with the device number 8? In fact, we have. Commodore has chosen to make its disk command compatible with those used for its larger computer systems, and those systems require a drive number to be specified following the command. Perhaps it's redundant and illogical (the two drive identifying numbers don't even agree), but it *is* Commodore. Suffice it to say that if you have a single disk drive, the number 0 and a colon (:) must always follow this command letter/word.

A typical formatting command you might make would look like this (assuming your name is Bilbous Noser):

```
OPEN 15,8,15,"NEW0:BILBOUS NOSER,01":CLOSE 15
```

NEW may also be abbreviated as simply the letter N with the same results. You could save two whole keystrokes by typing in:

```
OPEN 15,8,15,"N0:BILBOUS NOSER,01":CLOSE 15
```

You could save several more keystrokes by changing your name to Al Lu or by simply giving each disk an identifying number instead of a name. However, if you omit typing anything for the disk name *or* if you omit giving an identification number to a disk, *it will not format properly.*

When you type RETURN after any of the above commands, the red LED (a Light-Emitting Diode, a tiny indicator light) should glow on the front of the disk drive, and the drive's motor should switch on and whirl. In a second you'll hear a quick rattle as the read/write head inside the disk drive puts itself in the proper position to start recording the formatting signals. About every five seconds thereafter you'll hear a "click" as the head advances to record the formatting information for the next track.

In a little over a minute, the machine will quiet down, and the red LED will go off. That means everything went well and your disk is now formatted.

If the red LED remains flashing after the disk drive quiets down, however, some error has occurred in the formatting process, even if no error message appears on the monitor screen. In that case, your disk is NOT safe to use to store programs or data. Check to be sure your formatting command exactly matches the one shown above, then try to format the disk again.

Some Commodore 64 programs have built-in "utility" subprograms that allow you to format disks much more easily when the application program is running. Commodore's *Easy Script* word processor is an example.

If you don't care what name you give a disk and just want to get a disk formatted so that you can start to use it to store programs, you can use the PERFORMANCE TEST program provided on the *Commodore VIC-1541 Test/Demo* disk packed with every Commodore 64 drive. Not only will it format any usable disk; it will also check out the entire disk-drive mechanism. The procedure is a bit more complex mechanically but will help you get started if you have problems

convincing your computer to accept your typing as commands in BASIC.

First, put the *Test/Demo* disk in your disk drive. Then enter:

```
LOAD "PERFORMANCE TEST",8
```

When the red LED on the front of your disk drive stops glowing or flashing and the Dreaded Ready appears on your monitor screen, remove the *Test/Demo* disk from the drive and insert the disk you'd like to format. Make sure that the *Test/Demo* disk is not still in the drive (so you don't accidentally erase it). Then, and only then, type RUN. If you still haven't replaced the *Test/Demo* disk with one that you don't mind erasing and formatting, you'll be given a second chance. The program asks you to insert a "scratch" (blank) disk. If you've already done that, just press RETURN. If you haven't already done it, *do it!*

After you do press RETURN, the disk drive will whir and send strange messages to the screen. If the gods are smiling on you, the messages will indicate all is well with the drive and that it has passed a number of performance tests. After about 80 seconds, the whole process will come to a quiet halt. Your new disk will now be formatted and electronically labeled TEST DISK. More importantly, it will be ready to use to store programs.

If your system does not pass the performance tests, first try it again to be sure there is a real problem. Turn off your entire system, then turn it back on again before your second test just to be certain that no strange commands are floating around in memory. If your drive still does not pass the performance test, you may have a bad disk or bad disk drive. First try the test on another blank disk. If the second disk doesn't make things work, you may have a shoddy disk drive. A return visit to your dealer is in order.

SAVEing Programs to Disk

Once a floppy disk is formatted, you can use it for SAVEing programs almost exactly the way you would use the Commodore Datassette. But the disk drive is much, much faster than cassette units and requires slightly different commands. If you have only one disk drive and have not changed its device number, the command to SAVE any program to disk is simply:

```
SAVE "[program or file name]",8
```

The name of the program can be virtually anything you want that will identify the program to you so long as it is 15 or fewer characters long. Don't forget that your Commodore counts a blank space as a character. Also, the program name must be enclosed within quote marks—otherwise, your command will cause a TYPE MISMATCH error message.

Again, the 8 in the command is the device number of your disk drive, and it merely indicates to your computer where you want your program saved. If you fail to specify a device number, your computer will attempt to save the program to cassette. If you specify the number of a device that's not attached to your computer or turned on, you will get a DEVICE NOT PRESENT error.

Commodore has tried to catch all the mistakes you might make in using the SAVE command, but currently the 64 does not warn you of some errors that you might make that can be potentially fatal to your program and your sanity. If, for example, your drive is turned on but does not have a disk in it or has an unformatted disk inside it, your computer will run through the complete SAVE routine and think that it has properly SAVEd your program, which it can't have without a proper disk! The only clue you will get is the flashing red LED on the front of the disk drive. When you finally try to load the program, you'll discover only the unfriendly error message FILE NOT FOUND.

If your imagination runs dry and you attempt to give the same name to a program that's already on the disk or if you update a program and want to save it under the same name as the old version, the red disk drive LED will flash. When you LOAD the program, you will get the first version or first program SAVEd under that name. The second, later program will be gone forever.

To avoid such problems, Commodore provides another disk-drive function, SAVE AND REPLACE, which is just a variation of the general SAVE command. To replace an existing file with a new program or version of the same program with the same name, type:

```
SAVE "@0:[file or program name]",8
```

The "@" indicates the SAVE AND REPLACE function.

LOADing a Program from a Disk

Once you've successfully SAVEd a program to disk, it is useful to be able to retrieve it again. The proper command to LOAD a program from disk into your Commodore 64's RAM memory where it can be RUN is the following:

```
LOAD "[program or file name]",8
```

The program name you request must match one you've previously used to SAVE a program on the same disk. The 8 is, once again, the device number of your disk drive. If you set your drive to a different number, substitute that number instead of 8.

Although improper typing or use of the LOAD command cannot destroy or even injure programs that you have previously stored, any variation from the computer's required format will generate an error message. TYPE MISMATCH usually indicates that you forgot the quote marks around the program name. FILE NOT FOUND indicates that either you mistyped the program name so that it does not *exactly* match the name you originally gave the program *or* that you have the wrong disk (or no disk at all) in the disk drive. The cure for either error is merely to correct your command and/or retype it and/or find the proper disk and put it in the machine. (If you can't remember the name you gave a file or program, something called "pattern matching" can help you find it. See Box 12.)

Once you've LOADed and RUN the C-64 WEDGE program that's on the *Test/Demo* disk you get when you buy the C-1541 disk drive, the command for LOADing a program from disk can be shortened to simply:

```
/[program or file name]
```

No quote marks or trailing 8 are necessary! You'll learn about that time and lifesaver, the C-64 WEDGE, shortly.

The Directory—See What's on a Disk

One reason for using a disk drive to store programs is that the memory of a disk is probably better than your own. After you've been playing with programs for a while, you'll have

BOX 12: WILD CARDS AND PATTERN MATCHING

Occasionally, you'll forget the precise name you gave a program or a file. Other times you may want to erase more than one file and don't want to type a separate command for each erasure. The Commodore 64 Disk Drive's "wild cards" can save wear and tear on both your fingers and temper.

A wild card is simply a symbol that can represent any character or group of characters in a file or program name. When a wild card is encountered, the Commodore 64 uses pattern matching to find all the file or program names that match the wild card name. In theory wild cards could work with any command, but in reality they are only useful when you want to SCRATCH multiple files or LOAD a program the name of which you cannot precisely remember.

Using a question mark (?) as part of a disk-drive command file name tells the Commodore 64 to match any letter with the symbol. Similarly, an asterisk (*) means to match any group of letters. Hence, the file name F?O? will match FROG, FOOT, FOOD, but not FLAP or TROG. And "G*" will match GEEK, GOOD, GONDOLA, and any other legitimate file name beginning with G.

The Commodore 64 will ignore any letters following the asterisk, so G*T will match GEEZER, and *TWIDDLE will match *any* file. That means if you command the Commodore 64 to SCRATCH *BLOAT, you will SCRATCH every file and program on your disk. Be careful.

Because you can LOAD only one program into memory at a time, the Commodore will LOAD only the first file that it finds matching the pattern given when it is instructed to LOAD a program containing a wild card.

dumped dozens of them onto your disk collection. About that time you'll begin to discover you don't know what is on which disk—or even sooner, after a flurry of FILE NOT FOUND errors, you'll discover you can't remember the exact name you gave to a program. Fortunately, the disk drive keeps a record of the names of every program and file that it writes on each disk, and it will willingly display this information for you. This record is called a directory because it keeps track of all the files on the disk by name, as well as in-

formation that the disk drive uses to locate each of the programs listed.

The directory is recorded on track 18 of your disk drive and can be LOADed into your Commodore 64's RAM memory as if it were a program. Of course, it is not a program and will not RUN, but it will LIST and show you the name of each file and program on the disk, its length and type, the name of the disk itself, and how much space is left on the disk to store programs and files.

To put the directory into your computer's memory, simply type:

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

A FILE NOT FOUND error message indicates that your drive is not turned on, has no disk in it (or one that's not properly inserted in the drive), or the disk inside the drive has not been properly formatted. But if you get no error message, all you have to do is type LIST to see your directory on the screen.

Although LISTing the directory in this way has no effect on the programs stored on the disk, it will have some unwanted effects on whatever program you have in your Commodore 64's RAM memory and your view of the directory itself. The directory will be LOADed over the existing file and may erase your previously existing program or get mixed in with it. File lengths are likely to be hopelessly confused with line numbers, and the programs will be mixed in the directory like a bad shuffle of a deck of cards. If you have a program in RAM memory, the solution is to SAVE the program, then use BASIC's NEW command before LOADing the directory.

An alternative to the possible chaos of program and directory mixing is to use the C-64 WEDGE program provided on the *Test/Demo* disk that is packed with the Commodore C-1541 disk drive. This program adds a new directory function to your computer. When using the C-64 WEDGE, you can call up the directory with neither confusion nor injury to your program. When the C-64 WEDGE program is running and the appropriate command is given, the directory and the Wedge program itself are stored in a part of your computer's RAM where they won't mix with your existing program and can be displayed without upsetting the program you have in the Commodore's RAM. The command to see the directory *once the C-64 WEDGE is properly running* is simply

```
>$
```

or

```
@$
```

To Erase a File—SCRATCH It Out

If you couldn't erase files or programs, they'd soon be multiplying faster than rabbits or software companies. Soon all your disks would be filled with a confusion of worthwhile and worthless programs, some that you'd like to never see again and some that you could never find. Commodore provides a program and file-erasing command called SCRATCH. Command your Commodore to SCRATCH a file and you'll never be bothered by it again. But be careful. Resurrection is a problem better handled by clergymen than neophyte programmers.

To erase a file with the SCRATCH command (assuming that you've already OPENed a file and the command channel to the disk drive) type:

```
PRINT#15,"SCRATCH0:[program or file name]"
```

or

```
PRINT#15,"S0:[program or file name]"
```

If you don't have a file and the command channel OPEN to the disk drive, type OPEN 15,8,15 *before* the SCRATCH command. Otherwise, you'll get a FILE NOT OPEN error message.

Backing Up for Protection with COPY

Because disks can be damaged easily, it's a good idea to keep a backup copy of important programs and files. The best place to put a backup file is on a totally different disk that can be kept in a safe place separate from the original. The Commodore operating system that is built into disk drives includes a COPY function that can duplicate files with a single command. However, if you have only one disk drive, the COPY command won't help you much. It allows you to duplicate files onto the *same* disk as the original file only. If

for some reason you decide to make backup copies of programs or files on the same disk, make certain you give the duplicate a name different from that of the original. Having two files with the same name on one disk would boggle the stodgy mind of your Commodore, so the 64 won't let you do that.

To make a duplicate of a program or file on the same disk using the copy command, type:

```
PRINT#15,"COPY0:[new name]=0:[old name]"
```

or

```
PRINT#15,"C0:[new name]=0:[old name]"
```

A typical command might look like this:

```
PRINT#15,"C0:GOLD=0:LEAD"
```

If you have invested in two (or more) disk drives, you can copy all of the programs and files on a disk to another disk (which need not even be formatted first) by using the COPY/ALL program included on the *Test/Demo* disk packed in the C-1541 box. To use the program, first set the device numbers of each of your drives to different values. Then LOAD the COPY/ALL program by typing:

```
LOAD "COPY/ALL",8
```

and RUN the program. All you need to do is answer the questions the program asks. Be sure to replace the *Test/Demo* disk with the one you want to copy *before* running the program! If the disk you want to copy *to* has never been formatted before, the program will ask if you want to "new the output disk." Answer Y.

You can make backup copies of important disks even if you have only one disk drive by using the *1541 Backup* program included in the *Commodore Disk Bonus Pack*. This set of utility programs is often given away as a premium when you buy a Commodore 64. *1541 Backup* will format the disk it puts the backup copy on, so you can use a brand-new, unformatted (or dingy, old, used, formatted) disk for your backup copy.

Version 1.0 of the program, however, is very perplexing. When you LOAD and RUN *1541 Backup,* you get a white

screen with blue lettering and boxes and not a hint of how to proceed.

The first thing you should do is type either B or D. The difference is that B copies only the parts of the disk where you have written programs or files. D copies the whole disk down to the last detail. The biggest effective difference to you may be that D takes longer, particularly if very little of the disk's capacity is used.

The next step is to type in the name you want to give your disk copy. Press RETURN when you're done with the name, and the program automatically adds a comma and waits for you to type in a disk number (and type another RETURN).

Once the naming is done, put in the disk you want to use as your backup and follow the screen instructions in the box marked "Operator Intervention." Prepare to shuffle disks a number of times and spend a long time doing it.

At this point, the program is almost self-explanatory and quite goof-proof. It won't let you get your disk shuffling out of sequence. If you don't know what to do next, try pressing the RETURN key.

Combining or Concatenating Files Using COPY

Sometimes two is not better than one. Perhaps, while concentrating on solving all the world's problems through computing, you built two files that by rights should be combined together into one bigger file. Using built-in disk functions, you can make one big file out of two or more little ones (and still have the little ones left over to SCRATCH away when you please).

Programmers often use the term *concatenate* to mean the process of combining files. The word combine will do, however. To combine files, just use COPY as you would to duplicate a file, but string together the old files that you wish to combine by putting commas between them. Once file number 15 has been opened, a command to combine three files would be typed like this:

```
PRINT#15,"COPY0:[new file]=0:[old file 1],
     [old file 2],[old file 3]"
```

or

```
PRINT#15,"C0:[new file]=0:[old file 1], [old
     file 2],[old file 3]"
```

A typical (albeit a bit unusual) command might look like:

```
PRINT#15,"C0:SANDWICH=0:BACON,LETTUCE,TOMATO"
```

Changing Program and File Names with RENAME

Sic transit gloria and all that. You thought you had a perfect name for your program, and you get a certified letter in the mail that WordMangler has already been registered as a trademark and the Undocumented Software Empire is threatening to sue you for an arm and a leg and the twelve dollars left in your savings account. What to do? Change the name of your program on the disk using the Commodore disk drive command RENAME! It's simple once you've figured out how COPY works. After you've opened file 15, just type:

```
PRINT#15,"RENAME0:[new name]=[old name]"

PRINT#15,"R0:[new name]=[old name]"
```

or, to use simulated program names:

```
PRINT#15,"R0:FROG=TADPOLE"
```

If All Else Fails After an Error, INITIALIZE

Usually, disk drives are understanding machines. When you make a really big mistake, instead of shouting your stupidity for the whole world to hear, they just sit there, flashing their red error LEDs and silently saying, "Dummy! Dummy! Dummy!" to themselves. Sometimes, though, one of your transgressions will raise the hackles of this otherwise reliable machine, which will then refuse to accept another command. It will just sit there, flashing at you. Before the machine can do anything else, it must have the error condition reset. You can do it two ways.

One will cause you to reel back in horror. Turn off the entire computer system, including the disk drive, then turn it back on again. The reason for the horror is easy to understand. Usually, the command your disk drive won't accept is SAVE; of course, turning off the computer will wipe out its memory, eliminating the need for the SAVE in the first place. So much

for that solution to the problem. Although Commodore does not advise that you do so, you can switch off the disk drive alone, then switch it back on. This will clear some error conditions, but it could possibly hurt your computer system.

The better alternative, and the one recommended by Commodore, is to *initialize* your disk drive. In other words, put it back in the condition it was in when you turned it on. You do this by using the (you guessed it) INITIALIZE command. This command simply makes the disk drive *think* that it has been switched off and back on again.

After opening good old file number 15, to INITIALIZE your drive and reset an error, type:

```
PRINT#15,"INITIALIZE"
```

or

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

Making More Disk Space with VALIDATE

In general, the Commodore 64 is lazy and only looks in a few likely locations to squeeze in new blocks of data onto a disk. After you've used a disk for a while and have SAVEd and SCRATCHed programs and files dozens of times, there may be some empty blocks or sectors that the disk drive lost or just can't find. That means valuable disk real estate will be wasted.

The VALIDATE command causes the disk drive to reorganize the files on a disk and squeeze out all the blank blocks. It also wreaks havoc with the "random" files that some programs (including your own) can create, possibly making whole sectors or whole files disappear. In other words, you must be very careful using this command. If you know you do not have any random files on the disk in question, you can use VALIDATE to squeeze extra memory out of a disorganized disk. If you have one or more random files on the disk or don't know whether you have them or not, *do not* use the VALIDATE command.

To VALIDATE a disk, be sure file number 15 is open and simply type:

```
PRINT#15,"VALIDATE"
```

or

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

The C-64 WEDGE—Operating the Commodore Operating System

If you haven't noticed, using the Commodore disk drive is not the easiest thing in the world to do. Your fingers could get cramped or arthritic just looking at the directories of half a dozen disks. Most computers use very simple commands to read their directories, usually just a few letters—not quote marks, commas, and all that other shiftiness that makes typing even a simple Commodore disk command a creative writing lesson in itself.

Engineers at Commodore are not dummies, however, and the Commodore people created a system that simplifies disk commands. It connects the Disk Operating System (or DOS) built into the disk drive with the computer using a special program that has an amazing property. Instead of being LOADed into the first available slots in your computer's RAM memory, the program takes a hike and "wedges" itself near the top of the RAM and protects itself if you shuffle other programs in and out of the RAM. Even a NEW command, which erases other BASIC programs from your Commodore 64's RAM, won't hurt the WEDGE program. It's nearly invisible (after it's RUN once it doesn't LIST), and you'd never know it was there if it didn't make life so much easier.

To use this special program, you LOAD and RUN the C-64 WEDGE program (from the *Test/Demo* disk supplied with your C-1541 disk drive) by inserting it into your disk drive and typing:

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

then typing RUN when the Ready returns.

The miracle that makes your life easier is *not* the C-64 WEDGE program, however, but the DOS 5.1 program on the same disk. The C-64 WEDGE program just copies the DOS 5.1 file off the disk and tucks it safely away high up in RAM. To demonstrate this for yourself, LOAD the C-64 WEDGE, then pull the *Test/Demo* disk out of the drive. If you then ask the WEDGE to RUN, it won't, because the file it needs to LOAD is nowhere to be found.

When DOS is running, your Commodore will become much smarter when it comes to recognizing disk-drive commands. The greater than sign (>) and the at sign (@, which has its own key and thereby avoids the big shift) are then recognized as

disk-drive commands and automatically open the command channel and route what follows to the disk drive. Your Commodore will now recognize disk file names without quote marks around them. The only thing you must remember to do is to tell the machine which disk drive the command is meant for when using the COPY and SCRATCH commands. In other words, your Commodore 64 got smarter but still cannot read your mind!

The C-64 WEDGE also lets you read error messages from the disk drive. That means if the red LED on the front of your disk drive flashes for some unknown reason, you can type a simple command (either @ or > followed by a RETURN), and a message will appear on your monitor screen. You'll be told what your disk drive *thinks* is wrong—whether there is a problem with the disk and/or drive itself, or whether you merely made a typing error.

In sum, the familiar disk commands become the following:

DIRECTORY becomes	`@$`
	or
	`>$`

SCRATCH becomes	`@0:[program or file name]`
	or
	`@0:ALL GONE`

COPY becomes	`@0:[new file]=0:[old file]`
	or
	`@0:GOLD=0:STRAW`

RENAME becomes	`@0:[new name]=0:[old name]`
	or
	`@0:BUTTERFLY=0:CATERPILLAR`

INITIALIZE becomes	`@I`
VALIDATE becomes	`@V`
LOAD becomes	`/[filename]`
READ ERROR MESSAGE	`@[RETURN]`

Changing a Disk Drive's Device Number

If you invest in more than one disk drive to use with your Commodore 64 computer system, it is important that each drive have a separate and different device number. If you try

to use two disk drives that both have the same device number, neither one is likely to function properly.

A disk drive's device number can be changed temporarily through a software command or permanently through a simple modification to each drive's internal circuitry.

If you want to assign separate device numbers to your multiple drives, it is imperative that you turn them on one at a time. First, turn on one disk drive, then send the appropriate command to change its device number. Then turn on the next drive, and issue the appropriate command to change its device number. If you had two drives running simultaneously (and both with the same device number), both would recognize your first command to change the number, and *both* would change to the new number, defeating your purpose in making the change.

The command to change the first drive to device number 9 is:

```
OPEN 15,8,15,"M-W:" CHR$(119) CHR$(0)
     CHR$(2) CHR$(41) CHR$(73): CLOSE 15
```

The general command to change a disk drive's device number from the X hardware setting to a new device number Y (suggested values: 8, 9, 10, 11, and 12) is:

```
OPEN 15,X,15,"M-W:" CHR$(119) CHR$(0)
     CHR$(2) CHR$(Y + 32) CHR$(Y + 64):
     CLOSE 15
```

Yes, all of that has to be typed in without error every time you turn on your computer system. If it seems like a bit much to you, there is a permanent solution to changing the device number of a C-1541 Disk Drive. The drive determines its own device number when it is turned on by sensing the setting of two *jumpers* (small connecting wires) inside its case. If both jumpers are intact, then the drive knows it's supposed to be device number 8.

To change a disk drive's device number permanently, you merely cut one or both of these jumpers.

Cutting only the jumper marked 1 will change the drive's device number to 9.

Cutting only the jumper marked 2 will change the drive's device number to 10.

Cutting both jumpers 1 and 2 will change the device number of the drive to 11.

To find these jumpers and make the changes, follow the instructions below:

1. Turn off your computer system, including the disk drives, and disconnect all cables from the drive unit itself.

2. Lay the drive upside-down on a tabletop and remove the four screws holding its case together.

3. Turn the drive over, putting your hands around the drive so that it does not come apart. Then remove the top of the case and set it aside.

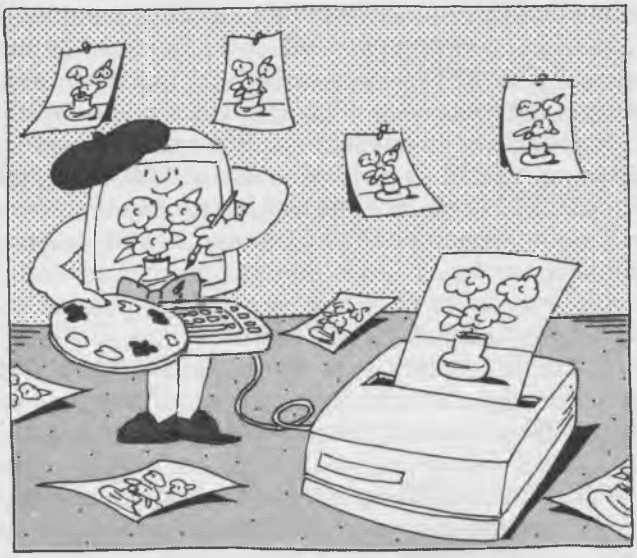
4. Remove the two screws on the side of the internal metal housing. Remove the housing.

5. Orient the drive so that its front (the drive slot) faces you. The device number jumpers are on the left edge of the printed circuit board, near the middle. Cut the appropriate jumpers.

6. Reassemble drive in reverse order of disassembly—replace metal housing: screw it in place; replace the top of the case; invert the drive (carefully) and screw in all four screws to hold the whole thing together. Connect it up and turn it on.

8

THE COMMODORE PRINTERS



By itself, the Commodore 64 is an actor without much of an audience. Certainly it can tell tales of dismay and delight about your home budget or your income tax or even write your most important letters and homework assignments across its screen. But it keeps everything it does a secret, privy only to you and the select few of your friends and family who can huddle around the pale glow of the monitor.

A printer can change all that, taking the elusive and volatile letters and graphics off the screen and making them permanent, pass-around, printed documents. Why else is it called a printer?

Commodore offers several different printers, and a plotter/printer, that are all directly compatible with your Commodore 64: the VIC-1515, VIC-1525, MPS-801, and C-1526 printers and the VIC-1520 printer/plotter.

The most popular Commodore printers, the VIC-1515, VIC-1525, and MPS-801, use exactly the same mechanism. The only difference is that the VIC-1525 and MPS-801 can use wider paper, 8.5 inches (once the perforated edges are removed) a full 80 columns across. The VIC-1515 allows only 40 columns. Not only are the VIC-1525 and MPS-801 more

desirable, but you'll discover one of them is likely to be the only Commodore printer your dealer carries.

The difference between the VIC-1525 and MPS-801 looks obvious but is more subtle. The MPS-801 is the newer model, and it has a sleek new case. But if you dig inside that case, you'll find essentially the same mechanism as operates the VIC-1525. Commodore has added an extra serial connector to the back of the MPS-801 to give you more versatility in setting up your computer system, but the meaningful differences between the two models stop there. Since any description of the operation of the VIC-1525 will be essentially the same for the MPS-801, we'll concentrate on the more common of the twosome, the VIC-1525. Assume that all references to it in this chapter apply to its younger brother as well.

The more expensive C-1526 earns its higher price by printing much faster, about three times the rate of either lesser machine. The characters it makes on paper have a crisper, sharper, and more detailed look to them. Those in the know will immediately recognize that there are more dots in its dot matrix.

The VIC-1520 printer/plotter can draw smoother and clearer pictures than any of the other Commodore printers and with proper manipulation of its drawing pens can produce images in four colors. As a printer of text, however, it's a sluggard, painstakingly drawing each character like a laboring draftsman.

Should you go out and investigate printers, the VIC-1515 and VIC-1525 will soon look very familiar, especially since they appear in various guises and disguises. A Japanese company called Seikosha makes the basic mechanism, which is also available under several other names, perhaps the most familiar one being Gorilla Banana.

These printer clones are easy to spot because they all look alike, with the same basic shape, the same smoked plastic cover, and the same tiny paper-advance dial on the right-hand side. However, be aware that all these similar-looking printers are not the same.

The Commodore duo of Seikosha printers (as well as the other Commodore printers) have ROM integrated circuit chips inside them containing the special codes needed to print the full Commodore 64-character set. Other brands of printer will not have these ROMs and will not print all the characters you see on the screen. Other brands of this print mechanism use different *interfaces*—the combination of plugs, wires, and signals that connect the computer to the

printer. And other brands may not directly plug into the Commodore 64.

The bottom line is that if you don't want to delve into programming, you might want to settle for an official Commodore printer. Your best choice for most routine home and hobby printing chores will probably be the VIC-1525 because it's both cheap and available. And although it's definitely not the best printer in the world, it works and is workable.

Although the mechanics of operating the different Commodore printers differ on some fine points, the software instructions that the Commodore 64 uses for all printers are very much the same. A discussion of how to use the VIC-1525 will help get you started putting your computing into print.

Getting to Printing—Loading Paper

Before attempting to run your new Commodore VIC-1525 printer, be sure to get a supply of paper for it. The C-1526 can use single sheets or continuous-form paper, but probably no one told you that the Commodore VIC-1515 and VIC-1525 printers absolutely must use continuous-form paper with drive sprocket holes on its edges. The print mechanism is entirely unable to handle single sheets of paper such as normal stationery. If you need to use high-quality paper, you can buy special paper with extremely fine perforations.

Loading continuous-form paper into the Commodore-cum-Seikosha printer mechanism is a bit tricky the first time, but is merely bothersome once you've had practice. First remove the smoked plastic cover. Then open the paper drive tractors, the black plastic assemblies at either side of the top of the printer. Put your index finger or fingernail under the front edge of each tractor and lift it up. It should snap into its open position.

Slide the tractors on their mounting rail to line up with the approximate edges of the paper you will be using. The correct position for them is to make the pair centered, each tractor about ½ inch from the end of its travel. Moving them farther left, however, may give you a more pleasing left-hand margin on program listings. Between the tractors are three soft rubber "rollers" that support the paper as it goes through the tractors. Arrange these rollers so that they are equally spaced between the tractors.

Insert the front edge of the paper you plan on using from the rear of the machine. Push it down flat on the angled rear

surface; then, using your fingertips, push it through the slot under the print mechanism. Keep your fingertips feeding the paper into the slot until the front edge pops into view behind the printhead.

Push the paper a bit farther until you have enough paper protruding from the front to grab. Then pull it up about 4 inches and bend it over the tractors. Line up the left edge and press the feed holes in the paper over the drive sprockets in the tractors.

Snap the left tractor closed. Then adjust the right tractor so that the holes in the paper fit neatly over its sprockets and snap the right tractor closed. Make sure that the paper is properly lined up and does not run diagonally through the printer. The paper should extend exactly the same distance and show exactly the same number of open sprocket holes above each tractor. It's easy to be a single sprocket hole off.

Next adjust the tractors so that the paper curls flat around the rubber rollers between them. The idea is to make the tractors far enough apart that there is tension on the paper but not so much tension that the feed holes in the paper are distorted by the feed sprockets.

You could start printing without doing more. Note, however, that you'll be starting halfway down the first sheet—and that first sheet probably isn't in such good condition after being your first experiment in loading the printer.

Using the paper-feed dial on the right edge of the printer, advance the paper so that the perforations between the first sheet and the second one are located just at the top of the printhead—the part that moves back and forth to hit the paper. The paper-advance wheel is much easier to operate if you remove the smoked plastic protective cover from the printer first.

You don't have to be particularly clever to figure out that you've just wasted a sheet of paper. What's more, you'll waste another one every time you type out a single sheet and tear it off the printer. That's the price you have to pay for using continuous-form paper.

As printers go, the Commodore VIC-1515 and VIC-1525 are weaklings. Their miniature mechanisms struggle with the rather ordinary job of running paper through the machine. To ensure against problems, you must make sure that the paper is lined up properly with the printer. Neither printer is able to lift a long string of several sheets of paper from the

floor, so you must place the entire stack of blank supply paper on the same tabletop that the printer is on (or one of the same height).

The paper must also be directly in line with its path through the printer. Stand in front of your loaded printer and move the pile of supply paper so that its edges are even with the edges of the paper emerging from the printer's tractors. If the paper does not exactly line up or has the least resistance in feeding to the printer, it will go askew or bind up after the first several sheets are used.

The Ribbon

Some printers use ribbon cartridges that are easy to load but somewhat expensive because you end up buying a plastic cartridge along with the ribbon that you need. Other printers use spooled ribbons much like those on old-fashioned typewriters. The ribbons are cheap, but they're likely to turn your fingers purple no matter how careful you are in threading them through the machine. The inexpensive Commodore printers opt for a compromise solution: the ribbon moves between two half-cartridges. Depending how you look at them, they're either twice the convenience or twice the bother.

Actually, threading the ribbon on the Commodore printers is relatively easy. First familiarize yourself with the ribbon and half-cartridges that come in the clear plastic bag packed with the printer. Note that the two half-cartridges are not identical but are mirror images of one another.

Remove the ribbon from its package and stretch it between the plastic shells, holding it in front of you. When you do, you'll have one chance in four of having the cartridge/ribbon assembly oriented properly for threading in the printer.

Note that each cartridge half has a tab on one of its sides that is parallel to the ribbon. These tabs should be nearest to you. The top of each half-cartridge has a ridge on it parallel to the ribbon. The bottom of each cartridge has an L-shaped plastic ridge and a small post at its outside edge.

If you have the ribbon properly oriented, when you stretch the two shells apart, you should note that there is a tensioning spring in the *right* half-cartridge. Tug the two shells apart, and you'll feel a little "give" from the tensioning spring.

Once you have the ribbon properly oriented, installing it is easy. First slide the unsupported length of ribbon farthest

away from you between the printhead and the paper. Use your fingers if you must. If you don't want to ruin your manicure, you can use the eraser end of a pencil to slide it down.

Stretch the two half-cartridges apart. Note the two shiny steel tables on either side of the printhead chamber. Place the half-cartridges atop them, still stretching them apart. Wriggle them around until the tabs and posts on the bottom of the half-cartridge shells seat themselves on the edges of the tables. When they are properly seated atop the tables, release them. They should snap into place, held down by the L-shaped tab on the bottom of each half-cartridge and the tension of the spring inside the right shell.

Although the printhead usually stops at the leftmost end of its travel, a location that's rather inconvenient for ribbon loading, *do not* try to move the printhead to make your life easier. Manually moving it (which actually involves forcing it along) will end your printer's life rather abruptly. Quite literally, you'll strip its gears.

Finally, the ribbon must be clamped into the ribbon advancing mechanism. You'll note a metal shield near you on the platform holding the printhead. On the side of this shield farthest from you is a small plastic lever that is supposed to clamp against the ribbon. When the printhead moves forward, this clamp pulls the ribbon along, advancing it. When the printhead returns, the clamp opens, and the ribbon stays stationary. Just push the clamping lever counterclockwise, then slip the ribbon into the gap that opens up between it and the shield.

After you start printing, if the characters printed on the paper appear to be too light or if the ribbon smudges against the paper between characters or between lines, adjust the printhead force by rotating the other lever (on the left of the ribbon advance) on the printhead mechanism. Turning this lever counterclockwise lightens the print; clockwise darkens it.

When you move the lever, you'll notice that it clicks over several small depressions on its mounting plate. To ensure that the printhead force does not change all by itself during printing as the lever is jostled around, make sure that the lever clicks into one of these depressions and "locks" itself in place.

The Smoke Test

You can easily check to see if you've properly loaded both paper and ribbon into the VIC-1525 (or 1515), as well as determine whether your printer is mechanically in top form without even having to connect the machine to your computer. The printer has a built-in function called "self-test" that lets it automatically and continuously type out its complete repertoire of characters.

To run the printer self-test, find the self-test/device select switch. It's a small blackish tab of a slide switch on the back of the printer. When you face the rear of the VIC-1525 printer, the self-test switch is at the far left, next to the jack that connects the printer to the computer system. This switch is labeled (in hard-to-see, raised plastic characters) T54.

The mysterious moniker is easy to explain. Besides handling the self-test function, this switch also controls the device number assigned to your printer—either 5 or 4. If you have an astute perception of the obvious, you can probably figure out that setting the switch to 5 makes the printer's device number 5. Setting it to 4 makes the device number 4. And setting it to T (you guessed it) gives you the self-test.

To run the self-test, first make sure both paper and ribbon are properly loaded into the printer. Running the test without paper won't reveal any results, and it can also damage the printhead! Slide the self-test switch to T (as far left as it will go), then turn the printer's power switch on. The power switch is the one next to the power cord.

The printer should begin to rattle out every character in its memory. And it will keep rattling until you turn its power off, slide the self-test switch to another position, or both.

If you don't get any results when trying to run the self-test, make sure the self-test switch is in the right position and that the printer is getting power. The red LED labeled "power" on the right front of the printer should glow; if it doesn't, make sure the printer is turned on and plugged in. Still no go? Make sure the outlet the printer is plugged into is live by plugging a lamp into it.

If all else fails, check the printer's fuse. The fuse holder is the cylindrical plastic projection just left of the power cord on the back of the printer. Make sure the printer is unplugged from all power, then use a small screwdriver to twist the fuse out counterclockwise. A good fuse has a tiny, continuous

wire inside it. In a bad fuse the wire is broken or nonexistent, and the glass shell of the fuse may be cloudy or blackened inside. The fuse is size 3AG, and it's rated at ½ ampere.

If the printer's "power" LED lights and the self-test switch is in the right position, but you get no self-test, plan a return trip to the dealer who sold you the machine.

If the self-test runs okay but the characters look too dark, smudgy, or too light to read, adjust the printhead force lever.

If the printhead chases back and forth across the paper and rattles as if it wants to work but doesn't leave a trace on the paper, you probably loaded the ribbon improperly. Check it. If you only get half-characters (either the top half or the bottom half), try readjusting the ribbon.

If the printer prints complete trash—not a letter of the alphabet visible among its strange hieroglyphics—a return engagement with your dealer is appropriate.

And if the printer executes the self-test flawlessly and you see the complete Commodore character set boldly emblazoned on paper, your printer has passed with flying colors and is ready to be put to work.

Using the Commodore Printers with Application Software

In most cases, getting a Commodore printer to work with an application software package like a word processor or an electronic spreadsheet is an elementary matter. Just connect the printer to the Commodore's serial output (see Chapter 3) and set the device number on the printer to match the one that the software package expects (usually number 4) using the self-test/device number switch.

If written especially for the Commodore computer/Commodore printer combination, the software should already know all the secret codes to access the printer and make it do its most elaborate tricks.

BASIC Operating Procedure for Commodore Printers

Although what a printer is, what it does, and how it does it are all straightforward, using a printer with the Commodore

64 (and without specialized software) is not. Just as pressing the RUN button will not make anything run on your Commodore, there's no single button to press to bring the printer to life. You have to learn all the commands and control codes that make each printer do its job.

The only problem with operating your Commodore printer using the BASIC language that is built into your computer is that the commands just don't seem to make sense at first. You tell your computer to PRINT, and what does it do? Scribble across the screen. In order for paper-and-ink printing with BASIC to work, you must understand BASIC a little better.

The Commodore 64 is rather fastidious and likes to keep the data and programs sloshing around in its memory organized. Instead of having papers scattered all over its electronic desk, it tries to put everything in a *file*. Any information it wants to send to a peripheral must be put into a file before the Commodore 64 knows what to do with it.

The first job you have, therefore, is to prepare a file folder to put all the information you want to send into the printer. In terms of Commodore BASIC, you want to OPEN a file. First, you assign a number to each file to identify it, the way you would put a label on a file folder.

With the Commodore 64, you can assign any number between 1 and 255 to these files. File numbers 128 and over will force the printer to add an extra blank line between every line you tell it to print. The choice of file numbers is yours. Of course, since the number identifies the file, you'll always have to use the same number whenever you want to identify the file to add something to it or take something out of it.

When you OPEN the file, you must also tell the Commodore 64 where you want to send the information in the file. By doing that, anything you put in the file should go directly where you want it to go. Again, you must use a number to identify the destination of the data, but this time the choice is not yours. This function belongs to your computer system. The destination must be specified by device number. If you want to send something to your printer and its device number is 4, you must identify it as such in the OPEN command.

An OPEN command for a file bound for the printer also demands that you specify whether you want the printer to use uppercase letters and graphic symbols, or uppercase and lowercase letters for everything it prints.

Appending a zero (or nothing at all) to the end of the

OPEN command instructs the printer to use capitals and symbols. The number 7 tells the printer to use uppercase and lowercase letters.

The command to open a file bound for the printer (as device number 4) takes the following form:

```
OPEN [file number],4,[mode]
```

The command

```
OPEN 42,4,7
```

creates a file number 42 that is assigned to the printer (device number 4) in the uppercase and lowercase letter mode.

Once a file is OPENed, you can send data (or whatever) to the printer to be typed out. You can do this two ways. The first uses a command similar to the PRINT that sends characters to the screen. Each individual line to be sent to the printer is prefaced with PRINT#[file number] followed by a comma.

The # sign distinguishes the command from an ordinary PRINT. If you accidentally leave a space between the # and the PRINT, you will get a ?SYNTAX ERROR. Your Commodore 64 recognizes the word PRINT and thinks the command is an ordinary PRINT followed by a symbol it does not understand (the #).

The file identification number must follow so that the information to be PRINTed is sent to the correct file. You will also get a ?SYNTAX ERROR if you forget to put a comma after the file number.

As with the regular PRINT command, any strings or lines of characters that you want to have printed exactly as you type them *must be* enclosed inside quotation marks.

A typical command to send data to the printer would be as follows:

```
OPEN 42,4,7
PRINT#42, "This is a test of this printer."
```

The printer can also be turned into another outlet of your computer's ideas. If you use the CMD command, it will mimic the monitor screen.

After you type CMD[file number] into your Commodore 64, everything that would normally go to your monitor is instead sent to the file you've specified, typically the printer. (Com-

modore says that the CMD instruction makes the printer "listen" to the data channel.) The first thing your printer will type after a CMD is always a Dreaded Ready!

As an example, to have your printer print out a LISTing of the program currently in your computer's memory, you could type in the following series of immediate commands:

```
OPEN 4,4,0
CMD 4
LIST
```

Note that after you use CMD, all PRINT instructions are sent to the printer but *not* to the monitor screen. What *you* type into the computer is still displayed on the monitor but *not* the printer. That's just how the command works.

Now that you know where the gas pedal is, it would be handy to have a brake, wouldn't it? To stop sending everything to the printer and route it back to the monitor, all you need do is send a PRINT# command. At the end of the PRINT# command, the printer assumes that you're done sending it data, and it stops listening. Commodore says the printer is "unlistened." Doesn't that make everything clear?

The net result is that while CMD is in effect, all PRINT statements are routed directly to the printer. But as soon as the printer receives a PRINT# statement, the CMD is canceled, and all PRINTs go back to the monitor screen where they belong. Subsequent PRINT#s, however, will still go to the printer.

Using files to send information to the printer seems like a neat idea. You can set up several files at the same time—say, one in the graphics mode and one in the lowercase mode—to use as your data demands. The data in one file would turn into printed graphics, and the data in the other file would be printed as uppercase and lowercase letters.

You could even OPEN a new file for every statement, except that the Commodore 64 does not allow more than ten files to be open simultaneously. And don't forget that files are used not only for sending data to the printer but to the disk drive as well. If you're not careful, you could soon run out of files.

Obviously, some way of unOPENing files is necessary so that you don't quickly use up your allocation of ten. As you should have guessed, the proper command is CLOSE followed by the file number that you wish to CLOSE. No other specifications are necessary. All the information from the OPEN

statement is automatically canceled by the CLOSE. The file number in the CLOSE statement is merely necessary to tell the Commodore 64 which OPEN statement to refer to and eliminate.

To CLOSE file number 42, just type:

```
CLOSE 42
```

After you are done using any file, it should be CLOSEd. That way you should never have to worry about how many files are still OPEN at any given time.

Special Printing Effects

Instead of having dozens of switches on its front or back panel to control the variety of special printing effects that it can handle, each Commodore printer uses software instructions called command codes to switch modes (see Table 3).

Your computer system stores and sends a command code exactly as it stores and sends a letter or number—as a single byte of 8 data bits. The only difference is that a command code does not print out on the monitor screen or the printer. Out of the 256 possible combination of 8 bits that the Commodore 64 can manipulate, several have been reserved to represent commands instead of letters of the alphabet, numbers, or

TABLE 3: VIC-1525 PRINTER BASIC COMMAND SUMMARY

Start Double-Width Character Mode	CHR$(14)
Start Dot-Addressable Graphic Mode	CHR$(8)
Start (return to) standard characters	CHR$(15)
Switch to uppercase and lowercase	CHR$(17)
Switch to uppercase/graphics	CHR$(145)
Reverse type (1 line only)	CHR$(18)
Cancel reverse type	CHR$(146)
Repeat graphic column	CHR$(26)
Character-width tab of printhead	CHR$(16)
Dot-width tab of printhead	CHR$(27)CHR$(16)
Carriage return	CHR$(13)
Line feed	CHR$(10)

graphic symbols. These command codes can be sent to the printer through a file using the PRINT# command, just as other data are sent to the printer.

The only problem you face in sending them to your printer is that since these control codes are not letters of the alphabet, numbers, or graphic symbols, they do not appear on the Commodore 64 keyboard! Instead, they are identified by their ASCII (8-bit code) value and sent as a command using the CHR$ function in BASIC.

The Commodore VIC-1525 printer understands a dozen different command codes. The following examples should make clear how to use them.

Double-Width Characters

The Commodore VIC-1525 can print bold, headline-like letters and numbers exactly twice as wide as normal characters to emphasize parts of your text. It makes these characters by printing two horizontal dots every time a single dot would occur in the normal matrix. The command code to set your printer to Double-Width Character Mode is CHR$(14).

Because its effects are so obvious, Double-Width Character Mode makes a good introduction to using command codes. To produce double-width characters after you've OPENed a file to the printer (in the example, the file number is 42), you simply send this command to the printer:

```
OPEN 42,4,0
PRINT#42, CHR$(14)
PRINT#42, "THIS LINE IS PRINTED DOUBLE WIDE."
```

Any printable data (including graphic symbols) sent to the printer after the double-width character command will be printed in double-width, until the command is canceled by entering Dot-Addressable Graphic or Standard-Width Character Mode. Remember that only half as many double-width characters will fit on each line.

With the Commodore VIC-1525 printer, the longest line of double-width characters can be no more than 40 characters wide. If you command the printer to type a line more than 40 characters long, it knows that the long line won't fit on a sheet of paper, and the printer will divide the single line into two shorter ones. Alas, the printer's mastery of English is not

very good. It will split the line up without regard to grammar or sense, always starting the second short line with the forty-first character of the longer line, splitting in the middle or at the end of words, however the characters lie.

Standard-Width Characters

To resume printing normal characters after printing double-width characters or using other nonstandard effects, the command to use is CHR$(15). To demonstrate the effect, following the previous example, type this (remember—file number 42 must be OPEN):

```
PRINT#42, CHR$(15)
PRINT#42, "THIS LINE IS BACK TO STANDARD
          WIDTH."
```

Uppercase and Lowercase Characters

When you OPEN the file bound for the printer, you decide whether you want to use the uppercase and graphic symbol font *or* the uppercase and lowercase font, by the last number in the OPEN command line. If you want to switch between modes, you can OPEN two files to the printer, each in a different mode. That strategy may soon lead to a shortage of available files. Remember, the maximum is ten.

Another way to switch between fonts is to use command codes. To switch the printer to uppercase and lowercase letters *no matter what font you specified when opening the printer file,* use the command code CHR$(17). This command will stay in effect until you cancel it with an uppercase and graphics command.

Note that the monitor screen does not change when you use this code. Typing a mixture of letters and symbols on the screen will appear as uppercase and lowercase at the printer. And changing the monitor display (by pressing the Commodore key and SHIFT simultaneously) will not affect the mode of the printer.

Uppercase and Graphics Mode

To have the printer type out in uppercase letters and graphic symbols, *no matter what font you specified when opening the printer file,* the command to use is CHR$(145). This com-

mand will remain in effect until superseded by an Uppercase and Lowercase Mode command and is not affected by the monitor settings.

Reverse Type Mode

Another way to emphasize special characters in your printouts is by "reversing" the characters. That is, instead of being printed as black dots on a background of white paper, the background is made from black dots that let white (unprinted) characters show through. The command to use reverse type is CHR$(18).

The Reverse Type Command affects all letters, numbers, and graphic symbols in either Standard-Width or Double-Width Character Mode. However, the Reverse Type Command remains in effect *only for the duration of the line that contains the command.* The end of a line cancels the command.

Therefore, to print a line in reverse type, the Reverse Type Mode must be on the same line as the characters you want to print in reverse type. Try the following example lines to see how the Reverse Type Mode command works:

```
PRINT#42, "THIS LINE IS NORMAL TYPE"
PRINT#42, CHR$(18) "THIS LINE IS REVERSED
          TYPE"
PRINT#42, "THIS LINE HAS REVERTED TO NORMAL
          TYPE"
```

Canceling Reverse Type Mode

Although the Reverse Type Mode is automatically canceled at the end of each line, sometimes you might want to cancel the command sooner—for instance, to make just one word on a line stand out. To cancel the Reverse Type Mode command before the end of a line, the command is CHR$(146).

Try this example to see how you can use this command:

```
PRINT#42, "REVERSE TYPE "CHR$(18)
          "EMPHASIZES" CHR$(146)"KEY WORDS"
```

Dot-Addressable Graphic Mode

The Commodore VIC-1525 printer (and its little brother VIC-1515) is capable of placing individual dots on a sheet of paper to create charts, graphs, and even rough, newspaper-

like pictures. To take advantage of this feature, you first must command the printer to enter Dot-Addressable Graphic Mode by sending it the command code CHR$(8).

Warning! Dot-addressable graphics is not a playground and definitely not for the meek at heart, the impatient, or the neophyte programmer. You'll be able to while away days trying to put patterns on paper using dot-addressable graphics.

After receiving the Dot-Addressable Graphic Mode command code, the printer becomes an entirely different machine. Instead of putting 6 lines on every vertical inch of paper and leaving a little blank space between each one, it squeezes them together at 9 lines per inch so that there is no blank gap between lines. Instead of printing characters as blocks 6 dots by 7 dots (each character is five dots wide, with one dot skipped between each letter), it prints individual columns of dots 7 high. You must specify which dots of the possible 8 to print for each column. That means for each graphic line you want, you (or the program you write) must specify 480 different dot patterns per line. That's 480 single-dot-wide columns instead of 80-character columns each 6 dots wide.

The dot pattern for each column is determined by a single number. The number for each column pattern represents the sum of the numbers assigned to each dot position to be printed in the column *plus 128*. The top dot is assigned the value 1; the second dot, 2; the third dot, 4; the fourth dot, 8; the fifth dot, 16; the sixth dot, 32; and the bottom dot, 64. In other words, to print the first, third, fifth, and seventh dots in a column, you would add 1 + 4 + 16 + 64 + 128, to get the number that represents the pattern (213).

Once you have all the numbers you need, you have to send them to the printer. Trying to send a seemingly endless stream of numbers in BASIC's Direct or Immediate Mode is merely an exercise in frustration. For instance, merely to print one Commodore symbol takes the following command:

```
PRINT#42,CHR$(8),CHR$(156),CHR$(162),
     CHR$(193),CHR$(193),CHR$(182),CHR$(162)
```

Data can be sent to the printer much more easily in program mode. A cursory look (which probably won't help you much) at how to do it is given in the Commodore printer manual. If you're serious about using the dot-addressable graphics capabilities of your printer and you don't want to devote your life to breaking drawings into individual dots and adding numbers for hours, you should investigate applications software that supports such capabilities. You can also

invest in one of the available graphics utility packages that makes the intricate printer programming much easier.

Repeat Graphic Column

Your life with dot-addressable graphics is made easier by the command code CHR$(26), which causes the dot pattern of a single column to be repeated as many times as you desire. The correct form of the command is:

```
PRINT#42, CHR$(26)+[number of repetitions]+
     [column dot code]
```

A typical command that will print a nearly solid bar 100 dots long would be:

```
PRINT#42, CHR$(26)+CHR$(100)+CHR$(255)
```

Printhead Positioning

To "tab" the printhead to any particular 5-dot (1-character-wide) column, the command to use is CHR$(16) followed by the column position where the next character should print, expressed as a two-digit number. The 2 characters following the CHR$(16) command are interpreted by the printer as the column information no matter whether expressed in CHR$ form or inside a string.

Both of the following commands will start typing at the twentieth 5-dot-wide column. The first line just returns you to Standard Character Mode.

```
PRINT#42, CHR$(15)
PRINT#42, CHR$(16)CHR$(50)CHR$(48)"HELLO
     THERE"
PRINT#42, CHR$(16)"20HELLO THERE"
```

In Dot-Addressable Graphics Mode, the printhead can be positioned to begin at an individual single-dot column by the command CHR$(27)CHR$(16) followed by the number of the column to start printing in, expressed in 2 bytes. If the column number is less than 256, the 2 bytes are CHR$(0) and CHR$(column number). If the column number is greater than 255, the 2 bytes are CHR$(1) and CHR$(column number

minus 256). Because there are 480 dot-columns on a sheet (for the VIC-1525 printer), the maximum value for this tab is 480.

Carriage Return and Line Feed

Individual lines of BASIC need not conform exactly to the lines that the printer prints. The carriage return code, CHR$(13), acts just like the RETURN key on many electric typewriters. It causes the printhead to scuttle back to its leftmost position and advances the paper 1 line. Inserting a CHR$(13) in the middle of a line will split it in two.

For example, try this:

```
PRINT#42, "THIS IS ONE LINE" CHR$(13) "AND
     THIS IS ANOTHER"
```

On the VIC-1525 printer, the line feed command, CHR$(10), has exactly the same effect.

```
PRINT#42, "THIS IS ANOTHER LINE" CHR$(10)
     "AND ANOTHER"
```

Why two commands to reach exactly the same end? On many other printers (but not the VC-1525) the carriage return and line feed are separate commands. A line feed advances the paper 1 line without altering the printhead position, and a carriage return merely sends the printhead back to the leftmost column and sets it to print right on top of the previous line.

Other Printers and Other Problems

If you really want to, you can use printers other than the standard factory-issue models with your Commodore 64. The easiest ones to work are those that are designed to work specifically with the Commodore 64 and have a special "Commodore serial interface." Those key words mean that such a printer will plug into your computer exactly as a Commodore printer would.

Some of the printers from outside suppliers that are designed to work with the Commodore 64 even have built-in knowledge of the special characters and codes that your com-

puter uses. These printers are as easy to connect and use as any standard Commodore product.

The printers offered by outside suppliers often have features the ordinary Commodore printers do not. In most cases, these special features can be activated using CHR$ commands, just as the special functions in the Commodore printers are controlled. The command codes of other manufacturers' printers are probably *not the same* as the code numbers used by Commodore. You will have to check the programer's manual of each printer to find out what the right command codes are.

The printers of outside suppliers often use "escape sequences" for command codes. An escape sequence can be sent to a printer like any other command code—the only differences are that the commands use more than 1 character per code and that an "escape" character, CHR$(27), is nearly always the first character sent. The "escape" character, in effect, tells the printer that the characters following it are commands.

Printers offered by outside suppliers often do not use the special Commodore serial interface. Rather, they are likely to use an RS-232C serial (or asynchronous) interface, or a standard Centronics parallel interface. Neither of these interfaces is directly compatible with the Commodore 64.

Most standard Centronics parallel interface printers *can* be brought to life and used with the Commodore 64 (if you allow for some of the Commodore's nonstandard character codes) by using a parallel interface adapter. The adapter recommended by most computer stores appears to be the "Card? A" Printer Interface (Cardco, Inc., 313 Mathewson Ave., Wichita, KS 67214; (316) 267-6525; $39.95). The adapter merely plugs into the Commodore 64. It comes with the proper cable and connector to plug into the parallel interface of most printers.

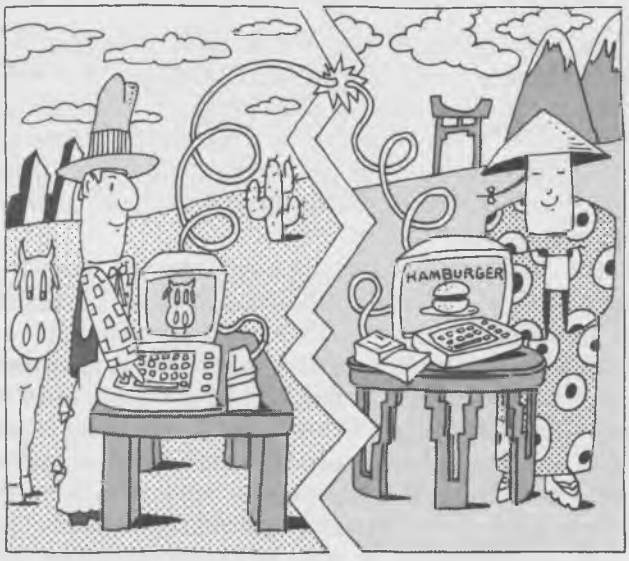
Using a printer that has an RS-232C serial interface with the Commodore 64 is a definite challenge. Although Commodore claims that its Commodore serial interface is the same as an RS-232 except at a different voltage level (5 volts for Commodore, 12 volts RS-232 standard), and an adapter can change the voltage level, using the serial port is an adventure in frustration. In fact, it has been known to take *experienced* programmers *hours* to figure out how to get some standard RS-232 connection to work properly. Besides, using an RS-232 interface may not be generally very rewarding. Since any RS-232 communication with the 64 is limited to

low data rates, about 300 baud to those in the know. That translates into very slow printing.

If you're looking for a good challenge, an RS-232 adapter for the Commodore 64 is available from O.E.M. Inc. (2729 South U.S. #1, Suite 12, Fort Pierce, FL; (305) 464-7549).

9

MAGIC WITH MODEMS AND CONQUERING CP/M



Making the Most of Modems

You have your choice of two factory-issue modems to use with your Commodore 64. The chief difference between them is convenience. The VICModem (Model 1600) is a bare-bones model that does nothing but the essential conversion of Commodore computer signals into telephone signals. The AutoModem (Model 1650) not only does the essential converting but also takes a load off your fingers and will dial the telephone for you or answer it.

Both modems come from Commodore as plug-in cartridges. Both come supplied with programs that change your Commodore 64 into a data terminal that can communicate with nearly any other computer in the world. Software for the VICModem comes on cassette. Although programs for the AutoModem are *supposed* to come on diskette, you may find you get your software in cassette format.

Using either modem is simply a matter of plugging it in (remember, your Commodore 64 *must* be turned off whenever you insert any cartridge!), LOADing the appropriate termi-

nal program, and dialing up the computer or information service you want to communicate with.

Using the VICModem

After you have plugged the VICModem into your Commodore 64, turned everything on, and LOADed the "64 Term" program (using the side of the program cassette marked "Commodore 64"), it's time to check communications parameters and dial your telephone. Remember, the VICModem works only with modular phones—the kind that have a detachable handset or receiver.

Once the VicTerm program is running, the first thing you should do is type F6 on your Commodore 64 keyboard (that means, press SHIFT and F5 simultaneously) to see the "communications parameters." Displayed on your monitor you'll see a menu of all sorts of wonderful parameters that you can change—baud rates, word lengths, stop bits, parity, and stuff like that. You don't have to know what they all mean. Just make sure that what is displayed matches the parameters of the system that you want to communicate with. In most cases, and particularly if you want to talk to the giant CompuServe database, you won't have to change anything. The default settings built into the program are the proper ones for CompuServe (and work fine for talking with many other computers). To change a communications parameter when the menu is on the screen, merely type in the highlighted letter on the line you want to change.

Although the proper command is hard to find on the communications parameter menu, typing T takes you back to "terminal" mode—the condition your Commodore 64 must be in to communicate through the modem. If you type E for "Exit to BASIC," you'll have to reLOAD the terminal program again! If you choose T, you'll know you're in terminal mode by the message TERMINAL READY displayed on your monitor screen.

You should also check to be sure the one control on the VICModem (the switch marked O/A) is in the O position. The O stands for *originate,* which means that you will be dialing the telephone. The A stands for *answer.* You would set the switch to that position only if you were answering a call from another modem. The switch sets the frequencies the modem uses to send out and listen to data.

Dial the telephone as you normally would. After whatever

you want to communicate with (probably a database system computer) answers, you should hear a high-pitched squeal in your telephone receiver. That means there's another modem at the other end waiting to talk to your modem.

Now quickly (you have less than 30 seconds) unplug the handset end of the coiled cord that runs from the base of your telephone to the handset. Lay the handset down but don't hang it up! That will disconnect you!

Plug the now-free end of the wire into the VICModem. If you're not quick enough, the far-off modem will probably hang up on you. If all goes well, the small red LED on the VICModem will light up, and you'll be communicating.

The AutoModem

The first step in communicating with an AutoModem is to plug it in. To use the AutoModem, you must have a phone that uses a standard modular (small, squarish) connector either for the connection between the wire and the wall, for the connection between the wire and the phone, or for both. If you can't disconnect the wire from the wall, you're out of luck until you install a modular jack.

If your telephone has a modular connector that attaches it to the wire from the wall, simply unplug that wire at the telephone and plug the resulting free end into the AutoModem jack marked LINE. Next plug the telephone wire supplied with the AutoModem into the jack on the modem marked PHONE, and plug the other end of this wire into the telephone exactly where you unplugged the first wire.

If you cannot disconnect the wiring between the wall jack and the telephone from the phone itself, disconnect the wire from the wall jack and plug the resulting free end into the AutoModem jack marked PHONE. Then plug one end of the wire supplied with the AutoModem into the modem's LINE jack and the other end of the wire into the wall jack exactly where you unplugged the first wire.

The switch on the AutoModem marked T/D now controls whether the modem or the telephone is connected to your phone line. With this switch in the T (for telephone) position, the telephone works normally and the modem does not. The D (for data) position lets the modem use the line.

The ground rules for the AutoModem are much the same as for the VICModem. Plug it in while your Commodore 64 is turned off. Then switch everything on and LOAD the AutoTerm

20/64 program. Be certain to LOAD the side that says "For use with the C-64." As with the VICModem, the O/A switch controls the originate and answer modem modes.

After the communications program is running, the next step is to set communications parameters. It's done exactly the same way as with the VICModem, except that to change communications parameters you press the F5 key. Typing T takes you back to the terminal program.

When the word DISCONNECTED appears before the legend TERMINAL READY, you're assured that the modem has hung up and/or is not listening to your telephone line. To make a call using the AutoModem, press F6 on your Commodore 64. You'll see this message:

```
MOVE T-D SWITCH TO D
MOVE O-A SWITCH TO O
INPUT PHONE NUMBER
```

Make sure that both switches are in the positions that the screen asks for. Then type the telephone number you wish to dial. When you hit the RETURN key, the modem will dial each number in sequence exactly as you've typed them—up to 30 digits. Do not use hyphens or other punctuation (such as the # and * signs). However, you can use the letter p to indicate a pause in the dialing sequence. This can be helpful in providing time for a second dial tone when using a long-distance service or office telephone that requires you to dial 9 first to get an outside line. If you make a mistake in typing the telephone number you want to dial, the DEL key won't help. You must press the "[up arrow]" key on the right side of the keyboard to erase the whole number and start over again.

The red LED on the modem will light when the AutoModem hears the squeal of another modem. The TERMINAL READY message should reappear on the screen and you should be in communication. If the red LED lights but the TERMINAL READY message does not appear, press the RUN/STOP key on your Commodore 64 to start communicating.

Talking to CompuServe

Probably the first thing you'll do with the modem you buy will be to talk to CompuServe. Not only is the first hour of CompuServe usage included free with the purchase of either Commodore modem, but it also gives you an opportunity to

check out your modem and check out yourself in using it. Remember, all the defaults on the Commodore-supplied communications programs are set at the proper values to chat with CompuServe.

Your first chore is to find the appropriate CompuServe telephone number. A list of toll-free numbers should be included in the packet that comes with your modem. Odds are CompuServe has a local number in your area.

The first thing CompuServe will ask you when you connect up is: Host Name? With the VICModem you might miss these wonderful words because they may be sent while you are in the process of plugging the modem into the telephone.

The proper response is to press the F1 key, which should cause an "[up arrow] C" to appear on your screen, followed by a request for your ID number. If instead CompuServe asks you for Host Name again, you can either type CPS or CIS—or HELP, which will advise you to type CPS or CIS. One of those combinations should elicit the response requesting your ID number. If not, you'll be disconnected after about five tries. When CompuServe asks for your ID number, type in the number on the secret CompuServe envelope that was provided with your modem. As you type each number, it will appear on the screen. Be sure to include any punctuation (usually a comma) included in your ID number. Don't include anything extra—not even a space.

If you get that much right, CompuServe will ask you for your password. Type in, letter for letter—including the punctuation between words—the two-word password hidden inside the secret CompuServe envelope. As you type, the password will *not* appear on the screen. Because your password is not displayed, you can keep your secret password secret from people peering over your shoulder.

If you get that much right, CompuServe will welcome you and ask what kind of terminal you have. If you don't know the right answer and ask for an explanation, you'll probably get the idea that the proper response is "1" for "personal computer with terminal program." It isn't, and CompuServe will tell you so and probably hang up on you. CompuServe classifies your Commodore 64 as a number 4, other.

If you're foolhardy, you'll keep pressing the RETURN key every time CompuServe tells you "Press S or Enter to Continue." In so doing, you'll get explanation upon explanation. You'll probably go through the same menus over and over again until you have them memorized. That's just CompuServe's way of making money. They charge by the minute, and the

more of your time they waste with lengthy explanations, the more you put in their pot.

You may want to look through the menus the first couple of times you log on to CompuServe just to see what's there. Later on, you'll probably discover that the smart response when CompuServe first asks you to "Press S or Enter to Continue" is to type in "GO CBM." That command takes you directly to the Commodore menu. There you can hunt around, join the SIG (Special Interest Group), and take a gander at the various goodies available especially for Commodore owners.

While you're on-line, you can order extra-cost additional documentation about various CompuServe features. It's definitely worth paying a few bucks for the literature, since it's chockful of tips that you can peruse at your leisure without incurring a charge. On a service like CompuServe, time truly is money. You'll save plenty of both if you learn the tricks and plan your sessions ahead.

When you want to *log off* or exit CompuServe (remember, they charge by the minute), a good command to type is BYE. Why don't they tell you that when you start? OFF will also get you out in most cases. When in trouble, hold down the CTRL key and press C. That will usually interrupt what you're doing and let you send a new command to the CompuServe system.

Inter-Commodore Communications

Giant databases aren't the only computers you can talk to with your modem. Nearly any computer with a modem is fair game for your Commodore 64, including other Commodore computers. To carry on a long-distance keyboard-to-keyboard conversation, both ends of the call must have modems and compatible terminal programs.

Your only big concern is that the modem on one end of the call be set in the O mode and the other end be set in the A mode and that all communication parameters be the same at both ends. By setting obscure combinations, you and your conversation mate can give yourselves a primitive form of security system. Such a secret code is quite easy to break, but it may befuddle inadvertent electronic eavesdroppers, if you're worried about such things.

With the VICModem, just dial your telephone (if you're originating the call) and connect the modem (with the O/A switch in the O position) to the handset wire when you want

to start sending computer characters back and forth. You can connect up the modem when the phone starts to ring or wait until after it has been answered and you've gotten your verbal communications out of the way. The person at the other end should have that modem's O/A switch set to the A position and connect the modem to the phone when he hears the squeal from your modem.

The AutoModem will handle all the handset-manipulating chores on its own. Dial with it just as you would to reach CompuServe, as explained above. If the AutoModem on the receiving end has its T/D switch in the D position and its O/A switch in the A position (*and* a communications program is running on the Commodore 64), the modem will automatically answer the telephone and be ready to chatter away.

Other Programs, Other Features

Commodore has slipped a big one past you—and you never noticed. Your new modem and free terminal software let you communicate with almost any other computer in the world, right? The first time you try it, you'll probably notice an amazing fact: as soon as the information on your monitor screen scrolls off, it's gone! There's nothing left! If you want to take another look at something that zoomed past on CompuServe, you've got to go through the same endless stream of menus to find the information all over again. If you listen closely, you can hear the CompuServe cash register ringing.

The free Commodore communications software lacks one major (and necessary) feature: it does not permit you to capture the information your computer receives and SAVE it in a file. If it did, you could *download* or dump into cassette or disk memory all the information that the databases send you. After you disconnected from the database, you could romp through all the data without worry about the register ringing. Another feature the Commodore terminal programs lack is the ability to *upload*—to send disk or tape files from your computer system to someone else's with the modem.

Several terminal programs from other companies have downloading and uploading features built into them. But even many of the more advanced terminal programs suffer from a more subtle omission. Although you can use them to send files and programs to and fro, you cannot RUN programs that you receive. Programs are sent through modems as if they were text rather than programs. They must be converted

back to programs before they can be RUN, but software that can accomplish this conversion is rare.

One program that not only handles the file-to-program conversion but lets your Commodore 64 function just like a major computer database (with password keys and the like) is the Smart 64 Terminal PLUS (Microtechnic Solutions, Inc., P.O. Box 2940, New Haven, CT 06515; (203) 389-8383).

Controlling CP/M

You've probably heard all the claims. "With CP/M you have a library of over (fill in any number between 2,000 and 18 billion) business programs that you can run on your machine!" And, so they say, the Commodore 64's CP/M adapter cartridge will open that massive software library to you.

That's great! It's wonderful! It's terrific! But what *is* CP/M? And are any of the claims actually true?

CP/M stands for Control Program for Microcomputers. It's a Disk Operating System, or DOS. Just like the DOS that comes on the *Test/Demo* disk with the 1541 Disk Drive unit, it determines how files are recorded on disks and how the computer accesses the information in those files.

The chief difference between CP/M and the DOS that comes with the Commodore Disk Drive is that CP/M works with a microprocessor called the Z80 and *only* with that specific microprocessor. (Unfortunately, the Commodore 64 uses a 6510 microprocessor.) Because the two different microprocessors do not understand the same commands, programs written for one microprocessor *will not work* with the other one.

Thousands of programs have been written to rely on the CP/M operating system, primarily because many of the first business microcomputers were built using the Z80 microprocessor. CP/M was the first standard operating system, and it caught on. But none of the programs written for Z80-based computers will RUN on the Commodore 64 by itself.

That's where the Commodore CP/M adapter cartridge comes in. It contains a Z80 microprocessor, which, when plugged into your Commodore 64 and given the proper instructions, takes command of the rest of the computer. Because the cartridge adds the requisite microprocessor to your computer, it promises to make your Commodore 64 capable of running nearly all of those thousands of CP/M programs.

Alas, at least for now, the promise falls a bit short of reality. The existence of programs does not necessarily mean

that they are usable on a Commodore 64 with the added Z80 microprocessor cartridge. The big problem is that it's nearly impossible to get any of the available programs into the Commodore 64's RAM memory so that they can be RUN.

Perhaps the most common way of moving business programs around is on disk. You might assume that once you have a program on a CP/M disk it would be a simple matter to shove the disk into your Commodore 64's disk drive and put its programs to work. However, things in the CP/M world are not quite that simple.

One of CP/M's features is that it's flexible and allows computer designers to customize the electronic format of the disks used in their computer systems. Taking advantage of CP/M's versatility, dozens of different disk formats have been created, nearly one for every different computer. The big problem is that a disk written in one format will not be readable on a machine that uses a different disk format, even though both disks might be CP/M.

The Commodore 64's CP/M disk format is particularly different from any other CP/M disk format because Commodore's engineers faced a nearly insurmountable problem: Commodore 64 CP/M disks have to be compatible with both CP/M and the Commodore DOS. Otherwise, the disk supplied with the CP/M adapter cartridge that holds all of the CP/M and all of its utilities would not be loadable into the Commodore 64.

The Commodore engineers solved the original dual compatibility problem. But they could not solve another, even greater problem—that few, if any, programs are available in disks that use the Commodore 64's special CP/M disk format. Hence, few (if any) CP/M programs are available on disks that will function in the Commodore 64.

There are other ways of getting programs into a computer. One is to use a modem and transmit the programs from one computer to another. In theory, you could dial up a remote CP/M system (a large number of them exist, operated by generous-at-heart computer hobbyists) and transfer programs from the remote system into your computer. But the Commodore presents several difficulties in accomplishing that goal.

When the CP/M cartridge is in use, you cannot use a modem, or so Commodore advises. Of course, you can try to use the Commodore modem when you're not running CP/M and download the programs from a remote system to a Commodore disk. But once you have CP/M running on your Commodore 64, the CP/M system cannot find the programs on the disk

you've created using the Commodore operating system. Other file transferring schemes are similarly frustrated.

The net result is that in the absence of CP/M programs in the Commodore CP/M disk format, the only way to use a CP/M-based program on your Commodore 64 is to type the assembly language instructions of the program into your computer line by line. That isn't easy or fast or fun.

If that's not enough to discourage you, note that the Commodore 64 CP/M disk format allows storing only about 136K of data on each disk (compared to 170K using the Commodore operating system). One further shortcoming: when the Commodore 64 is running CP/M, only 48K of its RAM memory is available for running programs.

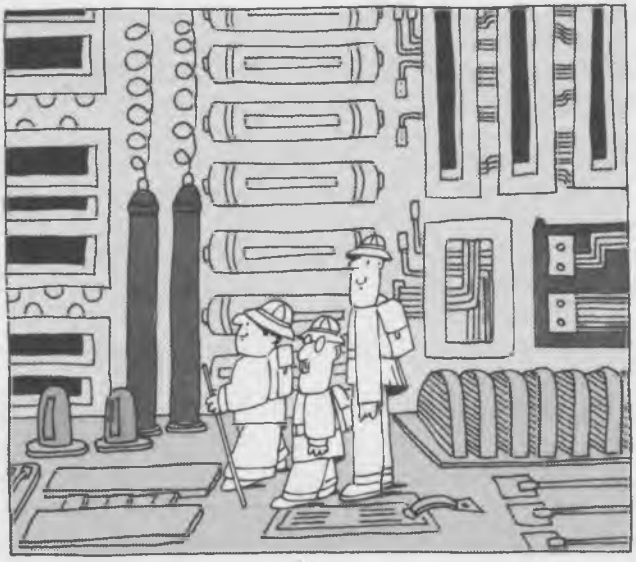
The CP/M supplied for the Commodore 64 is the complete operating system and includes features like the mysteriously named *PIP* (which stands for Peripheral Interchange Program), which copies files and disks; *STAT,* which tells you how much space is left on a disk and protects files from being overwritten; *ED,* a primitive line editor that makes the standard Commodore screen-oriented editor look like heaven; and *ASM,* a Z80 assembler. If you've worked with CP/M before, CP/M on the Commodore 64 will seem like an old friend—familiar, a little crotchety, and kind of slow.

The discouraging words should make you think hard about whether you really need CP/M. Although there's no denying it's a popular operating system, it is by far not at its best when used in conjunction with the Commodore 64. The primary advantage of using CP/M—the availability of a large program library—is constantly decreasing in importance as more and more programs are being written for the Commodore 64 itself.

If you absolutely need CP/M, investigate a non-Commodore add-on CP/M adapter that claims to have solved many of the problems and shortcomings inherent in the Commodore CP/M conversion cartridge. The Estes Engineering, Inc. (825 North Fifth St., Salina, KS 67401; (913) 827-0629) Z80 interface board is designed to reconfigure the Commodore 64 as a 49K CP/M system that is compatible with standard (IBM-formatted) 8-inch floppy diskettes. Most CP/M programs are available in that disk format, and they would therefore be usable (but still slow) on a Commodore 64 system with the Estes interface.

10

THE GRAND TOUR



Twenty years ago the television set was a work of high technology. It was also almost repairable by the average handyman/housewife who had enough dexterity to wield a screwdriver and remove the back from the set. The tubes—bottles full of warm, red electricity—were easy to get at, and all you had to do to repair an ailing set was find the one that had lost its glow and replace it.

For all there is to see inside today's television sets, the backs could be molded shut. No warm, glowing tubes—just impersonal black chips that yield no indication whether they're working or not, save for what you see or don't see on the screen.

The view inside the Commodore 64 is equally, if not less, rewarding. Soldered inside you'll find little more than three dozen black integrated microcircuits that tell no tale about what they do, let alone whether they're doing it.

All that said, now's the time to pry open your Commodore 64 and see what makes it go. What you will see will be both boring and fascinating, the miracle of microelectronics. And likely, though you won't learn how to build one, you will learn *not* to be afraid of the computer machine.

If you carefully follow the instructions below, no one will ever know you've been inside. I've found no disclaimer on the

warranty that opening the machine will forever destroy your chances of having it repaired by the parent company. In fact, at least one software package *requires* that you dig inside to install a chip that will make the program work.

Getting Under the Commodore's Skin

If you appreciate how complicated a computer is, you probably have more than a few reservations about opening your Commodore 64. You're probably afraid that one wrong move and you'll have ten thousand parts flying in every direction as if you've opened a box of hyperthyroid grasshoppers. You're probably worried that even if opening the box goes all right, you'll have dozens of extra parts left over when you screw the lid back down.

Actually, the only reason you should think about opening your computer is that it's so easy and that you're very unlikely to do any harm. The total effort required involves removing three screws. Period. Even if you lose all three, your Commodore will still work. The only tool that you need is a Phillips screwdriver of small-to-medium size.

You must take one precaution. Make sure electricity doesn't even get near your Commodore 64 while you perform your surgery. That means unplugging the black power supply box from the wall and from your Commodore 64. In fact, unplug everything from your Commodore 64. You might want to take greater precautions still and unscrew the fuse going to the outlet that powers your computer.

Note: The descriptions given here are for the current model Commodore 64 at the time of writing and may vary for older production runs and later ones. Primary differences between models are chiefly in interference protection, although more sophisticated revisions deep inside are possible at any time.

Clear off a small tabletop. Turn your Commodore 64 over (so that it is upside-down). Near the front edge (that is, the one nearest the space bar) you should find three Phillips screws at the bottoms of three small depressions. Remove the screws and put them someplace where they won't get lost. Try to remember where you put them.

Turn your Commodore 64 over again so that it is right side up, space bar closest to you. Carefully lift the top front edge. It should rise up easily. When it is about halfway up and you can get your hand inside, note the twisted pair of wires dangling down from the right side of the keyboard. They

attach to a slip-on connector on the bottom half of the computer. Carefully pull the wire half of the connector from its mate on the computer. You'll have to pull it gently upward and to the left. The only function the signals in this wire have is to light the red LED on your Commodore 64 when the power is on.

By now, accidentally or otherwise, the top of the Commodore 64 has probably come free from the bottom, because the rear is held down solely by plastic tabs. Note the large tangle of wires leading down from the top half of the computer to the left side of the bottom half. At the lower end of this group of wires there is a connector like the one on the other pair of wires—but black and much longer. Put your index finger on one end and thumb on the other and carefully pull it straight up. It should come free with little effort. These wires handle all the signals to and from the keyboard.

Your Commodore 64 is now in two pieces—the top half containing the keyboard and the bottom half containing the computer itself.

Before you put the keyboard in a safe place, take note that each set of the interconnecting wires is knotted through a black metal-like ring. Be careful. This ring is actually a readily breakable ceramic compound called ferrite. Far from a decoration, it serves as a radio frequency interference filter. The ferrite in the ring absorbs high-frequency energy that may be in the wires going to the keyboard. These rings help prevent your Commodore 64 from interfering with your family's and friends' television and radio enjoyment.

Probably the first thing you'll note about the bottom half of your Commodore 64 is that it's made from cardboard, or at least looks as if it is. This top layer of the computer, in reality aluminum foil-covered cardboard, is not so much to protect your computer from the world as it is to protect the world from your computer. It is a shield that also helps prevent radio frequency interference from leaking out of your computer. By now you may have the idea that a computer generates a *lot* of potential interference. It does.

Near the rear left of this metal-and-cardboard sheet you should find a piece of copper-colored tape. It's copper colored because it's made from copper. It electrically connects the cardboard cover to the computer itself. Carefully lift the tape off the cardboard—but not from the metal box to which it is soldered rather than stuck. When you are done, you should be able to lift the rear of the cardboard cover and fold (or bend) it forward. *Do not* try to remove this cardboard cover.

Underneath you will see the works of the computer itself.

Hello, Mr. Chips

Presumably you felt at least a tiny twinge of patriotic pride when you first gave your Commodore 64 and its box the once over. Both are emblazoned with the joyous words "Made in USA." Before you go further, glance at the front edge of the now-exposed green printed circuit card, lifting up the cardboard shield. What's this? "Made in Hong Kong"? Indeed, although the cabinet of your computer may have been screwed together on America's shores, the majority of the assembly work was done "offshore," where labor is cheaper. In fact, the whole of the computer *is* the assortment of parts on the green circuit card. The American work was only done on the case.

You'll note that the green circuit board is made from glass-epoxy. That means that it's a top-notch, high-quality material. Give the circuit board a quick inspection, even if you don't know what anything is.

The two most prominent things you should notice after the circuit board are the numerous black integrated circuit chips and the dingy orange, disk-shaped capacitors that lean this way and that.

The disk-shaped capacitors are essentially all the same, and each serves the same purpose. They trap signals that the various integrated circuit chips may inadvertently generate and which may interfere with other circuits connected to the same power supply. In effect, they protect the Commodore 64 from interference among the circuits inside itself.

The black integrated circuit chips do the heavy-duty thinking work of the computer. They interpret your commands, manipulate signals, and even generate all the pictures and sounds the Commodore 64 makes. Each one contains the equivalent of hundreds of thousands of individual transistors or hundreds and thousands of tubes like the ones inside that twenty-year-old television set.

An overview of the circuit board should give you the impression that it's divided into three distinct areas: the black chips on the left, a central walled-in area with a tin box behind it, and a collection of miscellaneous parts on the right. Each of these three areas does have an identifiable purpose. The left side is the computer itself. The middle is the video section that forms the electronic version of your monitor's picture and also turns the computer-style composite video signal into RF (radio-frequency) signals for use with your television set. The right side is mostly power-supply circuitry,

which modifies and conditions the current coming from the black external power supply box.

The one user-replaceable item in the entire computer is the fuse in the power-supply area of the circuit board. If you look closely at it, you should see a thin, squiggly, and continuous wire inside. If you're inside your Commodore 64 on a repair mission and notice that this wire has turned to a gray film in the glass tube of the fuse, you should snap the fuse out of its clip and replace it. The correct value for this fuse is 1.5 amperes. It is a size and type called 3AG. If you replace this fuse and its replacement immediately blows when you turn the power on, do not replace it again, but take your Commodore 64 to the repair shop.

The Cast of Characters

Each of the large integrated circuit chips inside the Commodore 64 is a complete subassembly within itself.

On the far left side of the rear of the circuit board you should see a pair of twins—two identical circuit chips. Commodore calls these Complex Interface Adapters (or CIAs). They adapt the Commodore 64's internal signals to the level and data bit arrangements that can be used by external devices.

The left CIA handles the signals from the keyboard (which to the computer-on-a-card is an external device) and interprets the signals that come *from* the Datassette unit when you use one in your computer system. The other CIA handles the signals going *to* the user port and the serial port. It also contains the time-of-day clock that you access through BASIC as TI$.

As your eyes wander rightward, the next three comparatively stubby integrated circuit chips are the Commodore 64's ROM (read-only memory). The leftmost contains the digital code for the high-level BASIC language you usually use for programming the Commodore. The middle ROM chip contains the digital commands for the computer's kernel, the set of instructions that tell the Commodore 64 exactly how a computer is supposed to behave—what to do when the power is turned on, where to look in memory for particular information, how to handle specific problems. The rightmost of the ROMs is the Commodore 64's character memory. Coded in its memory is every location for every dot of each character displayed on the monitor screen.

In the rather ignoble position of sixth from the left is the heart and brain of the Commodore 64—the microprocessor chip. This chip does all the computing inside the computer. The Commodore 64's specific microprocessor is a type 6510. You can probably see that number silk-screened on top of the chip.

The 6510 is a slightly modified version of the 6502 microprocessor used in a number of other computers, including the Commodore 64's little brother VIC-20 and big brother PET, as well as computers from other manufacturers, including Atari and even Apple. The principal difference between the 6502 and the 6510 is that the newer 6510 has two registers that the 6502 does not have—an I/O port and a data direction register.

One frequently asked question is that if the Commodore 64's microprocessor is so similar to the brains in so many other computers, why can't the software that runs on these other computers—particularly the vast business and educational software library of the Apple—easily be modified to run on the Commodore? Alas, there is no good answer.

The last in the top row of large integrated circuit chips has not been blessed with a Commodore-approved formal name. For that reason alone you're not likely to encounter references to it. But it does handle important internal "housekeeping" functions by keeping an eye on what goes on in the internal circuitry of your Commodore 64.

The integrated circuit chip directly in front of the anonymous one is what Commodore calls the Sound Interface Device, or SID, also known as the 6581. The resident noisemaker, it synthesizes the waveforms of all the sounds that the Commodore 64 makes from your programming instructions in either BASIC or machine code.

The gaggle of smaller integrated circuit chips nearer the front of the computer's printed circuit card are various logic support devices, except for eight identical chips that deserve special mention. These eight chips hold the Commodore's RAM memory, 64K bits per chip, totaling 64K bytes. (Remember, 8 bits per byte.)

Surrounded by a steel wall just right of the center of the circuit card is the famed Video Interface Controller or VIC-II chip that handles every bit of the Commodore 64's monitor screen image. It's penned up almost all by itself because its interference-generating potential is greater than any of the other circuit-board residents. After all,

nothing can interfere with a television signal like another video signal.

Another notable chip resides in the VIC-II corral—the quartz crystal that controls the frequency of the pulses that control the speed at which the Commodore 64 thinks. It's the silver "can" at the far right of the VIC-II cage.

The metal module directly behind the video area is the RF modulator, which combines the output of the VIC-II chip and the SID chip with a television frequency to make a signal compatible with the antenna terminals of a television set. In effect, it's a tiny television transmitter. It's completely enclosed in a metal case to prevent it from broadcasting to (and thereby interfering with) nearby television and radio sets.

You don't have to be sharp-eyed to notice that nearly all of the big integrated circuits inside the Commodore 64 bear the cryptic epithet MOS. What does it mean?

It means that the chip beneath the label was made by MOS Technology, a wholly owned subsidiary of Commodore International Ltd. And that means that Commodore makes most of the integrated circuits it needs to build computers itself. And *that* means that Commodore has a guaranteed steady supply of circuit parts at a reasonable price. And that's why Commodore can make computers cheaper than nearly anyone else, or so it seems.

Picking Up the Pieces—Reassembly

All too often, the reassembly instructions for anything from sewing machines to dirigibles are simplified to a single line, "Just reverse disassembly procedure." About halfway through the reassembly reversal you discover step 12 of disassembly was something akin to "scatter pieces to the four winds," which proves to be rather difficult to reverse. Although putting the Commodore 64 back together is quite a bit easier than reassembling, say, Humpty-Dumpty, these few kind words may make the simple task go even smoother.

First, find all the pieces. You should have five—the top (keyboard), the bottom (computer with circuit card), and three Phillips head screws. In a pinch you can get by with fewer screws, but don't skimp on the other parts.

Fold down the aluminized cardboard cover in the bottom half of the computer. You might bend the "hinged" crease a little extra so that the cardboard stays down. Next, fold over

the copper tape and press it down so that it holds the cardboard cover down. Ensure that it makes good contact with the aluminized top of the cardboard.

Raise the keyboard to vertical on its left edge and place it next to the left edge of the bottom of the computer. Mate the long black connector with its matching header (the black plastic thing with the square, bare gold wires sticking up) on the left side of the circuit board. Note that one pin is missing from the regularly spaced row of pins on the header and that one hole is missing in the connector. Obviously, these two areas must match. If the white wire from the keyboard is closest to the front of the computer, you've got the connector properly oriented. Press it firmly into place.

(Reassembly note: do *not* remove the strip of "packing tape" on the aluminized cardboard covering the green circuit board. It performs the important function of insulating the bottom of the switch operated by the SHIFT LOCK key from the aluminum interference-prevention layer on the cardboard cover. Removing the tape can cause the SHIFT LOCK key to short out—with undesirable on-screen consequences!)

Now lower the right edge of the keyboard so that it is directly above the right edge of the circuit board, by *rotating* the keyboard. Do not raise it up and don't worry about the other wires—yet.

Lift the front edge of the keyboard about a third to half way up and insert the tabs on the rear part of the keyboard into the matching channels on the rear of the bottom section to form a "hinge." Lower the front edge of the keyboard.

Now slide the small white connector onto the three-pin header on the far left of the circuit card. It will fit either of two ways, but its orientation does not matter here. Just make it fit.

Lower the keyboard until its front edge mates with that of the lower half. Do not force it down. Occasionally, the connecting wires or their ferrite loops will prevent tight closing. If they do, lift the keyboard and rearrange them until you can easily mate the top and bottom halves of the computer.

Now turn the Commodore 64 over and screw in the three Phillips screws in any order. If you've lost all the screws, you can just tape the computer together.

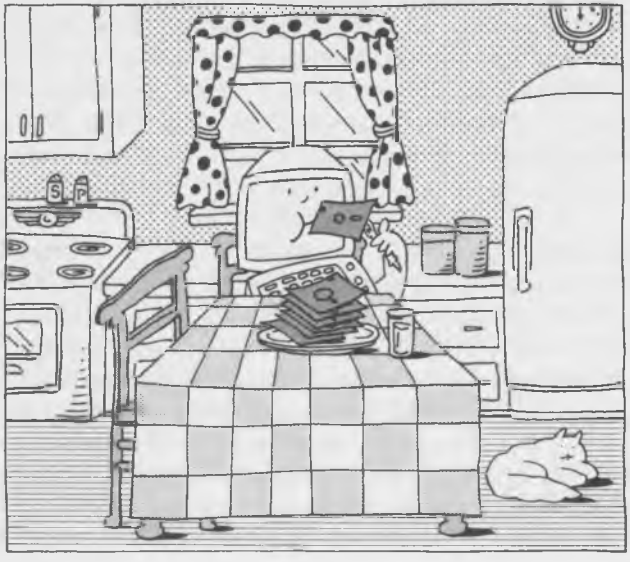
If smoke comes out when you turn your computer on again, you've likely done something wrong.

Lucky for you, however, almost nothing you can do in disassembling or reassembling your Commodore 64 (short of becoming a vandal) will harm its operation. If you've care-

fully followed these instructions step-by-step, no one will ever suspect that you've been peering at the works, and your Commodore will spring to life and continue to give you contented hours of computing.

11

THE CARE AND FEEDING OF THE COMMODORE



The Commodore 64 requires little care and will probably take any punishment that you might give it short of a deliberate hammer stroke. Yet if you have a generous heart and want to do more, you can keep it happy and working and looking like new for a very long time.

There's little doubt that giving a computer the specialized care it requires, and even going beyond what is merely necessary, will prolong the life of your sophisticated and expensive electronic investment. The key words in the common jargon of computerdom are "preventive maintenance."

What should you do?

The minimum you can do is the absolute minimum: nothing. Factory recommendations are minimal because the microelectronics of the Commodore 64 (or any computer, for that matter) require no mechanical care or maintenance. There are no moving parts to wear out.

Rather than slowly grinding down, modern solid-state electronics fail catastrophically. One moment they work, and the next moment nothing is left except your frustration. No matter what you do, there is a chance your Commodore 64—or any computer—might suffer a catastrophic failure.

Fortunately for the space program and any other critical application that relies on microelectronics, such catastrophic failures are rare. After marginal components have been weeded out of any modern electronic device, you shouldn't expect a disaster more often than once in tens of thousands of hours of operation or years of normal use.

The first preventive thing to do is to try to prevent your Commodore from failing *after* its warranty runs out. Rather than trying to extend its life by keeping your use of it to the minimum, push your Commodore 64 to its limits.

The "weeding out" of marginal components in any solid-state device occurs during the first few hours of operation. If your Commodore 64 works after you first turn it on and runs happily along for the first few days, barring an outside cataclysm, your computer should cogitate for years without your lifting a finger except to massage the keyboard.

The logical thing to do when you become the owner of a brand-new Commodore 64 is to pull the fledgling computer from its box, turn it on, and leave it on for the next day or two. You can use the added "burn-in" time for familiarization, learning the new language the Commodore speaks, playing your favorite game, or just allowing the fledging computer to perk along on its own.

Once the computer has proved itself, you can take added precautions that go beyond the factory's recommendations to assure yourself that your Commodore 64 will remain in perfect shape.

A Happy Home

The mandatory care the Commodore 64 requires is, in fact, nothing. But the Commodore 64 is not entirely trouble-free. Although it requires no maintenance, it also requires that you do not abuse it. That means that you must give it a happy home in the type of atmosphere it likes and take care of its particular needs.

In fact, the Commodore 64 does have some moving parts and delicate electrical contacts—in its keyboard.

The keyboard's big enemies are mostly airborne—dust and extreme humidity—but the keys (and the contacts below) can also be tainted and trapped by more worldly problems, from grease to peanut butter and runaway coffee spills.

The Commodore 64's computer circuitry is likely to suffer

ill effects from its environment, too. The climate can be either too hot or too cold or too dry for it, or the machine can suffer from a diet of noxious nasties creeping down the power line, ready to give microcomputers a fatal form of indigestion.

The best advice in trying to give the Commodore 64 a happy home is to be reasonable. You wouldn't put your clock radio in a sandbox or swimming pool. Don't use your computer there, either.

Maybe your thoughts aren't that absurd, but you might be considering a computer room in your dank basement or in your hot attic. Even though the computer doesn't complain about its surroundings, it still is a bit particular about them. Its internal circuitry is designed to work within a wide temperature range, roughly from freezing to nearly 150° F. The so-called commercial operating range for semiconductors is 0° C. to 70° C. Even if the room you keep your computer in doesn't seem too hot to you, remember, its circuits generate heat, too, and the temperatures can add up.

Rather than treating the Commodore 64 as a mindless machine, pretend that it is a sensitive and feeling good friend. Computers are generally happy to operate under any conditions of both temperature and humidity that humans would be happy with. That does not mean that a computer needs the same perfect temperature that you might demand for absolute comfort. Rather, the best preventive measure is to avoid putting any computer in a hostile environment. Don't store your Commodore 64 in the freezer, and don't boil it to thaw it out if you do.

Moreover, you should give your Commodore 64 the same consideration that you would give an expensive machine or to valuable papers and documents. Just as a sugar-laden cup of coffee would be disastrous if spilled on the deed to your home or poured into your stereo receiver, the Commodore 64 would likely succumb to such a drenching.

In other words, the table on which you set up your computer should be reserved for the computer alone. Keep your breakfast of overbrimming Coca-Cola and chocolate-covered doughnuts away unless the equipment is covered by flood insurance.

All liquids should be kept away from the Commodore 64, because its keyboard is relatively unsophisticated and has contacts that can be easily gummed up by sugar pollution. These contacts can be ruined in other ways. In highly humid environments, for instance, contact corrosion will be accelerated.

The keyboard is the most vulnerable part of the Commodore 64 in another way. Although the internal computer circuitry itself will withstand severe shock because it is built on a high-quality glass/epoxy printed circuit board, the keyboard is held together by a cheaper "phenolic" circuit board that is much more easily fractured, say, from a drop from the tabletop. Too much anger and a heavy, hammerlike hand massage of the keyboard can be fatal. Don't pound; just press.

Power-Line Problems

Although the electric line that your Commodore 64 plugs into is its lifeblood, it's also your computer's worst enemy. Not that electricity is bad for the computer. In theory, a solid-state computer won't electrically wear out no matter how long you leave it plugged in. But the current supplied by your favorite electric utility is far from the purity that would grant that theoretically endless life.

Supplied free of charge (and free from recourse) along with the nominal 110–120 volt, 60 Hz (cycles per second) alternating current delivered to your fuse box is a collection of spikes, surges, pulses, glitches, noise, and absences of very brief or prolonged duration. Your computer expects a certain norm, and any deviation is potentially life-threatening to your computer or its data.

Glitches and spikes are real trouble for most computers. Both terms describe essentially the same phenomenon: a temporary pulse of abnormally high voltage on the power line. "Temporary" can mean anywhere from a few millionths of a second to nearly a second long. Abnormally high voltage can range from just a couple volts above what you are supposed to get to 10,000 volts and higher. Because of the brief durations of spikes and power-line glitches, they are invisible. They won't cause the slightest flicker to your lights, and only specialized equipment can determine whether or not they are present on your power line.

Spikes and glitches can be caused by anything from lightning striking near a power line to brief pulses of power added to the electric line when large motors (in all kinds of appliances from refrigerators to air conditioners and industrial equipment) switch on and off. One large spike can be enough to damage the microcircuits in your Commodore 64 permanently.

Some studies have shown that the effects of spikes can be cumulative, like getting small doses of poison over weeks and

months. Each spike can slightly damage the tiny internal structure of the Commodore 64's microcircuits. When enough damage has been done, maybe over a year or more, the computer may mysteriously stop working or just work erratically, refusing to execute simple functions or giving strange answers to obvious questions.

The best way to protect your Commodore 64 from power-line spikes and glitches is with a surge suppressor. Many brands and styles of suppressors are available from a large number of manufacturers.

Although operating principles vary, the most common of these devices use a novel electronic device called a "varistor," which effectively swallows up most spikes and glitches and prevents them from getting to your computer. One suppressor can be used to protect your Commodore 64 and all its peripherals.

Closely related to spikes and glitches are true "surges" and overvoltages, which are power-line voltages higher and longer lasting than normal—with less change in voltage: a dozen or two volts rather than hundreds and thousands. Surges are long enough that you might be able to see the lights in your home flicker or become momentarily brighter.

The low-voltage rise of surges means that most inexpensive spike or surge protectors actually do little good in eliminating them. Only the best and most expensive "ferroresonant" types of surge suppressor or voltage regulator offer genuine protection.

Fortunately, in most cases such protection is genuine overkill. Practical experience indicates that the Commodore 64 is probably impervious to most minor surges, and complete protection is likely to cost more than the computer itself.

Power problems in the other direction, low voltage or none at all, are unlikely to harm your Commodore 64 itself but are likely to wreak havoc with the data and programs in the computer's RAM memory. When voltage dips low enough, long enough (and long enough is not long at all), the power-line condition is equivalent to turning the computer completely off. That means everything in its RAM memory is wiped out, and when power again reaches the more normal level, the turn-on cycle will begin afresh.

The worst damage will be suffered by your patience as you must reload whatever was in RAM, wasting seconds (if you had been using a program safely stored on disk) to hours (if you were developing a lengthy program that you had laboriously entered keystroke by keystroke and not bothered to SAVE).

The anguish caused by voltage drops can be prevented by using a voltage regulator, which automatically adjusts the power supplied to your computer to the optimum level. Your computer's memory can be kept fresh through complete power outages by using an uninterruptible power supply (or UPS), a device that has a built-in set of batteries and the necessary electronic circuitry to convert their output to normal line current so that its output is constant no matter what happens to the electric supply from your power company.

Even the least expensive regulators and uninterruptible supplies are likely to cost several times more than the price of a Commodore 64. Your frustration and the prevention thereof must be worth a great deal if you are to justify the protection such devices afford. The best solution is to SAVE your work early and often and keep backup copies of everything you do.

Static Protection

The other electrical enemy of any computer is static electricity—the blue sparks you feel at your fingertips after shuffling across the carpet and touching a doorknob on a dry winter's day. Static electricity is genuine electricity, as the tingle at your fingertips tells you. The voltage generated by a short shuffle can be tens of thousands of volts.

Nearly everyone knows that there is little current behind this voltage, so static shocks are mostly only annoying and rarely fatal—except to computer circuitry. To delicate microcircuits, a static spark can be more deadly than a spike on the power line. The tiny sparks of static can be enough to blast a silicon microchip to data heaven and run the repair bill into hundreds of dollars.

The only protection from static is prevention. If you don't make sparks, they won't hurt anything. Dozens of products are available to help keep static under control—special carpets, chairs, mats, and sprays. There's little doubt that they work. However, although some form of static control may be necessary in the business computer room where 1 byte of data may mean a million dollars, expensive static protection may be unnecessary in your home.

The Commodore 64 itself does a good job of warding off static sparks. Sparks usually jump from finger to metal—that is, from your body (which can store a great deal of static electricity) to a conductor (which can drain that electricity off

quite quickly). Touching the Commodore 64's insulating plastic case doesn't drain off the static charge fast enough to generate a damaging electrical spike.

The best static preventive is inexpensive and good for you. It's worth doing even if you don't care about your computer's health. Static electricity can build up in your body (or anywhere) only when the relative humidity is low enough that the moisture in the air does not slowly drain off the static charge as fast as it is created. Raising the relative humidity of your home to 30 percent or higher through the use of a humidifier will effectively prevent static shocks to you and your computer. And according to some people, a higher wintertime humidity will help prevent colds and make you feel more comfortable at lower temperatures.

Disk and Cassette Care

As with the Commodore 64 itself, the worst enemies of its magnetic storage media (disks and cassettes) are invisible. Certainly tapes and disks are vulnerable to the same enemies that plague your important documents and even your furniture—the spilled coffee, the dropped cigarettes, the teething pooch. But compared to the records kept in their paper counterparts, magnetic media require extra thought and care in both use and storage.

The disks used by the Commodore 64 need the same special treatment that all computer floppy disks demand. You've probably seen many of the warnings a dozen places already. Keep diskettes away from magnets such as those created by hi-fi speakers, power transformers, electric motors, and the bell in your telephone. The errant magnetic fields from such devices can alter or erase irreplaceable stored data.

Take care always to keep diskettes inside their protective sleeves when they're not in the disk drives. That way dust, airborne contaminants, grease, and grime won't pollute the sensitive magnetic surfaces.

When handling a floppy diskette, guard particularly against touching its magnetic surface where it shows through the slots in its black plastic case. Tiny drops of oil in a fingerprint can be enough to mess up a disk and gum up the works of a disk drive.

Cigarettes pose a particular danger, as they can have lethal effects on computers. Errant ashes have a devious way of collecting on high-precision parts, which include disk drives

and even the diskettes themselves. A flyspeck of ash or even a single particle of tar from a cloud of cigarette smoke can be enough to foul some disk units. In an amazingly short time, the airborne tar from cigarette smoke can slowly coat the mechanical parts and bring your 64 to a halt.

Cassettes and their recorders tend to be more robust than disks and disk drives and are able to withstand much more abuse. For years stereo cassettes have been tossed around, taken to the beach, and left inside cars on incendiary summer days with few ill effects.

Nevertheless, carelessness with cassettes can lead to disaster. Touching the tape surface with your fingers can pollute it with oil and shorten the lifetime of both the cassette and the Datassette drive. Throwing a cassette loaded with important programs and files into a desk drawer with a pair of magnetized scissors is also inviting ruin.

Taking the same care of computer tapes as you would with stereo tapes is both adequate and necessary to keep your programs and files healthy and long lasting.

The Datassette and Disk Drives

Although there is no standard recommended care for disk and cassette drives besides proper protection, their proper operation can be safeguarded by regular care and preventive maintenance.

Both disk drives and the Datassette work exactly like other magnetic recording machines and use "heads" to record and play back (write and read, in computerese) their signals to and from the disks or cassettes. Just as with stereo tape recorders, computer cassette and disk-drive heads can get dirty and magnetized, causing damage to the signals they read and write.

Stereo tape players begin sounding muddier and more muffled as dirt and magnetism accumulate on their heads. Because computer cassettes and disk-drive heads use digital signals, they give no hint that they are encumbered by dirt until they actually misread or miswrite data. Obviously, it's best to clean computer cassette and disk-drive heads regularly to avoid data errors.

Although various sources disagree on how often head cleaning is necessary, once every 10–50 hours of use should be enough to prevent problems and can extend the lifetimes of disks and the drives themselves.

Head-cleaning cassettes and disks make this routine chore simple. Just insert the head-cleaning cassette in the Datassette drive and run it for a few seconds. If you can't get the Datassette to play the cassette, just tell your Commodore 64 to LOAD an imaginary program. When you're done with the head cleaning, press RUN/STOP.

To clean the head of your disk drive, dampen any commercially available head-cleaning disk with special cleaning solvent. Insert the disk into the drive. Then exercise the head by OPENing file number 15 and then instructing your Commodore 64 to LOAD "IMAGINARY FILE", 8. You can use any file name you like. Just be sure to add the device number of the disk drive to the end after a comma.

If you don't want to deal with sloshing solvents around, you'll welcome products like the Datalife system of head-cleaning disks made by Verbatim, in which a pretreated, presoaked cleaning disk is provided in a sealed envelope. The disposable disk is put into a reusable jacket and inserted into the drive slot for a fast, easy, and safe cleanup.

If you're more venturesome, you can go right to the heart of the matter and directly clean the read/write heads of your cassette and disk drives. A cotton swab soaked in a suitable head-cleaning solvent can easily be scrubbed across the heads of the Commodore Datassette to clean them effectively. Just open the lid of the Datassette by pressing EJECT. Pressing the PLAY tab on the Datassette pushes the head outward into a more accessible position. The read/write head on the Datassette is the assembly that moves outward when the PLAY tab is pressed.

Similarly, the heads of a Commodore Disk Drive can be cleaned by first removing the cover from the drive. Make sure that the disk drive is unplugged both from the wall outlet and from the Commodore 64. Obviously, it should be turned off before opening it up! Turn the drive upside down and remove the four screws from the bottom. Turn it back right side up and remove the top cover. Then gently scrub the head (on the bottom side of the arm stretched out into the disk holding area) with lint-free cloth (a photo chamois, available from most photo supply stores, works well) or a video head-cleaning applicator. *Do not* use cotton swabs when cleaning disk-drive heads. Strands of cotton fiber can come off and cause drive problems.

The recommended solvent for cleaning either cassette or disk-drive heads is tape-head cleaner, a special fluorocarbon solvent that can be bought in nearly any stereo store. Alcohol,

either isopropanol (ordinary rubbing alcohol—a 90 percent solution is preferred but 70 percent is okay) or methanol (methyl alcohol or shellac thinner), can be substituted as a tape-head or disk-drive-head-cleaning solvent. Even high-proof vodka will do in a pinch.

As with a stereo cassette recorder, computer cassette drives and disk drives benefit from an occasional head demagnetization. The constant rubbing of the magnetic disk or cassette past the read/write head can cause the head to become magnetized. If enough residual magnetism builds up in the head, the ability of the head to respond to high data rates (at the innermost sectors of disks, for instance) can be impaired. And magnetized heads gradually erase the recording medium.

There is no outward indication of heads' becoming magnetized until unexplainable data errors begin to occur with increasing regularity. Often that warning comes too late, after important files may have been destroyed.

Although the magnitude and extent of the effects of possible head magnetization problems are subject to debate, you, as a concerned Commodore 64 user, may want to regularly demagnetize the read/write heads in your computer system just to be on the safe side.

Any good-quality tape-head demagnetizer designed for use with stereo recorders will probably do a reasonable job on your Commodore 64's heads. For cassette drives you can use a battery-operated cassette demagnetizer (also called a degausser) built into a standard cassette shell. For disk drives you'll need the plug-in variety of demagnetizer that has either a single rodlike extended pole piece or two flat pole pieces that curve together. No matter which style you prefer, make certain that the pole pieces that extend from the main body of the demagnetizer are covered with a soft plastic compound to reduce the chance that you might accidentally scratch your read/write head.

Using any demagnetizer or degausser takes less than a minute and is a good adjunct to your regular head-cleaning routine. That means you'll have to disassemble your disk drive to do it. First turn off and unplug the unit to be demagnetized. Then bring the degausser near (or lightly touching) the head. *Only then* turn it on, slowly removing it from the proximity of the head (to about three feet away), and turn it off. The key to the demagnetizing process is the sudden turn on and slow reduction of the alternating magnetic field of the degausser.

Good Housekeeping

A clean computer is a happy computer. Dirt is the mortal enemy of any computer system. It cakes on print mechanisms and causes them to slow down and bind up and slowly grinds away like sandpaper at delicate disk-drive parts.

If your printer and disk drive are so layered with dust you need to hire an archaeologist just to find them, you'll likely soon be searching out the warranty cards and inspecting the phone book for repair shops.

The best way to keep dust from becoming a problem is by developing a regular cleanup program for your Commodore 64. As often as necessary (judged by running your finger over the equipment to see how much dust it collects), simply vacuum the danger away. Use the finest nozzle available to suck dirt from inside the disk drives, from between the keyboard keys, and from within the printer.

The dust that inevitably collects on your monitor or television screen can be removed with a soft cloth and ordinary window cleaner. Use paper towels and elbow grease as if the screen were just an ordinary piece of glass. Of course, if you feel wealthy and want to lavish extra care on your Commodore 64, you can buy special CRT cleaners, but in the end the results will be about the same.

The same window cleaner is perfect for removing smudges and grime from the Commodore 64 computer/keyboard, disk drive, and printer cases. Take care, however, if you plan on using any spray-type window cleaners. Don't allow any overspray to find its way into *any* openings in the computer (including slots in the monitor case and the disk-drive slot) or between the keys of your keyboard. Rather than spraying to remove smudges from the case or keys, dampen a soft cloth with window cleaner and gently wipe everything off.

For tougher dirt on the outer surfaces of your Commodore 64, disk drive, printer, or monitor, almost any household cleaner may be used. The tough plastic cases are immune to most everyday cleaning solvents.

Be wary of using any strong solvents that frost, glaze, or melt plastics (like acetone or nail polish remover) to remove really stubborn smudges, however. They can ruin the textured finishes of the outer plastic shell of the computer and peripherals and might even melt holes in them. If in doubt, dampen a cotton swab with the anticipated cleaning fluid and try the results on the back or bottom of the cabinets.

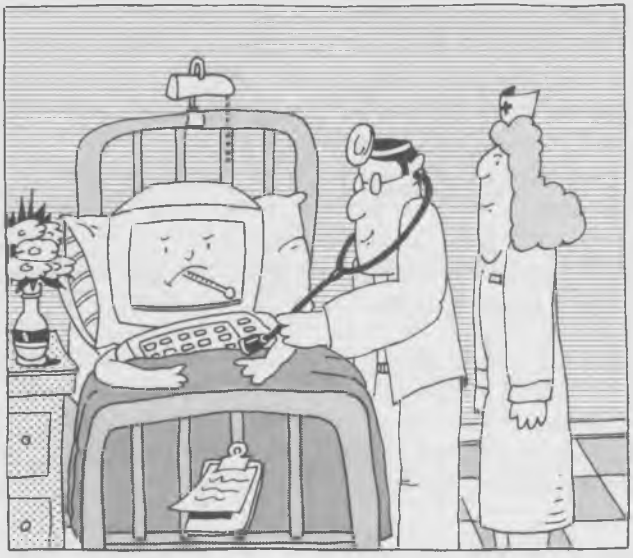
If you accidentally spill something on your keyboard, there's no really good quick fix. The best strategy is to *immediately turn the power off and unplug the computer,* then turn the computer over and let the excess drain out. While it is still upside down, remove the three screws from the bottom and open it up. Get whatever liquid is inside out as fast as possible and dry the machine thoroughly. If you spilled plain tap water on the machine and you reacted fast enough to prevent drips onto the main circuit board, your machine should spring to life after it is properly dried out.

Your Commodore 64 is better off when it is protected from the dirt and dust around it. Not only will protecting a 64 from dirt and grime keep it looking like new, but it will also help prevent potential mechanical problems.

The best protection against dust is a dust cover. Because the Commodore 64 has proven itself to be so very popular, several types of dust covers are available commercially, both from dealers and by mail order. Even the official Commodore brown vinyl cover is a good buy.

12

THAT DOES NOT COMPUTE: TROUBLESHOOTING YOUR 64



Is It Hardware or Software?

Sometimes things just don't go right. No matter what key you press, you get a ?SYNTAX ERROR, or your monitor picture shrinks to a tiny dot in the center of the screen, or you key in LOAD and discover the ten-thousand-line program you finished at 3:00 A.M. yesterday morning is nowhere to be found.

Your first urge may be to get a hammer to gently persuade your Commodore 64 to get back in line with your wishes. Often, however, a cooler head can prevail. Most of the problems that your Commodore 64 suffers are caused by minor slips of the programmer's fingers or misremembered routines, like the correct turn-on sequence. Just retrace your steps, and you probably will find a minor transgression on your part led to a supposed major failure on the part of your computer.

Sometimes your mistakes or omissions can be subtle, illogical, unreasonable, and obscure. Some of the more common and perplexing problems are listed below, along with the usual steps and checks that will lead to prompt repair.

Your first job is always to determine where the problem lies. Is it in the hardware or in the software? You have to know what's wrong to be able to repair it. If you wrote a program that crashes your system, you're likely to be able to fix it. Hardware problems can lead to the repair shop if they're serious. Obviously, then, the first thing to do whenever you run into a problem with your computer is to determine whether it's a "hard" or "soft" error.

Whenever something seems to have gone wrong with your program, your first step should always be to press the RUN/STOP key. Often, you'll be back in control with the welcome "Ready." If that happens, you've probably run into a software problem. Breathe a sigh of relief and prepare to spend time trying to find the error in your program.

Sometimes a program takes control and ignores mere RUN/STOP commands. Sometimes holding down RESTORE while pressing RUN/STOP (for extra emphasis) will clear everything up. Your screen should turn blue and give you a "Ready." If that happens, your problem is still most likely to be in your program. LIST it and go looking for your faux pas.

If all else fails, the radical test is to switch off your Commodore, wait about five seconds, then turn it back on. Your Commodore should show a blue screen, a "Ready," and a clean slate. Any program you had stored in RAM memory only will be wiped out. If you don't get a normal starting message and a Dreaded Ready, you've likely got a hardware problem.

The procedure for finding problems in canned programs is much the same. It is likely, however, that you're not going to be able to fix the program. First try the RUN/STOP button. If you get a Dreaded Ready, RUN the program again. Try RESTORE and RUN/STOP. Try to RUN the program again. If it still doesn't work, switch the computer off and try the program again. If that doesn't work, the program *still* could be at fault. Switch your Commodore off and back on, then LOAD and try another program. If that program works, your computer is probably alive and well, and your software is ailing.

Hardware Problems

If you've never tangled with anything electronic before, trying to repair a computer is probably not the best place to get your initiation. But there are some problems with computer hardware that anyone can fix: the wrong switch thrown, a tape in the wrong place. The only hard part is knowing exactly where to look for the problem.

To help you on your way to computer troubleshooting proficiency, here's a quick run-through of some of the most common hardware-related problems that you're likely to encounter and how to cope with them.

Symptom

a. Television screen filled with a television program rather than a computer image.

b. Power LED on Commodore 64 glowing red.

Solution

a. Wrong channel selected on television set. Make sure your set and the Commodore are tuned to the same channel.

b. Switch box in antenna lead to television set is improperly set. Flick the switch to the "computer" position. Then make sure the proper channel is selected.

Symptom

a. Television screen gray or filled with static.

b. Power LED on Commodore 64 dark.

Solution

a. No power—Commodore switched off. Switch it on!

b. No power—loose plug. Be sure the entire system is properly plugged in. Check power cord from black transformer to wall and from black transformer to Commodore.

c. No power—blown fuse in Commodore. Replace fuse (see Chapter 10).

Symptom

a. Television screen gray or filled with static.

b. Power LED on Commodore 64 glowing red.

Solution

a. Wrong channel selected on television set. Make sure your set and the Commodore are tuned to the same channel.

b. Switch box in antenna lead to television set is improperly set. Flick the switch to the "computer" position. Then make sure the proper channel is selected.

c. No signal getting to television—check cable from Commodore to television set. Make sure all connectors are in solidly. Make sure wires are solidly screwed to terminals both on switch box and television. Try wiggling the connectors. If the picture flickers and you can glimpse part of an image, it is likely that the cable is defective. Replace it.

Symptom

a. Monitor screen black or gray; no image.

b. Power LED on Commodore 64 dark.

Solution

a. No power—Commodore switched off. Switch it on!

b. No power—loose plug. Be sure the entire system is properly plugged in. Check power cord from black transformer to wall and from black transformer to Commodore.

c. No power—blown fuse in Commodore. Replace fuse (see Chapter 10).

Symptom

a. Monitor screen gray or black; no image.

b. Power LED on Commodore 64 glowing red.

Solution

a. Monitor switched off—turn it on!

b. No signal getting to monitor—check cable from Commodore to monitor. Make sure all connectors are in solidly. Try wiggling the connectors. If the picture flickers and you can glimpse part of an image, it is likely that the cable is defective. Replace it.

Symptom

a. Commodore does not respond. Pressing keys does not put characters on screen.

b. Dreaded Ready on screen; cursor not flashing.
c. Screen image "jumps" between two different sizes.

Solution

a. Disk drive connected to Commodore 64 but not turned on. Turn off Commodore, turn on disk drive, then turn Commodore back on.

Symptom

a. All the keys suddenly stop working. Pressing keys does not cause characters to appear on screen.

Solution

a. Software problem—inadvertent POKE switched off keyboard. Try holding down RESTORE and pressing RUN/STOP. If that doesn't work, switch Commodore off, wait five seconds, and switch it back on.

Symptom

a. Canned program running: can't stop program by pressing RUN/STOP or even RUN/STOP and RESTORE.

Solution

a. That's how the program was designed! The canned program can't be stopped because the software company doesn't want you LISTing it or COPYing it.

Symptom

a. Every command, or nearly every one, results in SYNTAX ERROR or DATA OVERFLOW ERROR message.

Solution

a. Software problem—inadvertent POKE put erroneous data somewhere in your Commodore's memory. Try holding down RESTORE and pressing RUN/STOP. If that doesn't work, switch Commodore off, wait 5 seconds, and switch it back on.
b. Hardware problem—RAM memory or other microcircuit has failed. Visit the repair shop.

Symptom

a. Strange symbols appear on screen instead of characters from keys pressed.

Solution

a. Nothing wrong—pressing SHIFT or SHIFT LOCK normally results in graphics characters on screen. To get uppercase and lowercase characters, you must press the SHIFT and COMMODORE keys at the same time. Pressing that key combination again will return you to the strange symbols.

Symptom

a. Lowercase letters instead of graphic symbols on screen.

Solution

a. Wrong keyboard mode—you accidentally pressed the COMMODORE and SHIFT keys at the same time. Press these keys again to return to other mode.

Symptom

a. Cartridge won't work, and cartridge has worked before.

Solution

a. Dirty contacts—the contacts that connect the cartridge to your Commodore 64 can become contaminated and fail to make good contact with the connector on your computer. The contacts are the small metalized areas on the inner edge of the cartridge that goes into the computer cartridge slot. Clean contacts by gently rubbing a pencil eraser over them. The eraser scrubs off the contamination so that good contact is made. Be sure to remove the tiny pieces of left-over eraser before plugging the cartridge back in to your computer.

Symptom

a. New cartridge won't work—strange patterns on screen or nothing at all.

Solution

a. Defective cartridge—sometimes bad cartridges get into the stores. Try running another cartridge or other program. If the other program works, your new cartridge is bad. Take it back.

Symptom

a. Cassette fails to LOAD.

Solution

a. Cassette recorder or Datassette not plugged into Commodore. Turn off the Commodore and make sure the connector is securely plugged in.

b. Tape problem—wrong cassette. You may have accidentally put the wrong cassette into the tape machine. (The results of that error are quite predictable.)

c. Tape problem—partial erasure. If the tape cassette has not been properly stored, it can become partially erased and might never be LOADable again.

d. Dirty heads—if your cassette unit is not properly maintained, the heads can get so dirty that errors result in LOADing or tapes may not LOAD at all. Cleaning the tape heads usually solves that problem (see Chapter 11).

e. Jammed tape—if a cassette jams or fails to properly run through the cassette machine, resulting in improper LOADing or SAVEing, the problem may be caused by a bad cassette (toss it) or by a badly wound cassette. The latter condition usually shows up after a very long tape has been rewound from one end to another. The actual problem is that the tape inside the cassette is unevenly wound and is binding against the cassette shell.

You can often make such cassettes workable by holding them in the palm of your hand and rapping them sharply against a solid object. An appropriate amount of shock causes the tape to fall back into line and frees up the sticky mechanism.

Symptom

a. Cassette won't SAVE.

b. RECORD button won't press down.

Solution

a. No cassette in recorder.

b. Write-protect tab missing from cassette—use self-stick tape to cover the write-protect holes on the cassette.

Symptom

a. Disk drive will not SAVE.

b. Green LED on front of disk drive not lit.

Solution

a. Disk drive not plugged in—plug it in.

b. Disk drive not turned on—turn it on.

Symptom

a. Disk drive will not SAVE.

b. Green LED on disk drive glows; red LED does not.

Solution

a. Disk drive not connected to Commodore—check cable. Make sure both ends are plugged in.

Symptom

a. Disk drive will not SAVE.

b. Green LED on disk drive lit; red LED *flashing*.

Solution

a. Disk in drive not formatted—format disk using NEW command (see Chap. Seven).

b. Write-protect tab on disk—remove write-protect tab, the small piece of tape that covers the write-protect notch on the edge of the disk.

c. Disk improperly inserted in drive—remove disk and insert properly.

Symptom

a. Modem fails to communicate, or strange characters appear on screen when using modem.

Solution

a. Modem not properly plugged in. Check cables—particularly be sure VICModem is plugged into the base of the telephone and not the handset. Turn off Commodore and make sure that modem is properly in its slot in the Commodore.

b. Communications parameters improperly set. Check the parameters in effect in the communications program you are using. To display the parameters using the Commodore 64 *Term* program, press F6. To display the parameters using the Commodore *AutoTerm 20/64* program, press F5.

Various communications systems use various parameter settings. The only important consideration is that both ends of the system use the same settings.

Two of the most common settings are shown below:

TABLE 4: COMMON COMMUNICATIONS SETTINGS

Baud rate	300	300
Word length	8	7
Stop bits	1	1
Parity	None	Even
Duplex	Full	Full

The other common error is to have the O/A switch set in the wrong position. Generally, if you dial the telephone, this switch should be set in the O position. More specifically, when two modems are communicating, the switches on the two modems should be set in the opposite positions; that is, one should be set to O and the other set to A. If these switches are set the same at both ends of a communications channel, neither modem will listen to the other one, and communication won't take place.

Symptom

a. Printer prints symbols instead of letters; or

b. Printer prints lowercase letters instead of graphic symbols.

Solution

a. Wrong print mode set when OPENing file to print.

b. CHR$(145) inadvertently sent—this command changes printer to print capital letters and graphics no matter how print file was OPENed.

c. CHR$(17) inadvertently sent—this command changes printer to print capital and lowercase letters no matter how the print file was OPENed.

Instructions for switching modes and an explanation of them are in Chapter Eight.

Symptom

a. Printer prints too light.

Solution

a. Printhead force too light—rotate printhead level one or more notches clockwise.

b. Ribbon dried out—replace ribbon.

Symptom

a. Printer prints too dark; smudges paper.

Solution

a. Printhead pressure too high—reduce by rotating printhead lever one or more notches counterclockwise.

Symptom

a. Printer prints only top or bottom half of each character.

Solution

a. Ribbon improperly mounted—either riding too high or low. Center ribbon on printhead.

Symptom

a. Printer chews holes in ribbon.

Solution

a. Ribbon threaded improperly—the ribbon will not advance if it is not routed between the paper-advance level and the front shield on the printhead mechanism. See Chapter Eight for proper ribbon-loading instructions.

Software Troubleshooting

BASIC isn't so bad. It is ready, willing, and able to solve your toughest problems, and it's a real help at finding errors you make in programming. If something goes wrong, BASIC doesn't just desert you and let you wonder what happened. Rather, it gives you an error message to tell you what went wrong.

Whenever you get an error message, the first thing to do is LIST the line specified by the error message and see if you can find what you did wrong (or what your Commodore 64 thinks is wrong). The error messages presented on the screen are meant to help you find mistakes or potential problems. Learning what error messages mean and the possible problems that they point to can speed your program debugging and get you into error-free computing faster.

The following is a complete list of all Commodore 64 error messages, what they indicate, and places to look for the problems.

Error Message: BAD DATA

This message occurs when data are being retrieved from an open file by a program and the program expects data of a different type from what it actually receives. Usually, the program is expecting numbers instead of string data (alphabetical characters).

Several problems can generate this message. You may be accessing the wrong file. Or the right file may have different contents from the ones you expected. You could be looking for the right data in the wrong place in the file, or vice versa. You might have your variables, string and numerical, in the wrong order, or you may have forgotten one. Or you just might have forgotten to put the ubiquitous dollar sign ($) after your string-variable name.

Error Message: BAD SUBSCRIPT

This message indicates that your program is trying to find an element of an array that is outside the dimension of the array you set with the DIM command. It can't find the data because the data can't possibly exist, at least to the program's way of thinking.

The obvious error is that you typed in the wrong number to

request the element of the array that you wanted or that you incorrectly DIMensioned the array.

A more subtle error can get you, too. If you refer to an array before it has been DIMensioned, BASIC automatically assigns the array ten elements in each dimension used in that first reference. If you later try to DIMension the array, after that first reference, you'll get a REDIM'D ARRAY error message. However, if your program has the right (or wrong) twists and turns, you could call for an out-of-range element before your DIMension command (which would produce the REDIM'D ARRAY error message) is called by the program.

In other words, when debugging, check not only the dimensions that you've set but where in the program you've set them.

Error Message: BREAK

Actually, this is not an error message. It merely confirms that you've squashed down the RUN/STOP key and halted execution of the program. Typing CONT will start the program from exactly where you broke it off. Typing RUN will start the program over from its lowest-numbered line.

A more elaborate message, BREAK IN LINE XXXX, occurs after the program encounters a STOP command. Don't worry about it. It's only natural. If the message bothers you, substitute END for STOP, and you'll get a Dreaded Ready instead of a BREAK IN LINE XXXX message.

Error Message: CAN'T CONTINUE

This message means that you haven't been paying attention. You've tried to CONTinue running a program without RUNning it in the first place. Or you're trying to CONTinue running a program that stopped because it couldn't run properly. Or you have changed the program by editing a line since you stopped it from RUNning.

The message is nothing to worry about. It just says that the computer can't do what you've asked it to. The program can be restarted *from its beginning* (rather then resuming where it broke off) by typing RUN.

Error Message: DEVICE NOT PRESENT

Someone's stolen the disk drive! More likely, however, your Commodore 64 just doesn't know that the peripheral device

that you've tried to rouse with an OPEN or associated command is attached.

One possibility is that you forgot to turn on the disk drive or printer or whatever. If a peripheral is not switched on, it will not respond to the computer's queries. The computer assumes that the peripheral is not attached. Or the peripheral in question could be attached but not plugged in. Although the Commodore 64 will let you OPEN a file bound for a peripheral it does not know exists, most likely it will not let you send a CMD, PRINT# INPUT#, or GET# command to or from the device.

If you've double-checked and have found that everything is plugged into the computer system properly and is turned on, check your program. In your OPEN command you might have specified a device number that does not exist.

Error Message: DIVISION BY ZERO

In math you were either taught that division by zero is "impossible" or "undefined." No matter which one you choose, your Commodore 64 refuses to deal with it. If you try to divide by zero, you'll get this message.

While you might not have intentionally tried to divide by zero, some variable in a formula may have sneakily worked its way down to that value and then turned up on the right-hand side of a division operation. Your program will run into this problem if you give your denominator a chance to equal zero at any time. If a variable that will appear in a denominator has a chance of zipping down to zero, you should probably insert an IF X = 0 THEN GO *SOMEWHERE ELSE* (where X is the variable on its way down) line to your program before the division occurs.

Error Message: EXTRA IGNORED

You're trying to fit a gallon of beans into a one-quart can. You've put an INPUT statement in your program and rigged it to accept a certain number of responses. Then you have gone and typed in more than it needs. Since BASIC is being a forgiving sort, it has accepted the numbers that will fit and let the rest of your responses drift away to oblivion.

The obvious answer to avoid this error message is to either type fewer responses or to add more variables to your INPUT statement so that it will accommodate all the responses that you type in.

Error Message: FILE NOT FOUND

This message means what it says. You've sent your Commodore looking for a file (or a program) on either your Datassette or disk drive, and it can't find one by that name. For this message to pop up when you are using a cassette, the Datassette must encounter an end-of-tape marker. The file in question may actually be on the cassette; but if it is, the section of the tape it is on may have already passed by the Datassette's heads when you called for it, and the 64 missed it. Therefore, rewind that cassette and try again. If the file is not on the cassette in the Datassette at all, your Commodore 64 is very unlikely to find it.

If you were looking for a file on a disk and got this message, either the file does not exist, or you have the wrong disk in the disk drive. You may have SCRATCHed it, or you may have tried to OPEN the file, and the disk drive gave a flashing LED error message that you missed, so it did not OPEN the file, after all. Try, try again.

Error Message: FILE NOT OPEN

This message occurs when you call a file that has not previously been OPENed with a CLOSE, CMD, GET#, INPUT#, or PRINT# command. No file exists for the command to work on.

The usual mistakes that lead to this message are getting commands in the wrong order (easy to do with GOTOs and GOSUBs), assigning the wrong file number in the OPEN command, calling the wrong file number with a subsequent command, or just plain forgetting that you must OPEN a file first before you can do anything to the file.

Error Message: FILE OPEN

This message will pop up if you try to OPEN the same file more than once without CLOSEing it in between. Either you forgot to CLOSE the file, or one of your OPENs is superfluous.

Error Message: FORMULA TOO COMPLEX

You've given your Commodore 64 more than it can handle at one time. If the message pops up and points to a mathematical formula, it indicates you've stuffed it full of too many pairs of parentheses.

Actually, parentheses are unlikely to be your problem. The

Commodore will handle more than a dozen pairs without a problem. More likely the problem is a string expression that the program is trying to evaluate. If you run into the message when you're dealing with a lengthy string, break the string into two or more smaller strings before operating on it.

Error Message: ILLEGAL DIRECT

You have tried to use a command in the direct or immediate mode of BASIC that will function only in the program mode. In Commodore BASIC the command that will most likely give this error is INPUT.

This error occurs most frequently when you've forgotten to put a line number in front of an INPUT command. Add the right line number, and the error will go away.

Error Message: ILLEGAL QUANTITY

If you try to use a number that the Commodore 64 can't use where you want to put it, you'll get this message. Common errors include integers outside the Commodore's limits (greater than 32767 or less than –32768) and CHR$ values greater than 255.

If you run into this error message, check the values of the numbers that you put on the line specified in the message. You'll likely find that you've used a variable that's crept up in value. For instance, the loop FOR I=1 to 500: PRINT CHR$(I): NEXT is guaranteed to give you this message as soon as it steps above 255. Or the program may just have gotten out of hand because of some overly complicated math.

Error Message: LOAD

This message indicates a malfunction in the process of loading a program from cassette tape. The Commodore 64 double-checks what it finds on the tape, and sends you this message if there is a problem.

When you get this message, rewind the tape and try to LOAD it again. If that doesn't work, try to LOAD another program (it's best if it's on another tape) to verify whether the problem is limited to a single program, a cassette, or the Datassette itself.

If the problem turns out to be with the program on the tape and you don't have another copy of it, you've just learned why you should back up all your important programs and files.

Although the Commodore disk drive checks for errors differently from the Datassette, the message is the same when it finds one. Try again.

Error Message: NEXT WITHOUT FOR

Generally, this error message is elicited when your program tries to work its way through a complicated program that you've created with dozens of nested loops. If it bumps head on into a NEXT command that has not been preceded by a matching FOR, it has a problem. Actually, it's an easy error to make when you put a loop within a loop.

The best way to locate the unmatched pairs and find your problem is to rewrite the program loops, indenting each nesting to keep them all straight. Verify that each loop ends with a NEXT and the same variable with which it begins.

Error Message: NOT INPUT FILE

You'll get this message if you try to send data the wrong way, that is, try to INPUT or GET data from a file that you specified as output only. This error will pop up when you are playing with cassette or disk-drive files.

When accessing a file on a disk, check the last digit of the OPEN command that you used to create the file you want to INPUT or GET data from. Also make sure that it is the proper one for the direction you want your data to go to or from the file (R for read only and W for write only). You can INPUT or GET data only from an R file.

Cassettes will have no problem reading data from the file if you forget to add a secondary address to your OPEN command. However, should you inadvertently make the secondary address 1 or 2, the file will have been OPENed for writing.

Make sure you've OPENed a file that goes the right direction.

Error Message: NOT OUTPUT FILE

This is another message that indicates that you've tried to send data the wrong way. That is, you've tried to PRINT data to a file that you specified as input only. This error will pop up when you are playing with cassette or disk-drive files.

When accessing a file on a disk, check the last digit of the OPEN command that you used to create the file you want to PRINT data to and make sure that it is the proper one for the direction you want your data to go to or from the file (R for

read only and W for write only). You can PRINT data only to a W file.

With cassettes, you must add a secondary address of 1 or 2 to your OPEN command to enable you to write to the file. If you specify no secondary address, your Commodore 64 will assume you meant 0 and make the file read-only.

Make sure you've OPENed a file that goes the right direction.

Error Message: OUT OF DATA

Getting this error message indicates that your program has tried to READ more DATA than you have supplied it. In other words, the READ command is executed more often than there are DATA statements to satisfy it.

Count up the number of READs you have, including the recursive variety lurking inside loops, and be sure that you've given them all enough DATA to be happy. And check your DATA to make sure that you haven't inadvertently left something out. This is a particularly troublesome problem when you have long lists of numbers as DATA. The item most often omitted is a single comma.

Error Message: OUT OF MEMORY

Although 38911 bytes (or however many you see with your Commodore 64's opening message) are an awful lot of memory, all good things must come to an end sometime. This message indicates that indeed you've filled up your computer's entire working memory. A program doesn't have to be really long to fill up the memory. If it has a lot of FOR/NEXT loops or many GOSUBs in effect at one time, the bytes disappear really fast.

If you encounter this message when trying to RUN a program, you may need to restructure it to eliminate loops and simultaneous subroutines. "Crunching" the program (eliminating all unnecessary characters, remarks, and statements) can also go a long way in providing needed memory space.

Error Message: OVERFLOW

The Commodore 64 can handle big problems and work with overwhelmingly large numbers, but there is a limit to its capacity—170,141,884,000,000,000,000,000,000,000,000,000 (or 1.70141884E+38) to be exact. If the result of a calcula-

tion is greater than this number, the Commodore 64 won't be able to handle it and will instead give this error message as its last whimper. You may be able to work around this limitation by defining your problem in terms of quadrillions or quintillions instead of single units.

Error Message: REDIM'D ARRAY

BASIC requires that all arrays have DIMensions set, so if you don't do it, BASIC will. If you reference an array without DIMensioning it, BASIC automatically DIMensions it to ten elements in each dimension used in that reference. But BASIC won't let you DIMension an array more than once. If you made it smaller, what would happen to the leftover elements? If you try to DIMension any array a second time *or* if you try to DIMension an array after BASIC has DIMensioned it, you will get this error message.

The DIMensioning problem can be subtle. If your program has the right (or wrong) GOSUBs and RETURNs, you could accidentally call for an out-of range element before you DIMension its array. That means that when you're debugging, you should check not only the DIMensions that you've set but where in the program you've set them.

Error Message: REDO FROM START

This message pops up when you've written an INPUT command into your program that expects to see numbers and you enter a string of characters instead. It warns of a type mismatch (like the error message of the same name) with one important difference. Instead of calling the program to a complete halt, it ignores your nonconforming data and patiently waits until you type in what it wants. If you persist in typing characters instead of numbers, it will flash back at you endlessly.

Of course, you might want to enter a string variable at the particular INPUT statement in question. If so, you should check the INPUT statement and ensure that you have named the variable that is asked for with the required string identifying dollar sign ($).

Error Message: RETURN WITHOUT GOSUB

As the program was leaping about from line number to line number following your GOTOs, GOSUBs and whatevers, it

happened across a RETURN when no GOSUB was in effect. A RETURN has meaning only when a GOSUB has been issued.

Your only course of action is to trace through the loops and knots in your program and figure out where you meant the program to start on its quest of the unmatched RETURN. Odds are that there's a corresponding unmatched GOSUB somewhere that will merrily lead the program on its way to oblivion once it is encountered.

Error Message: STRING TOO LONG

The gist of this message is that somehow in a fit of creativity you have managed to create a string of characters longer than the Commodore 64 can handle. Once you've located the string, all you need to do is to break it into two or more pieces, keeping the largest at less than the 255-character BASIC maximum.

Error Message: ?SYNTAX ERROR

It's likely that you'll find yourself staring at these words quite often. This is the Commodore 64's most popular (or most frequent) error message. It simply means that the Commodore 64 has no idea at all what you're trying to tell it.

Remember, computers are finicky machines that require instructions to be letter-perfect. This means a misspelled word, a missing comma or parenthesis, or just an extra character will befuddle the computer so much that it won't know what to do.

The only way to get the computer running again is to search out your transgression and fix it. A seemingly simple job, but you'll soon discover how commas and other marks trick your eyes into seeing them when they're not really there!

Error Message: TYPE MISMATCH

This error message alerts you that you have tried to set a numerical variable equal to a string or a string variable equal to a number. Check your work. If your variable ends in a dollar sign (a "$" that indicates that it is a string), it must equal a string that is enclosed in quotation marks. If your variable does not end in a dollar sign, it must equal a number not locked inside quotes.

Error Message: UNDEF'D FUNCTION

This message indicates that you have asked the program to execute a function but you haven't yet told it what that function is. The computer can't guess what you have in mind. A DEF FN command must appear before the use of that function in the program.

Error Message: UNDEF'D STATEMENT

Program lines can call up other program lines, but if a line is called and no one answers, this message will flash onto the screen. It indicates that a GOTO, GOSUB, IF . . . THEN or RUN command has called for another line number in the program but that the line number does not exist.

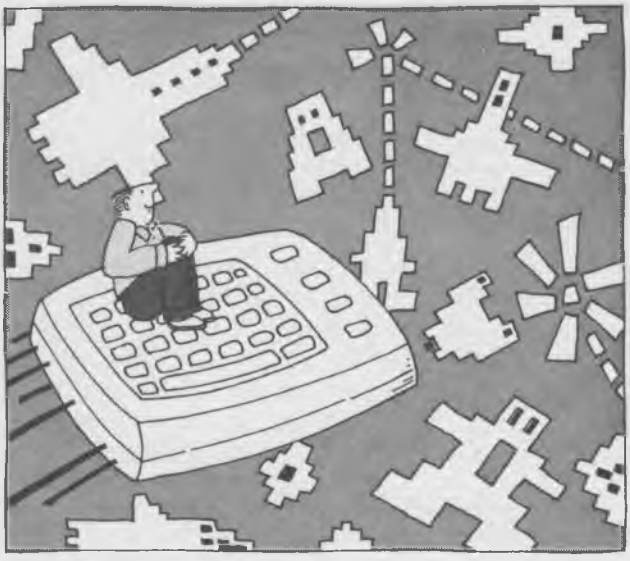
The error might have one of several causes. You may have renumbered lines and not changed the reference in the GOTO or other statements. You may have forgotten the requested line. You may have misnumbered the referenced line (or the reference itself). Or the GOTO (or whatever) could be a left-over and just floating around in your Commodore's memory.

Error Message: VERIFY

You can't do anything about this error. It merely tells you the sad truth that your program does not match the copy of itself on tape or disk. If you used the VERIFY command to make sure your program was properly written to tape or disk, this error message indicates that *your program has not yet been safely stored.* You should try to SAVE the program again, then VERIFY it again to make certain your SAVE was successful. Don't forget to rewind your cassette first.

13

WHAT'S NEXT? ADDING HARDWARE AND SOFTWARE



Sorting Through Software

Computer people, and especially computer writers, have a knack for assuming the rest of the world is familiar with what they regard as fundamentals. They toss about terms so often they don't remember how to communicate without using them and have forgotten how befuddled they were before they learned the secret jargon of computing. Perhaps the most perplexing of terms are those associated with software.

Software is essentially a set of programs to run a computer (or any programmable machine). A program is merely a list of instructions written to be carried out in a certain, fixed order. Every activity can be broken into such an instruction set: using the telephone consists of lifting the handset, spinning the dial or pressing the buttons, waiting for requisite acknowledgment (a "Hello" is always reassuring), and the beginning of communication.

Similarly, the instructions to the computer must be in a specific order, or the results won't be as anticipated. (Try

dialing the phone before lifting the handset!) And they must be step-by-step, because most personal computers can only do one thing at a time.

Other Voices—Computer Languages

The commands within a program (the instructions to the computer) must be in a form the machine can understand. Obviously, then, you cannot tell the computer what to do if you don't know the language that the computer understands.

A wide variety of labels are given to the languages used for programming, including proper names (like BASIC) and generic names (like machine language). The generic names divide languages into different types that work in different ways, and the proper names identify individual languages with different advantages and abilities. All languages revolve around the computer or microprocessor that must be controlled.

The microprocessor inside your Commodore 64 comprehends instructions given to it only as a list of numbers. Each number is a particular action to take or the information the action is to be taken on. The program that the computer understands, then, is nothing but a long sequence of numbers. This numbers-only language is called *machine language.*

Every type of microprocessor (as identified by its number, like Commodore 64's 6510) has a different repertoire of numbers it understands and the functions that each number means. Hence, machine language programs are "machine specific." In general, two microprocessors that are not in the same family (that is, not related by their design—the Commodore 64's 6510 and 6502 microprocessors are in the "6500" family, computers that run CP/M have microprocessors that are in the Z80 family, etc.) will not understand one another's machine-language programs.

Nor will most human beings understand machine language. The long strings of numbers befuddle and perplex more than they communicate. Human beings prefer to think in terms of words rather than digits.

The link between machine language and human language is called *assembly language.* Each number/instruction in machine language is given a "mnemonic" description or name that hints at its function (for instance, LD would stand for LOAD). Although confusing to the uninitiated, the mnemon-

ics are relatively easy for experienced programmers to deal with and understand.

After all the commands are properly arranged in order, a special prewritten program called an *assembler* is used to convert the assembly-language instructions of the human being into the machine language the computer understands. Commodore sells an assembler package for do-it-yourself assembly-language programmers.

A half step above assembly language is a "development system," which does more than just translate mnemonics into numbers. The development system aids in creating assembly-language programs by helping find errors and link program parts together, even letting you watch the microprocessor run the program in slow motion to simplify the complicated and complex challenge of thinking like a machine. One development system with excellent documentation (which makes it easy for beginners to understand assembly language) is *Develop-64,* available from French Silk Smoothware, P.O. Box 207, Cannon Falls, MN 55009.

For those who think like people rather than machines, even mnemonics are confusing and difficult to understand. The rigors of step-by-step assembly-language programming are generally more than most casual programmers want to deal with. The real links between people and computers are "high-level" languages like BASIC. High-level languages are characterized by relatively easy-to-understand names for functions and a structure for commands (a syntax) that is similar to human languages. No matter what the name of the high-level language is (BASIC, COMAL, Logo, Pascal, PILOT, or whatever), it can be classed as one of two types, an *interpreter* or a *compiler.*

An interpreter (or interpreted language) is translated or "interpreted" into machine language step-by-step as the program itself is running. Each command line in an interpreted language can initiate a series of dozens or hundreds of machine-language instructions.

Because each high-level command may take many underlying machine-language instructions to carry it out and because the program interpreting is done at the same time as the high-level language program is actually running, such programs tend to run very slowly in both computer and human terms.

Compiled programs are written very much the same way as interpreted programs. However, before they are run on the computer, they are first processed by a special program (the

compiler itself) that converts the high-level language into machine-language instructions. The program can later be run on the computer in machine language. Compiled programs run much faster than interpreted ones because the complicated and time-consuming translation to the lower-level language is not performed while the program is running.

In some cases, the same language code (the commands written in order by the programmer) may be either interpreted or compiled, depending on the language program running it. The same BASIC code, for instance, can either be RUN directly by an interpreter or processed by a compiler and RUN later, in machine language.

There are literally dozens of different computer languages, each with its own particular strengths and weaknesses. Besides the interpreted BASIC language that comes already installed in the Commodore 64, many other high-level languages are already available for it, and undoubtedly the selection will widen. There is no reason, other than convenience, that you must use BASIC with your Commodore 64.

BASIC

BASIC (which stands for Beginners' All-purpose Symbolic Instruction Code) was developed in the early 1960s by John Kemeny and Thomas Kurtz at Dartmouth College and has become the most common of all microcomputer languages. If you are not already familiar with the BASIC language, you soon will be, if only because it is built into your Commodore 64. BASIC is relatively straightforward and uncomplicated, and its commands have a familiar English-like look to them.

Perhaps the most important reason it is built into your Commodore 64 is that a special version of BASIC was developed to be built into ROM chips. But the Commodore 64's ROM chips cannot hold all of the possibilities that BASIC can offer. The standard-equipment BASIC that comes with your computer can be "extended" and made more powerful by adding additional commands and functions that can be loaded into your computer just as if they made up another software package. If you are at all serious about BASIC programming, you'll want to enhance the 64's rather rudimentary language.

One of the best choices for extending the Commodore's ROM-based BASIC is *Simons' BASIC,* which is available as a cartridge, so no disk drive or cassette recorder is necessary to use it. Named after its sixteen-year-old author, David Simons, the software package includes 114 additional commands that

add a plethora of welcome features to the BASIC language. Included are commands most experienced programmers find to be necessities—an automatic program line numberer, a line renumberer, program listing aids, debugging aids, special commands to make program lines invisible to prying eyes (so you can keep your programming strategy secret), a pair of commands to simplify disk handling (no device numbers or file numbers necessary), graphics and sprite enhancements, music enhancements, and more.

The program is not without its flaws, however. For instance, when most BASICs renumber program lines, line references in GOTO and GOSUB statements are renumbered to match the new line numbers they refer to. Simons's BASIC doesn't renumber GOTOs and GOSUBs, so you have to tromp through the program and do it yourself. Otherwise, your renumbered program will crash very fast. Many of the commands are similar to commands in other BASIC dialects but different enough that switching between another computer and your Commodore could get confusing. Nevertheless, the program is powerful and useful enough that it is being distributed by Commodore itself.

Logo

Perhaps the biggest threat to BASIC's microcomputer dominance is Logo. Widely used in education, it was developed from a language called LISP, which has a more structured syntax than BASIC.

Logo was developed by Seymour Papert and other scientists at the Massachusetts Institute of Technology as a combination of theories based on artificial intelligence research and the work of psychologist Jean Piaget. The name Logo is derived from the Greek word for "thought."

Logo extends LISP's power with easy-to-use "turtle" graphics. Logo graphics are called turtle graphics because the on-screen cursor draws a picture as it moves around, as if it were a turtle with a pen attached to its shell.

Although using Logo may not improve the look of the Commodore 64's extensive built-in graphics, it can make drawing pictures with them much easier and more fun. The turtle is moved about by independently controlling two of its characteristics, position and heading. For instance, instead of POKEing dozens of characters to draw a square, the Logo commands would be as simple as:

```
TO SQUARE
FORWARD 100
LEFT 90
FORWARD 100
LEFT 90
FORWARD 100
LEFT 90
FORWARD 100
END
```

Each FORWARD 100 command causes the Turtle to move 100 positions on the screen; the LEFT 90 commands cause the Turtle to change its heading by rotating to the left by 90°.

Although turtle graphics are Logo's most obvious feature, its most powerful aspect (and biggest heritage from LISP) is its ability to define functions using the simple command TO. For instance, the above TO SQUARE routine must only be defined once. To create further squares only requires calling the newly defined function SQUARE, which elicits all the commands that originally defined the first square. In effect, Logo lets you define your own even higher-level language.

Logo is the language preferred by some for education because of its ease of use. In addition, its function-defining function teaches a structured method of dealing with problems.

Pascal

To listen to academicians, the biggest problem with BASIC is the one single command GOTO. Loops and jumps infest BASIC programs like a nest of snakes, tangling themselves and everything else into what many consider a mess. One computer scientist in Switzerland decided to do something about the plague of GOTOs and wrote a new language in which they were notably absent. His goal was to develop a language to be used solely in the college classroom to teach the students how to write compact, structured computer code. The Swiss scientist, Nicklaus Wirth, named his teaching language Pascal, after Blaise Pascal, the seventeenth-century French philosopher, mathematician, and physicist.

Although Pascal was never meant to be used in the real world for real problems, it soon escaped the classroom and was adapted to genuine problem solving. Recently, it has become available for the Commodore 64.

The chief advantage of Pascal is still that it forces its users to write in a clean, highly structured form. Exactly how any Pascal program works is revealed to an experienced program-

mer by an almost casual glance because there are no mysterious GOTOs.

Pascal is generally reserved for college classrooms. Other languages with similar structures (like COMAL) are preferred for lower-class grades.

PILOT

The PILOT language for the Commodore 64, developed by Larry Kheriaty and George Gerhold, is a simple but powerful high-level language. But not only is PILOT a language; it functions as a language net and can tie together subroutines in assembly language or even disk operating system commands. The Commodore 64 version of PILOT has been designed to take full advantage of the Commodore 64's full graphics abilities (including sprite graphics) using a form of turtle graphics as well as the sonic abilities of the Sound Interface Device.

PILOT's strongest suit is text manipulation: sending a line of characters to the screen takes but a two-character command. Moreover, PILOT facilitates the programming of custom characters for specialized notation in language, mathematics, science, or your own secret codes. Because PILOT can tie together assembly language and disk commands, it has been successfully used to make "user-friendly" programs that simplify otherwise baffling procedures and is an excellent choice for writing programs that are easy for others, particularly children, to use.

COMAL

COMAL has more than a superficial resemblance to Pascal. Its indented listings automatically document the program structures. It includes commands similar to those of Pascal and is nearly as powerful. Most important, however, COMAL is designed to be easy to use. Whereas Pascal is designed to teach programming within a rigorous structure, COMAL teaches structure that will lead to a good understanding of programming. It's also the official language taught in schools in Denmark and Ireland, but it has not caught on in the United States.

As with human tongues, there is no one true computer language. Should you run out and buy another language for your Commodore 64? Probably not, unless you're a programmer experienced in some other language and would rather work in that particular one.

Utilities

A utility program is one that makes routine chores easier. They're usually short, and each utility specializes in one function that helps your computer run better.

Perhaps the most important collection of utilities are those included on the Commodore Disk Drive *Test/Demo* disk. They perform the necessary formatting, erasing, and other file-manipulating functions for your disk-drive system.

Another important utility add-on for your Commodore is the set of programs included in the *Commodore Disk (or Cassette) Bonus Pack*. The program that duplicates disks using one disk drive is worth the price of the *Bonus Pack* alone. The *Bonus Pack* also includes a program to copy disks using two drives (if you can borrow one from a friend) and a program to temporarily change the device number of one of the drives so you don't have to type in a mishmash of CHR$ codes.

Other utility programs of interest on the single *Bonus Pack* disk include a copy of the DOS WEDGE, a "PET Emulator," that allows your Commodore 64 to run BASIC programs written for more expensive PET-series computers; dumping, loading, and monitoring programs chiefly of interest to assembly-language programmers; graphics utilities to help you design your own character sets and sprites; sound programs, including eight sound effects you can LIST and learn from; and several simple games, educational programs, and personal financial programs, which are valuable chiefly to show you the Commodore's capabilities (rather than being really useful in themselves). Most of these same utilities, except for those aimed specifically at disk usage, like the DOS WEDGE and copying programs, are duplicated in Commodore's *Cassette Bonus Pack*.

Utilities vary in value to you depending on your interests. If you plan on drawing on the Commodore 64's graphics abilities, you'll have a much easier time using a graphics package. Instead of POKEing here and POKEing there to put patterns on the screen, graphics utilities add commands to BASIC that make drawing easy, closer to turtle graphics than point-by-point programming. For instance, *Screen Graphics 64* (Abacus Software, P.O. Box 7211, Grand Rapids, MI 49510; (616) 241-5510) adds twenty-four new commands to Commodore 64 BASIC to enhance your ability to draw sprites, high resolution, and multicolored graphics.

Another specialized kind of utility is called *diagnostics*. As the name implies, diagnostics help find or diagnose conditions that ail your computer. Diagnostics can check all bits of a computer's RAM memory and assure that all of the computer's peripherals are functioning properly. Although they do not fix problems, diagnostics can tell where the problems lie and whether your computer is due for a trip to the repair station.

Computer Software Associates offers a diagnostic program package called *64 Doctor* that can check your Commodore's keyboard, disk drive, Datassette, printer, RAM memory, and sound and video abilities. It's available from Micro Software International, Inc., The Silk Mill, 44 Oak St., Newton Upper Falls, MA 02164.

Application Programs

Amazingly (at least for the computer world), an application program is exactly what the name implies—a program written specifically for one application. That distinguishing mark is perhaps the only thing that application programs have in common. They may be written to suit any purpose, from finding the mass of an electron to playing an eat-em-up game. They may be written in any language from machine to mumbo-jumbo. They may be any length, from a single line to a dozen disksful. And they may be any price, from free to thousands of dollars.

Usually, application programs are described by the application that they are to serve. Here are some of the most popular applications and programs to work them.

Word Processors

A word processor does more than just turn your computer into a typewriter. In the worst case, a primitive word processor is like a correctable typewriter that lets you see your mistakes before you commit them to paper. Better programs allow you to manipulate the words without retyping them, rearranging sentences and paragraphs just by pressing a few keys.

Word processors differ chiefly in three aspects: editing abilities, which affect the ease with which you can change your mind and what you type; formatting abilities, which

include the ability to change margins and add headings, footings, and footnotes; and display abilities, which include the ability to display on the monitor screen something resembling what you get on paper.

Generally, the easier a word processor is to learn, the more tedious it will be in the long run. Many programs are "menu driven." You're presented a list of possible functions and choose what to do from the menu display. The menus make the program easy to learn because you don't have to memorize anything. But menus soon become a big bother. They take up valuable screen space that could be devoted to text, and changing and rearranging them takes so much time that many menu-driven programs run ponderously slow.

The number of word processors available for the Commodore 64 is rapidly increasing. Some of the more widely advertised (and obtainable) programs include the following:

Easy Script

Easy Script deserves to head the list of Commodore 64 word processors if only because it comes in an official Commodore box. There's a certain security in knowing that the add-on product you chose has been embraced by the original equipment manufacturer, and *Easy Script* won't disappoint you. It's a competent product. When viewed in the context of the other word-mashing products available for the 64, however, it's neither outstanding nor awful.

Certainly *Easy Script* is likable and well designed. It has built-in simplified disk commands. You can even format disks without leaving the program itself. You can make the screen any color you want any time you want. You can do all the normal word processing functions: write and edit text, move big blocks of words around, copy blocks of text inside a document, even combine documents together. As Commodore 64 word processors go, the largest possible size of a single document is huge, roughly twenty pages of double-spaced text. Other Commodore 64 word processors force you to break a large document into one- or two-page chunks that you have to link together to print, and waste commands entering and exiting to edit.

The current version of *Easy Script* is supplied on disk, so you'll need a disk drive to use it. It's also copy protected, so you cannot duplicate the program disk.

If you've never used another word processor, *Easy Script* may seem like heaven, but if you've tried processing words on

a more powerful machine, you'll find *Easy Script* to be purgatory at best. Perhaps its most obnoxious characteristic (which it shares with many Commodore products) is its ignorance of the simple concept of a word. If you type a line more than 40 characters long, *Easy Script* starts the next line with character number 41, no matter whether it is the beginning or middle of a word. Computers have no problem handling words that are randomly chopped in half. People do.

Easy Script also shares a characteristic with most programs that are designed to be easy for beginners to learn: tediousness after you're no longer a beginner. Whenever you want to command the program to do something, you must switch to command mode. There is a brief pause while the program shifts modes; then the program and the keyboard act differently until you switch out of command mode back to editing. For many users, the switch won't matter; for others, it's only a bother; for the rest, it may be a catastrophe.

Although the *Easy Script* program itself is written in a way that makes it easy to learn, its manual doesn't help matters much. In a short while, you'll realize that the manual was written in England. In a longer while, you may suspect that it was written by aliens in a distant galaxy who are vaguely hostile to earthlings. The best way to understand it (and most program manuals) is to read the whole thing and let it sink in, then give the program a spin, trial-ing and error-ing your way through.

Quick Brown Fox

If products were gauged by their publicity, promotion, and advertising, *Quick Brown Fox* would head the list of Commodore 64 word processors. In real life, however, it may be disappointing.

Perhaps *QBF's* biggest advantage is that the Commodore 64 version is available as a cartridge, so you don't need a disk drive to use it. With the *QBF* cartridge, you can write on your 64 and file away all your letters, stories, poems, or whatever on cassette. *QBF* will let you store your documents on disk, too, so it will grow with your computer system.

Alas, you may quickly tire of *QBF's* limited abilities. Its shortcomings are legion. It displays on your monitor screen in basic black, with white letters, even if you have a color monitor or color television set. Its file and disk-handling abilities are limited. It works line by line and can be powerfully confusing. As you roll through a document you're editing,

lines appear in reverse order, so that the previous line pops up below the current one. The instruction manual, although cleverly written, provides inadequate instruction. And sometimes the cursor has a tendency to disappear so that you never know quite where you are.

You can get used to the strange editing. If you use a monochrome monitor, the white-and-black display will be an asset instead of a bother. And *QBF* is nice enough not to break words in half at the ends of lines. In other words, if your needs are minimal, the *Quick Brown Fox* may match them perfectly. It's available from Quick Brown Fox, Inc., 548 Broadway, Suite 4F, New York, NY 10012.

Mirage Concepts Word Processor

The *Mirage Concepts Word Processor* for the Commodore 64 shows off many of the capabilities—and the limitations—of the 64. The disk-based program (you need a disk drive to use it) is straightforward and easy to use because it is menu oriented. All the commands available to you (from moving the cursor and moving huge blocks of text to formatting disks) are always displayed on the screen. But after you've got the commands memorized, you'll find that one-third of your screen is being wasted.

One exemplary feature of this program is its ability to show an 80-column display without requiring you to buy an add-on hardware module. Although the wide display won't be completely legible on most color monitors (because of limitations in the monitors themselves), *Mirage Concepts* advises the wide display be used only to check how a document will look before it is printed.

Besides lack of legibility, the only real disadvantage of the 80-column display mode is getting into it. Documents must be reformatted to go from the 40-column display to the 80-column display. And reformatting is where the biggest weakness shows—the 64 is so slow that when you switch a letter merely three pages long from 40 to 80 columns, you could probably take a nap or tame a leopard before the display is ready again.

Mirage Concepts allows you to select colors for text display, background, and highlighting before the program begins running. Once you've made your selection, however, you can't change it without starting the program all over again. And you can't see the results of your choice until the program starts running!

The manual states that the program takes up most of the Commodore 64's memory because of its 80-column abilities. What's left is enough to handle only about three pages of single-spaced text. You *can* link together documents in three-page sections that you save onto disk individually. But the program gives no warning when it is about to run out of memory. And when you do, you may lose complete control—and your entire three pages may be unworkable. Moral: save your text in two-page chunks.

Despite the defects (you'll quickly learn to work your way around them), the word processor is one of the better Commodore 64 products overall. It's available from Mirage Concepts Incorporated, 2519 West Shaw, #106, Fresno, CA 93711.

TOTL. TEXT 2.6

Conventional wisdom says an application software product (and particularly a word processor) written in BASIC is undesirable, mostly because BASIC is slow and most such products are skeletal, offering only the rudiments of their genre. Although *TOTL. TEXT 2.6 is* written in BASIC, that so-called limitation can actually be put to great advantage to make the program both useful and worthwhile.

Most application software packages are like sealed black boxes. You just plug them in, and they do their job, but you have no idea how they work. All the machinery is hidden inside the box. There's nothing wrong with that if you just want to get the job done. Sometimes, however, you might get curious about what's going on, or you might want to get into the program to do some tinkering and make the machine work better for your particular purpose. You can't tinker or even peek inside most word processors. Special programming tricks prevent you from looking inside to see what's there.

But you can look inside *TOTL. TEXT 2.6* and, if you want to, make changes and improvements to suit yourself, working in familiar Commodore BASIC. Modifications can be as simple as changing the display colors or switching to 80-column screen width (if you have the right hardware add-ons), to adding your own routines.

Although the copyright laws prohibit you from copying parts of the program and calling them your own, you can make a copy of the program (it's not copy protected), tinker with it to your heart's content, and use the modified version for your own purposes. For budding programmers, *TOTL. TEXT 2.6* can be educational, since you can peer into the

program code and learn how it works. (Of course, sorting through the umpteen hundred lines is a challenge for any programmer, novice or expert.)

Right from the box, *TOTL. TEXT 2.6* comes complete with most of the basic word processing features you'll need: editing, block moves, footnotes, pagination, and so on. Other desirable features are absent, such as the ability to copy blocks of text and the breaking of lines at spaces rather than in the middle of words.

The program's BASIC heritage does present some limitations: single, long, continuous documents are not possible (but you can link as many two-pagers as you like); when documents get long, the program gets slow; and quotation marks are forbidden in text (although another symbol can be substituted to force quote marks to be printed when they're necessary on paper). Because *TOTL. TEXT 2.6* works inside the Commodore's BASIC, you need familiarity with both that language and the Commodore disk-drive system to use it effectively.

It's probably not the word processor to use to write an encyclopedia, but it can be a great help with routine chores. *TOTL. TEXT 2.6* is available from TOTL Software, Inc., 1555 Third Ave., Walnut Creek, CA 94596.

Paperclip

When a software company that writes a word processor recommends another word processor as the "Cadillac for the Commodore," you know the product deserves serious consideration. And that's exactly how I learned about *Paperclip*.

Paperclip may be the most powerful word processor for the Commodore 64. It almost seems to think that it's running on a business computer instead of a $200 home machine. Nearly every imaginable word processing command is at your fingertips.

But the sheer power of the program can be a disadvantage if you're only an occasional user. Although the manual is comprehensive and includes a tutorial, learning *Paperclip* is not elementary. The more commands there are, the more there is to learn, but you may get confused by it all.

The *Paperclip* program disk is not copy protected. In fact, your first project is to make a working copy of the disk. However, *Paperclip's* publishers have protected themselves against program piracy. You can only use *Paperclip* on one machine at a time no matter how many copies you make.

Like a growing number of other Commodore 64 programs, *Paperclip* comes with a "key" (sometimes called a "dongle") that must be plugged into joystick port number one for the program to work. For further piracy protection, *Paperclip* comes with a ROM chip that must be installed *inside* your Commodore 64 to bring the program to life. If you're afraid to open up your machine, you can't get the program running. But you might be able to persuade your dealer to install the chip.

Paperclip is available from Batteries Included, 71 McCaul St., Toronto, Ontario, Canada M5T-2X1.

Scriptimus

Scriptimus is actually a subprogram, part of a package called *Master Key* that combines a home finance package, a file manager, graphics, a game, and a word processor. *Scriptimus* is not as powerful as a stand-alone word processing product, but it is an easy-to-use, menu-driven text editor that gives you full cursor control, insert and delete functions, and the ability to move blocks of copy and justify right margins. The program even lets you work simple mathematical problems from within the word processor. *Master Key* features a key or dongle to restrict its use to a single computer at a time. The package is available from International Tri Micro, Inc., 4122 E. Chapman, Suite 30, Orange, CA 92669.

Word Processing Enhancements

Closely allied with word processors are spelling checkers or "dictionary" programs that will quickly run through whatever you type and point out words you've typed that don't really exist in the English language—or at least in the computer's dictionary. "Mailmerge" programs give word processors the ability to combine lists of names and addresses with form letters.

Although spelling-checking programs can be useful, the Commodore 64's less-than-lightninglike speed makes using such software ponderous and time-consuming. To find and flag errors, the computer must read through an entire document and check each different word against its huge dictionary of correctly spelled words. Certainly your Commodore 64 can do the reading and comparing faster than you can, but no spelling program can give you instant answers. Nor are

spelling checkers flawless or completely reliable. Although such programs will find letter combinations that are not real words, like "etaoin" and "shrdlu," they'll blithely assume that "accept" is proper when the right word is "except." Spelling checkers only verify the existence of words and do not check for correct usage in context.

One of the first spelling checkers for the Commodore 64 is *TOTL.SPELLER 3.6*. Even though the program is compiled and has a slimmed-down 10,000-word dictionary (some spelling programs for other computers include 40,000–80,000 words!), *TOTL.SPELLER 3.6* is no speed demon. It will, however, verify spellings in any file created by the TOTL software series. Order it from the same address as *TOTL.TEXT 2.6,* above.

Spreadsheets

A spreadsheet is the electronic equivalent of an accountant's ledger book expanded on a grand scale. The spreadsheet provides spaces, like pigeonholes, to fill with rows and columns of figures, labels, and—most importantly—formulas. The contents of each pigeonhole can be made dependent on one or more others, so that when one changes, the whole document will reflect the effects of the original change.

Because of the great analytic power unleashed by making one entry dependent on one or more others, spreadsheets are chiefly used in financial modeling to answer "What if?" questions. It's easy to chart the differences in conclusions following a change in an underlying assumption.

Electronic spreadsheets vary in size (the number of rows and columns of figures that they can handle) and in versatility (the way the entries can be altered and manipulated). For instance, some spreadsheets allow the length of individual entries to be varied, while others unalterably fix the number of digits that can be slid into any pigeonhole.

Although you might suppose the 64 incapable of wielding such a program because effective spreadsheets require a computer to work with a large number of mathematical calculations very quickly, the Commodore is up to the task. Remember, the 64 is based on a microprocessor nearly identical to the brain in the Apple, and the combination of the Apple computer and VisiCalc spreadsheet is partly credited with starting the microcomputer revolution. In fact, you're likely to find

many more spreadsheets for the Commodore 64 very soon. The first was probably *EasyCalc*, available from Commodore.

Practicalc, from Computer Software Associates, is an alternative spreadsheet currently available for the Commodore 64. It allows you to work with up to 250 rows or 100 columns of figures but not with the full complement in both dimensions simultaneously. Although the monitor presents only a 40-character-wide by 25-character-deep window of the data in the spreadsheet, you can quickly scroll *Practicalc* the full length and width of the work space by simply pressing the cursor control buttons on the Commodore's keyboard. Individual column widths can be varied from 3 to 38 characters.

Practicalc handles normal mathematical, trigonometric, and statistical functions, can sort entries, and can even seek any particular entry on a spreadsheet. Every time you change the numbers or formulas in the work space, to see the effects on the rest of the spreadsheet, you must hit a key to recalculate the values. Nevertheless, *Practicalc* can give you the predicting power you might expect only from a multi-thousand-dollar personal computer. The program is distributed by Micro Software International, Inc., The Silk Mill, 44 Oak St., Newton Upper Falls, MA 02164.

Databases

A database is an electronic filing and sorting system. Information is written onto the electronic equivalent of index cards and then tucked safely away in disk or cassette files. As necessary, the computer can sort through all of the cards, arranging them in a particular order if necessary (for instance, by zip code, last name, or birth date) or separating out certain cards (for instance, finding all the blondes).

The big advantage to an electronic database is that it helps you keep all your files straight. In your home, it can help eliminate piles of paper by putting all the information you need in one place. Although the computer won't turn you from a slob to a compulsively organized neatness freak, it can help you straighten out your affairs if you are so inclined. The database can also help you sort things out and classify them. For instance, an address list in a database can be sorted by zip code or asked to display all its phone numbers with a given area code.

Databases differ chiefly in versatility and their report-generating capacity—what information they can display and

how. Some of the database programs currently available for the Commodore 64 include the following:

File Organizer

This totally menu-driven disk-based database program would be easy to use if it weren't for one defect. When you follow the instructions printed on the box or in the scanty mimeographed pamphlet-cum-manual, it may not work. From reading the disk directory, you could discover that two versions of the program are present on the disk, and LOADing FILE ORGANIZER *B* will actually get the program into memory and running.

Its capacity is not great (roughly two hundred files of four lines of thirty characters each), and its power is not awesome. But you can create files (typically as an electronic address book), store them on disk or cassette, retrieve them, and update, organize, and print them. If your filing needs are modest and rudimentary, *File Organizer* should be able to handle them.

The program, by Cardinal Software, is distributed by Virginia Micro Systems, Inc., 13646 Jefferson Davis Highway, Woodbridge, VA 22191.

Mini Jini

Mini Jini may be unique. It's perhaps the only database program available on a cartridge for the Commodore 64. Even though the program is on cartridge, it will save your files to cassette or disk, provided you have the proper drive. Getting the program running is simple. Basically, you push the cartridge in, turn your Commodore on, and you're ready to go to work.

Mini Jini is a kid brother of a program called *Jinsam* that runs on Commodore PETs. The translation to cartridge is not complete. Strange remnants of the program's past life remain. For instance, on the menu is a command to "exit" (presumably back to your Commodore's BASIC). But "exiting" *Mini Jini* won't take you anywhere, because the *Jini* cartridge completely takes over the Commodore's brain. The program misses its claims in other areas, too. Although it supposedly offers you control over screen colors, it seems the only way to change character colors is by pressing the CNTL and number keys.

The program is what it purports to be, however—a quick

and simple filing system. *Mini Jini* is workable, but don't expect a super-high-powered program. *Mini Jini* is manufactured by Jini Micro-Systems, Inc., Box 274 Kingsbridge Station, Riverdale, NY 10463.

Practifile

Practifile is a slick, menu-driven filing system. It seems to have been written to maximize the amount of memory in your Commodore available for working on files, because only a small part of the program loads at a time. The rest awaits your command, secure in its disk-based home.

Although it should be a good thing to keep as much memory free as possible, the way *Practifile* does the memory saving makes the program very tedious and bothersome to use. With a single disk drive, you can be forever inserting, exchanging, and replacing the program disk and data disks. Every time you make a menu selection, you have to wait while the program hunts for a particular part of itself on the program disk, loads itself, and decides to run.

The program does have good features. The menu makes it easy to use, and it tells you how much memory your files will use both inside your Commodore and when stored on disk *before* you do the storing. It's written in BASIC, so you can LIST it and see how it works.

It's distributed by Micro Software International, which also handles *Practicalc* (address above).

Research Assistant 2.0

Research Assistant 2.0 is a specialized file manager designed to keep track of reference information from books and magazines. It can build cross-reference lists and help find references by keywords. The program has you describe the source of information by author, title, and bibliography and lets you enter notes about the page number, keywords, and dates, as well as four screens filled with comments. The information can be saved on either disk or tape.

Research Assistant is written in BASIC, and if you can sort through its nearly one thousand lines, you can modify it to do whatever you want. By itself, and unmodified, it suffers a rigid but workable structure. The program does catch errors reasonably well. For instance, if you try to SAVE to a write-protected disk, it advises you of your error rather than simply crashing and letting you erroneously think your SAVE was successful.

The program is not as versatile as an omnivorous file handler but does suffice for its limited application. It's distributed by TOTL Software (address listed under *TOTL.TEXT 2.6)*. The same company also offers *TOTL.LABEL 2.6,* a similarly single-purpose product aimed at mailing-list making, organizing, and printing.

Your Filing Cabinet

As another selection of *Master Key,* a combined word processor, accounting, graphics, and file-handling program package, *Your Filing Cabinet* stores up to two hundred records in each database you create using it, with up to fifteen fields in each record. The program allows a limited amount of sorting and will make printouts of your address lists or whatever when you need them. (For distributor, see *Scriptimus,* above.)

Mirage Concepts Database Manager

Created by the same folks who produced the *Mirage Concepts Word Processor,* the *Database Manager* bears a strong family resemblance. It's menu driven and therefore easy to use, almost simple. It's powerful and can handle advanced sorting, calculations, and conditionals as well as the normal file-handling functions. The criticisms of the sibling product apply to this one as well.

Accounting—Home and Business Management

Computerized accounting makes the whole process easier. Instead of painstakingly scribbling rows of figures and looking up all the proper account numbers, accounting software can ask simple questions to which you give simple answers. Then your computer arranges everything and runs a balance just to make sure that you and reality agree.

Although an accounting system won't make money magically appear, such a system does help keep track of where it all goes. The only real advantage that a high-powered business microcomputer has over the Commodore 64 is speed. More expensive computers usually manipulate numbers faster. But the time-consuming part of any accounting package is entering the figures. For most people the speed of the actual

calculations doesn't matter as long as the results are right. With the growing availability of accounting software, the Commodore 64 becomes a better and better choice as a financial tool.

Home Finance Manager, prepared by the Center for Advanced Sciences and Computers, P.O. Box 593, Vienna, VA 22180, shows the Commodore's financial assets and liabilities. Written in BASIC, the program lets you set up a home bookkeeping system much the way a business system works, including a multitude of special codes to classify your expenditures. But the program runs so slowly that you'd swear your computer was mired in digital molasses. After you type in a word, it can be several seconds before your letters fill the blanks on the screen!

Games

Games are application programs, too. Their application is having fun and chasing your troubles (and spare time) away. Currently, games are divided into two major classes: arcade-style games, where graphics flash on the screen and things happen and explode; and adventure games, which match your wits and guessing ability with those of a computer and programmer.

In adventure games you're given scanty evidence and must achieve a goal, such as solving a crime or finding your way out of buried caverns. The computer display may very well be nothing but text, as the computer explains the situation and asks your response. Once you type in your action in a particular situation, it places you in new situations and asks for new responses until you solve the case, find your way out of the mine, die trying, or give up.

While arcade games challenge your dexterity, reflexes, and sometimes strategy-devising abilities, adventure games challenge your wits and imagination. Either can be fun or frustrating. It all depends on how well the game was designed and on your frame of mind.

Below is a sampling of games, both good and bad, that will while away the hours on your Commodore 64. Because so many games are available for the Commodore 64, the following are the ones that game companies recommended as their best. Who could be a better judge?

Oil's Well

Although it's another eat-the-dots game, *Oil's Well* goes well

beyond classic *Pac-Man*. Rather than a roaming, munching man, the mandibles do their work on the tip of a growing, drilling pipeline. Your joystick points the drill along its way; the "fire" button causes it to retract away from danger. Innumerable bombs and booby traps attack along the way. The graphics are good; the sound very well integrated. The game has proven to be a personal favorite. Disk from Sierra On-Line.

David's Midnight Magic

Turn your Commodore into a pinball machine! This program does an amazing translation—the little ball bounces, rolls, and skyrockets away from bumpers and paddles. There's even a "tilt" control for those who put their bodies into play. Alas, the electronic beeps and the minor flickers on the screen still don't convey the sheer excitement of *real* pinball, but compared to the price of a full-fledged pinball game, *David's Midnight Magic* is a winner. Disk from Broderbund.

Choplifter!

Your joystick becomes the control stick of a helicopter, and you rescue tiny hostages from the fire of crablike tanks while dodging attack airplanes and drone missiles. The tiny, tiny men waving for helicopter rescue are amusing; the tanks predictable; the planes, unpredictable; the drones, impossible. The copter coasts along with a good supply of inertia to keep things tense. But the limited repertoire of a cartridge-based game makes it a bit boring after the first few hours of play. Cartridge from Broderbund.

Snakman

Put *Pac-Man* in a Xerox machine, and this is what you might expect to get. *Snakman* is actually easier to play and master than is the original. The maze is smaller, the excitement more elusive. Cassette by Microdigital.

Frogger

This game may be the best translation of a real arcade game to a home computer. Graphics are great. The endless musical accompaniment is a delight—for a while. Lucky for your ears and sanity, it's easily switched off. The object (should you never have seen the game) is to pilot a frog across both a road

and an alligator-infested river to a safe home. Sounds simple, but as the traffic gets denser and threats loom in the lake, the challenge grows and grows. Genuine fun. Disk from Sierra On-Line.

Gridder

Your joystick controls a solitary character that "paints" a grid, changing its color as he passes, while being chased by spiders—at first two, then three and four. Though the graphics are simple and the sound moronic, the spiders behave with realistic predictability to make this a game of genuine strategy. Though modest, it has become a favorite. Cassette from Microdigital.

Labyrinth of the Creator

A cross between arcade and adventure game, *Labyrinth* is a maze of rooms infested with all sorts of fiends and foes you must adventure through to reach a goal. It's challenging enough and seems to be habit-forming. Is the second time through as exciting as the first? You may never know. Cassette by Victory Software.

Shamus

This is a maze-adventure game like *Labyrinth,* but the action is faster, the graphics better, and the whole even more addicting. The only way through is rapid-fire, blow-everything-up, plunge forward regardless. Disk from Synapse.

Jawbreaker

This is a game that might be fun if you're five years old but not much otherwise. *Jawbreaker* is an eat-the-dots *Pac-Man* variation with moving walls instead of a maze. Supposedly, it teaches you to brush your teeth after eating candy. Perhaps. Cartridge from Sierra On-Line.

Sammy Lightfoot

The concept is challenging: create a computer game that brings the excitement of a circus aerialist to the monitor screen. It borrows a bit from *Donkey Kong* along the way. The game is demanding, as every move of the joystick must be

precisely right. Alas, so precise that the game proves frustrating. Disk from Sierra On-Line.

Mr. Cool

Another Q-Bert clone, Mr. Cool is an ice cube that must hop down levels on a pyramid that change their colors along the way. Instead of snakes, etc., Mr. Cool is threatened with melting, etc. *Mr. Cool* proved to be an easy game to put away. Cartridge from Sierra On-Line.

Seafox

At first you may want to give this submarine action game the deep six, but it grows on you at about the same rate as you master it. The point is to maneuver your submarines, firing torpedoes to sink enemy subs before they get you. You score by sinking a convoy of surface ships. As you master one level, you face more challenges with depth charges and other surprises. Not high speed, but it is alluring. Cartridge by Broderbund.

Beach-Head

The perfect game to play if you must blow things up. *Beach-Head* offers a series of levels, each with its own goal. Pilot submarines through a mined maze; blow up a raft of attacking aircraft with ack-ack guns; sink an enemy convoy; maneuver tanks through an obstacle course; and match your vulnerable self against a final invulnerable cannon. At first, each level seems impossible; with practice, you can get most of the way through. Good graphics. Good shoot-em-up fun (if that's what you like). Disk from Access.

Skramble

This is another blast-the-enemy-off-the-face-of-the-earth game. You must pilot your plane through a web of missiles into a cave and onto a city. The graphics are both sufficient and second-rate, but the game is challenging. So challenging that even after several hours of play, advancing to the third level seemed impossible. Cassette by Microdigital.

Crossfire

Here is a fast-action, challenging game. You fire at aliens before they fire at you. The playing field resembles a block

map of intersecting city streets, and you've got to dash back and forth and perpendicularly to pursue your quarry and avoid their fire at speeds that seem like a thousand miles an hour. Disk from Sierra On-Line.

Quintic Warrior

If you like action as fast as the arcade game *Robotron* (in which you fire your laser at rates that make a machine gun seem like a popgun), you'll love this game. Your tiny spacecraft zips along blowing up others, all the while avoiding the crossfire of a protection-weapon-gone-wrong. Even at low speed your pulse will be racing. Cassette from Quicksilva.

Review copies of the following games were received but I couldn't make them work properly: *Fort Apocalypse* by Synapse; *Serpentine* by Broderbund; *Survivor* by Synapse; *Neutral Zone* by Access. I also bought *Radar Rat Race* from Commodore, only to discover the game cartridge actually contained *Kickman.* I was lucky—*Kickman,* I later discovered, is the better game. Moral: if the game doesn't work, visit your dealer.

Software Shopping

Compared to buying software for many other computers, the chore of selecting programs for your Commodore 64 is simple. While owners of more complex machines need to consider whether they have enough memory and the right accessory cards to bring their chosen program to life, you needn't worry. If a program comes on disk or tape, all you need is the proper peripheral for it.

At least for now, the Commodore 64 is relatively standardized. Software writers can therefore make certain assumptions about the 64 that will not vary from machine to machine. For instance, they know they have 64K of RAM memory and certain specific ports for getting data in and out. This means that for now you're nearly guaranteed that programs you buy that say Commodore 64 on them will run on your computer. Not all computer systems can make that claim.

As times change, however, so will the Commodore 64 and the accessories available for it. New programs will be written to take advantage of the specific features of new peripherals. Soon the Commodore 64's advantage in standardization will give way to the new virtue of adaptability, allowing the basic computer to do even more than its originators had ever intended.

The increase in versatility will bring new problems. When you pick up a package that says Commodore 64 on it, you'll have to be certain that the computer the program expects is outfitted exactly the same way as the one that you own. You might need a parallel port adapter, a Koalapad, an abstract radiation detector, or whatever special accessory to bring the software to life.

Cartridge, Cassette, or Disk?

Currently available software may come in one of three forms—on cartridge, on cassette tape, and on disk.

Cartridge programs have the big advantage of being nearly indestructible. Practically nothing you do can alter the programming instructions locked safely in the ROM memory chip inside the cartridge. The solid-state memory inside the cartridge will never wear out no matter how much you use it.

Cartridges can be used on the simplest Commodore 64 system of all—just the computer itself and a television set as a monitor. No peripherals like the Datassette or disk drive are needed to bring a program cartridge to life.

And cartridges are lightning fast. Because no mechanical parts are needed, data move in and out and all the way through them at nearly the speed of light. A program on cartridge means no waiting for a lengthy LOAD or even a long hunt by your disk-drive head.

Cartridges have a great big advantage for their makers, too. It's impossible for the typical Commodore 64 owner to copy them or even peek inside to see how the software instructions work. Since every machine that runs this program needs its own cartridge, each new user must buy it from the software company. That results in very little piracy, and the software company profits.

Unalterable cartridge programs do have their disadvantages, though. Just as you cannot change them accidentally, you can't change them on purpose, either. Most cartridges don't take into account that you may add some nifty accessory to supercharge your Commodore 64. They may obstinately ignore the wondrous addition or, worse yet, become so confused by the unexpected that the programs will reduce themselves to gibberish.

The big advantage of cassettes is cheapness. A Datassette is an inexpensive addition to any Commodore 64, and the cassettes you purchase to use on it are inexpensive, too, at

least when you compare the expense to that of disks and disk drives. Because duplicating cassettes is a rather common and uncritical affair (the technology is decades old; in fact, the same machines that copy stereo cassettes can make computer cassettes), cassette copies are generally less expensive than disk copies. And the low price of the Datassette makes it a prime target for software makers. Even the smallest computer system is likely to have a Datassette, so programs on cassette can reach a wide audience.

Although cassettes are subject to injury and can be damaged physically or by an inappropriate or accidental computer command, they are easy to "back up" by just copying the original with two tape recorders.

But how many times have you been told that the worst problem with cassette memory is its slowness? Cassettes are painfully slow. A long program can take many minutes to LOAD. Although the capacity of cassettes could be large, for practical reasons, cassette capacity is limited by LOAD time. Your kids might be grown up by the time their favorite game LOADs from cassette. No matter how patient you are, you won't want to wait 45 minutes for a program to get RUNning, especially when you want to use the program to save time!

Disks are delicate, quick, and more costly. You must take special care with disks. Because they are sheathed in a thin, flexible plastic shell, they can be accidentally bent, folded, stapled, or mutilated, any one of which is sufficient to ruin the disk and all the programs and data on it. You should always make a backup copy of a disk just in case the inevitable happens sooner than you expect.

Disks make LOADing programs quick—many times faster than cassette. Fast LOADing means longer programs can be dumped into your computer faster than you can lose interest in RUNning them. And the random-access ability of disks can help put bigger programs to work than would fit inside your Commodore's RAM memory. Only a portion of the program need be in memory if the computer can dig what it needs off the disk when it needs it.

The penalty is, as usual, expense. Disk drives cost four or five times as much as a Datassette. Disks generally cost more than cassettes. But the greater storage ability of disks actually makes them a better buy byte for byte.

Hardware Helpers

Add More Columns

You can widen your Commodore 64's display from the home computer-standard 40 characters wide to a professional business computer's 80-character width with an appropriate adapter card, such as the VideoPak from Data 20 Corporation (23011 Moulton Parkway, Laguna Hills, CA 92653). Of course, the primary reason for the Commodore 64's designers making the computer's display only 40 columns wide was that 40 columns is the most that the average color television can legibly display. If you opt for more columns, you'll need a high-resolution monitor as well.

Add More Cartridges

An expansion interface multiplies the number of cartridges you can plug into your Commodore 64 at one time. Normally, you're restricted to one, because the Commodore 64 has only one cartridge slot.

At first, an expansion interface seems like a pretty stupid idea. The reason the Commodore 64 has only one expansion slot is because it can use only one cartridge at a time. If you could connect two cartridges at a time, it is likely that your computer would become hopelessly confused.

But expansion interfaces for the Commodore 64 are not meant to run more than one cartridge simultaneously. They simply let you switch between cartridges. When using an expansion interface, you don't have to switch your computer off, remove a cartridge, plug another one in, and finally turn your computer back on to change program cartridges. Instead, just flick a few switches and press a reset button. Not only will it save a bit of time, but the expansion interface will likely save a lot of wear and tear on your Commodore 64's single expansion slot.

Is the little convenience worth the price? The people who make them seem to think so.

More Buttons

If you've ever done a lot of work with numbers on another computer, you'll note that the Commodore 64 lacks one big

necessity for really quick data entry: a numbers-only adding-machine-style keypad. But wherever there's a need (and a potential profit), peripheral suppliers are quick to leap in. If a computer means "number crunching" to you and you need a fast way of fingering in figures, choose any of the several add-on keypads that are available for the Commodore 64.

A Touching Experience

Do you think that the only reason computers have keyboards is because computers always have keyboards? Actually, the microprocessor within doesn't care where its commands come from. Keyboards can be convenient or confusing and time-consuming, depending on your experience.

The one perfect form of data entry for a computer has yet to be found. Some people think the answer is the *mouse,* a moving push button you roll across your desk (or a special mouse-pad) that indicates to your computer where to put its cursor. Others prefer to grab joysticks to zap aliens. Some like to draw with light pens.

The newest addition to the input armada is the KoalaPad, a touch tablet (from Koala Technologies Corporation, 3100 Patric Henry Dr., Santa Clara, CA 95050). Instead of mashing down dozens of keys on a typewriter keyboard, you sketch or point on the drawing pad. The pad has the necessary electronics built into it to translate the tactile commands of your fingers (or a stylus) on its pressure-sensitive surface into microprocessor computerese. Your nimble fingers can coax music from your Commodore 64's music synthesizer or draw or write on your monitor screen. It can even do duty as a paddle or joystick controller, with a quick-fingered difference.

An included disk-based program, *Koala Painter,* lets you use the KoalaPad to draw pictures on your Commodore 64's monitor screen without knowing a bit of programming. All you need is your finger as a paintbrush.

The Talking Computer

As good as the Commodore 64's built-in Sound Interface Device may be, it cannot (except by the furthest stretch of your imagination) mimic a human voice. But your Commodore 64 need not remain mute. For about $100, you can plug a voice-

synthesizing adapter into an appropriate slot and have your Commodore chattering away in a few minutes.

Both ease of use and sound quality vary from product to product. In any case, don't expect a voice synthesizer to look at printed words and instantly convert them into perfectly pronounced syllables. Either you must break words into phonemes (discrete individual sounds) or suffer through some unusual mispronunciations. You'll want to have some programming experience before you begin experimenting.

One source for a speech synthesizer—the *VoiceBox*—for the Commodore 64 is The Alien Group, 27 West 23rd St., New York, NY 10010.

More Information and User Support: Magazines

Technology and computers are changing so fast that book publishers are hard pressed to keep up with them. By the time most books are published, improvements and other modifications change products so much that books may no longer be accurate. New programs are being written every day.

There are two good ways of keeping abreast of the latest Commodore-related developments. If you have a modem and the patience and occasion to use it, you can check with the Commodore Information Network on CompuServe. Usually, however, your quest for knowledge and your access to a modem do not coincide. The answer is a magazine that focuses on your favorite computer.

A few of the more readable, interesting, and available magazines that offer news and features about the Commodore 64 include:

Commander

This magazine deals with all Commodore computers and tries to cover all areas of potential interest to users. With the rapid increase in popularity of the Commodore 64, an ever-increasing portion of the magazine is devoted to that machine. It includes typical computer magazine fare, from hardware and software reviews to tutorials and news about the Commodore line of computers.

COMMANDER
Micro Systems Specialties
P.O. Box 98827
Tacoma, WA 98498

Commodore

Commodore is the official voice of the Commodore company, *Commodore,* as you might expect, is a bit biased in favor of the company's products. In its pages, you'll find all of the good news and little of the bad. The emphasis is on getting you to buy and use more Commodore products. You won't find a lot of program listings; rather, the magazine concerns itself mostly with applications. Although the magazine covers all Commodore computers, the 64 is receiving more and more space. After all, it is their best seller.

COMMODORE—THE MICROCOMPUTER MAGAZINE
Commodore Business Machines, Inc.
Magazine Subscription Dept.
P.O. Box 651
Holmes, PA 19043
Subscription order line: (800) 345-8112
in Pennsylvania: (800) 662-2444

Power/Play

This fun-loving sibling of *Commodore* is published by the people who make your favorite computer. As you can guess from the name, emphasis is on games, entertainment, and having fun with your computer. Don't bother looking for it on the newsstand. When you send in your 64's warranty registration card, Commodore will send you a complimentary subscription (address as listed above).

Compute!

Compute! covers the entire lineup of home computers, including Apple; Atari; Commodore PET, VIC-20, and 64; Timex/Sinclair and the now-discontinued Texas Instruments 99. Its specialties are games and type-them-into-your-own-computer programs. Each issue may have dozens of program listings, including several for the 64. You get reviews, news, and columns, all written in a simple, clear style that even children can understand.

COMPUTE! THE JOURNAL OF PROGRESSIVE COMPUTING
P.O. Box 5406
Greensboro, NC 27403
Subscription order line: (800) 334-0868

Compute!'s Gazette

An offshoot of *Compute!*, the *Gazette* deals exclusively with the Commodore VIC-20 and 64 computers. If you're patient at typing and debugging, you'll find a bonanza of program listings between its covers. They're mostly games, and all are tried and tested by the magazine. You'll also find the latest Commodore news and tutorials that will help you understand the machine better, all written in the same easy-to-understand (if not elementary) style as is *Compute!* magazine (same address as above).

Creative Computing

Here is a magazine of general interest, with excellent news coverage of the whole world of computers and a monthly column devoted exclusively to Commodore products. The magazine is particularly strong on coverage of games and educational software.

CREATIVE COMPUTING
P.O. Box 5124
Boulder, CO 80321
Subscription order line: (800) 631-8112

User Groups

A user group is just a gathering of computer users who get together occasionally to swap tips, programs, and horror stories. Most user groups are local organizations, friendly clubs that meet in one or another member's home periodically.

Some user groups have national and international memberships. The bigger groups have vast software libraries whose programs are usually available to members at a minimal cost, which covers the expense of duplication. Many publish magazines for their memberships to exchange news, tips, discoveries, and headaches.

Although devoted to all Commodore computers, the Toronto PET Users Group has a large Commodore 64 chapter

and a growing library of Commodore 64 programs. Many of the programs in its PET library can be converted to run on the Commodore 64 as well. The group publishes a nearly monthly magazine, *The Torpet*, which is not nearly as slick as commercial computer magazines but is chock-full of information on the particular brand of interest. Issues of late have devoted a great deal of space to the Commodore 64. Although full membership implies the attendance meetings, the group allows associate (nonattending) members to share programs and receive the magazine.

Toronto PET Users Group, Inc.
1912A Avenue Rd., Suite 1
Toronto, Ontario M5M 4A1 Canada
(416) 782-8900

The Commodore 64 Users Group is a nationwide organization that specializes in what probably is by now your favorite computer. Besides a monthly newsletter, the group publishes a bimonthly magazine and gives reports of recent articles published on the Commodore 64. The group also has a growing library of public domain software available to its members.

The Commodore 64 Users Group
P.O. Box 572
Glen Ellyn, IL 60137
(312) 790-4320 (during business hours)

INDEX

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Y

Z

About the Author

Dr. Rosch is a contributing editor to *PC: The Independent Guide to IBM Personal Computers* and *PCjr* magazines. He is an attorney and a member of the instructional staff at the Cleveland Institute of Electronics. He started his engineering career building radio and television stations, and for the last ten years has written regularly about consumer electronics for *The Cleveland Plain Dealer* and a variety of technical publications. Dr Rosch is also the author of *Introduction to Digital Circuits* and *Introduction to Microprocessors*.

ALL THE C-64 INFORMATION YOU'LL EVER NEED!

THE COMMODORE 64 SURVIVAL MANUAL

THE NEW COMPLETE GUIDE—FROM PROGRAMMING TO PROBLEM SOLVING!

- **MEET THE C-64**—What your machine can do, what you can hook up, applications, shopping tips.
- **CREATING AND CONNECTING A SYSTEM**—Monitors, printers, plotters, cassette recorders, disk drives, modems, paddles, joysticks, light pens, graphic tablets, and more.
- **GETTING STARTED**—The Commodore command center, keyboard hints, customizing color and character sets, exploring memory, a hands-on peek inside.
- **PROGRAMMING STEP-BY-STEP**—Creating exciting programs, mastering color and sound, learning computer logic, animating your screen with sprites.
- **STORAGE SECRETS**—Cassette and disk loading, saving, copying, testing, finding, naming, organizing, and more.
- **COMMODORE PRINTERS**—Typeface tricks, special effects, playing with characters, customization, graphics magic, solving problems.
- **MAKING THE MOST OF MODEMS**—Plugging in to national networks, auto-dialing, talking to bulletin boards and other computers, downloading free software.
- **CARE AND FEEDING**—Important precautions, power line problems, static protection, disk and cassette care, money-saving tips.
- **TROUBLESHOOTING**—Dozens of symptoms, error message tips and specific cures, hardware and software problems, and how to spot and fix them.
- **SORTING THROUGH SOFTWARE**—How to find the best word processors, enhancement programs, spreadsheets, databases, accounting packages, languages, utilities, games, plus shopping secrets.
- **HARDWARE HELPERS**—Adding more peripherals, using digitizers and touch pads, making your computer talk.
- **USER SUPPORT**—Magazines and user groups, help hotlines, and hundreds of valuable tips you won't find anywhere else.

34127-8 • $9.95 (IN CANADA $10.95) ★ A BANTAM COMPUTER BOOK