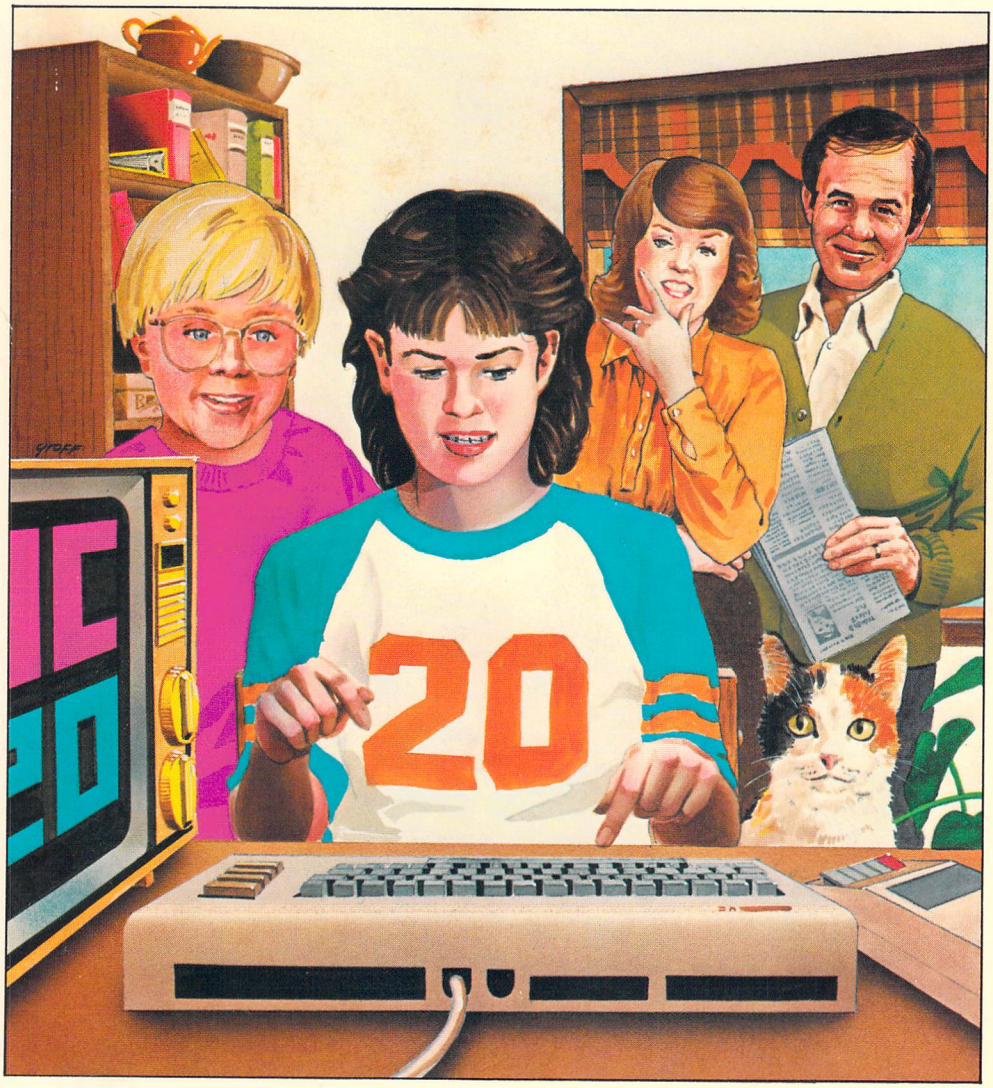
dilithium Press

How to Use The VIC 20™ Computer



Jerry Willis and Deborrah Willis

How to Use the VIC 20™ Computer

How to Use the VIC 20™ Computer

Jerry Willis and Deborrah Willis

dilithium Press
Beaverton, Oregon

10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging in Publication Data

Willis, Jerry.
How to use the VIC 20 computer.

Includes index.
1. VIC 20 (Computer)—Programming. 2. Basic (Computing program language) I. Willis, Deborrah. II. Title. III. Title: How to use the V.I.C. 20 computer.
QA76.8.V5W54 1984 001.64 83-17147
ISBN 0-88056-134-3

Cover: Vernon G. Groff

Printed in the United States of America

dilithium Press
8285 S.W. Nimbus
Suite 151
Beaverton, Oregon 97005

Trademark Acknowledgements

Apple II	Apple Computer, Inc.
ATARI 400, 800	Atari Computer
SuperCalc	Sorcim Corp.
TI-99/4A	Texas Instruments
TRS-80 Color Computer	Tandy Radio Shack Corp.
VIC 20, PET, Commodore 64	Commodore Computer
VisiCalc	VisiCorp
IBM PC	International Business Machines Corp.

Table of Contents

Chapter 1

Well, What Do We Have Here?

This book is aimed at the novice computer user who owns, or is thinking of buying the VIC 20 computer from Commodore.

The VIC 20 was a revolutionary computer when Commodore President Jack Tramul announced its introduction in 1980 at a London meeting of Commodore executives. The machine has a full size, typewriter-style keyboard, can be programmed in a standard version of the most popular computer language, BASIC, and has very good color graphics and sound synthesis features. None of the features noted above, however, are revolutionary. The revolution was the price—$299. In 1980, no other company had offered so much computing power for so small a price.

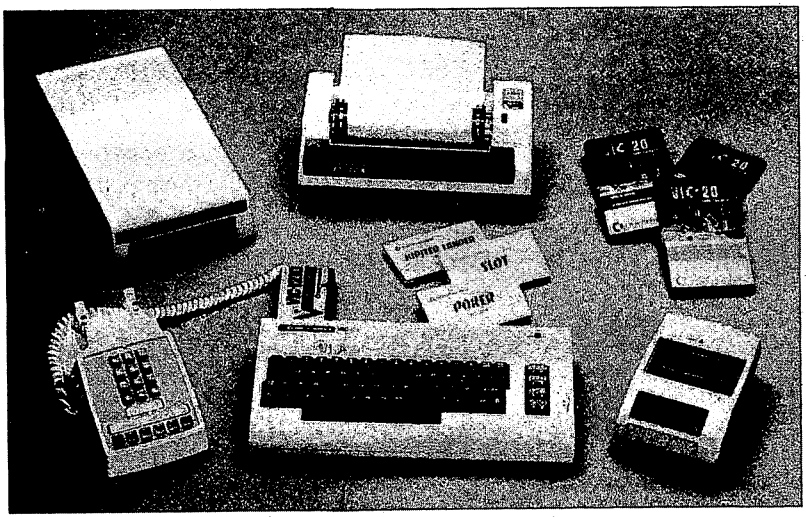


Figure 1.1 The VIC 20 and accessories.

By the summer of 1983, the VIC 20 had become one of the best selling computers ever, with sales estimated to be between 5 and 1.5 million. The revolutionary price of $299 had dropped to as little as $86 and the computer could be purchased in such mass market outlets as K-Mart, Montgomery Ward, and Toys-R-Us. The VIC 20 competes today in a crowded market where the TI-99/4A, the TRS-80 Color Computer, and the ATARI 400 all sell for under $100.

Is a computer that costs less than $100 to be considered seriously? Will it do anything besides let you plug in video-game cartridges and play space fantasies? The answer is a definite YES. The VIC 20 is a real computer and can do plenty in your home, office, or classroom. This book is geared to the beginning computer user with little background in computer science or electronics. It will provide you with the information needed to make maximum use of the VIC 20. We wrote the book with the following objectives in mind:

1. To introduce you to the computer and its basic components, what they do, and how they work together (Chapters 1, 9 and 10).

2. To provide an overview of the things you can do with your computer (Chapter 1).

3. To give step-by-step instructions on how to set up or "install" your computer and tell you how to get it "up and running" (Chapters 2 and 3).

4. To describe how to load and save programs on standard audio cassettes and diskettes (Chapters 4 and 5).

5. To introduce the novice to the standard computer languages and to the use of color graphics and sound synthesis with the VIC 20 (Chapters 6, 7 and 8).

6. To provide information on what accessories you may want to buy for your computer and how to select, buy, install, and use them (Chapter 8).

7. To provide information on other sources of information about your computer such as magazines, books, user's groups, and computer-based information networks (Chapter 12).

8. To provide information on the software available for the VIC 20, how to select the best programs for your needs, and where to buy it (Chapter 10).

If you are eager to get started using the computer, move ahead to Chapter 2.

A LITTLE HISTORY

In 1974, Commodore International was known as a successful manufacturer of calculators. It was a relatively small company that had demonstrated its ability to match development and marketing skills with giants, such as Texas Instruments, and hold its own.

The development of hand-held calculators in the early '70s required manufacturers to design systems that made use of electronic devices called "integrated circuits" or ICs. These ICs enabled designers to replace hundreds of discrete transistor circuits with three or four integrated circuits. A standard integrated circuit could perform all the functions of 50 to 200 transistors and it drastically reduced the steps and time required to manufacture a calculator. Today, the technology of creating integrated circuits has reached the point where many hand-held calculators have only one large integrated circuit in them. ICs routinely are manufactured that replace tens of thousands of individual transistor circuits.

The little IC, which was the foundation of the calculator industry, was also the foundation of the personal computer revolution. One of Commodore's chip suppliers, MOS Technology, was also a pioneer in the development of the "computer on a chip," also known as the microprocessor integrated circuit. MOS developed the 6502 integrated circuit or "chip" which is used in many of the most popular computers. The Apple II, the ATARI 800 and 400, the Commodore PET, and the VIC 20 all use the 6502 chip.

When Commodore decided to move into the developing field of personal computers, it bought MOS Technology and thus created a company that had both marketing experience (from its calculator background) and technical expertise (from its chip development work). The result was a remarkably successful product, the PET computer. In 1977, Commodore announced the PET, a $495 computer that was the first "appliance" computer. That is, it was a computer that could be taken home, plugged in, and used immediately.

Although the original PET had a hard-to-use calculator style keyboard, orders exceeded all expectations and almost overnight Commodore become one of the major players in the personal computer game. Some would say the PET was too successful. The company was not prepared for the product demand and as orders came in the time until delivery also increased. At one time, Commodore had a five-month backlog of orders. Commodore, in fact, was never able to meet production schedules, establish a strong dealer network, and provide necessary support. Their share of the market in the United States dropped steadily after the first year. Today

it is not uncommon to find bitter computer dealers and PET purchasers who remember the problems of Commodore during the 1977-1981 period. Phone calls were not returned, accessories were announced but not manufactured, problems occurred because of design errors, and a generally slip-shod mode of operation created a poor image of Commodore in the minds of many.

While Commodore was developing a poor reputation in the U.S., it was moving toward domination of several major European markets. That difference is based on two facts—the computers they built were fine products, and the marketing strategies used in Europe were excellent, especially when compared to the ill-conceived U.S. promotions. Between 1977 and 1981, Commodore expanded its original PET computer into a line that included five different models with four different styles of disk drives. The 8032 computer, when combined with an 8050 disk drive, was one of the best computers in the small business market in 1981.

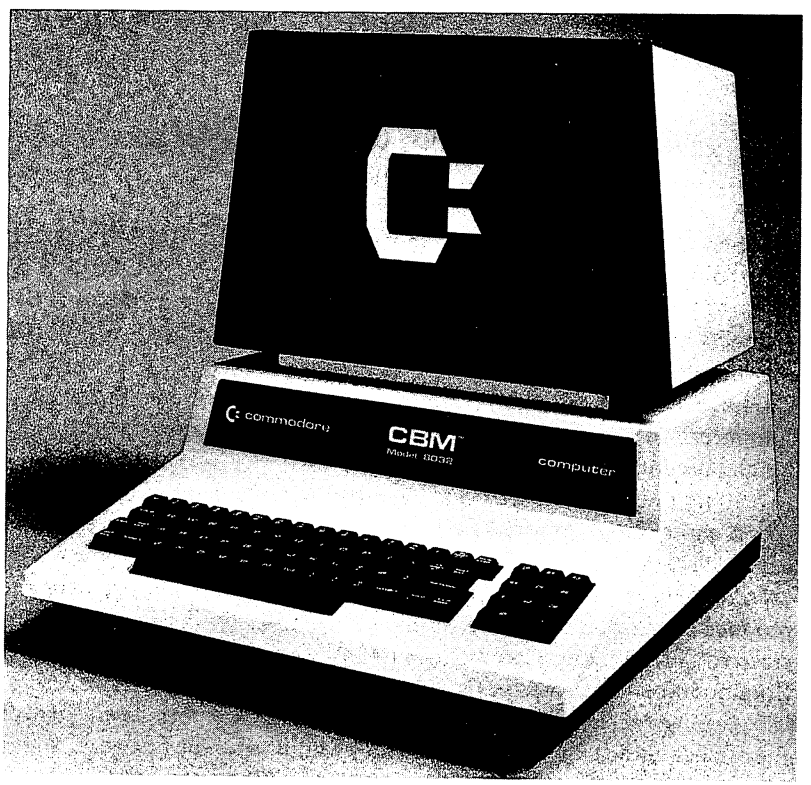


Figure 1.2 The Commodore 8032.

By 1981, however, good equipment was not enough. Competitors also had good machines and they answered their phones, gave strong support to their dealer network, and were generally easier to deal with. Commodore still competes in the business market, although it has only a small percentage of the sales despite its excellent products. Dealers have long memories and Commodore still has problems with support and timely production of announced products.

By 1981, the rush to buy computers for the home provided Commodore with another market—the "under $300 home computer." Commodore used the design and manufacturing expertise of MOS Technology to create the VIC 20, an inexpensive but powerful computer. Commodore introduced the VIC 20 in Japan in 1980 and began U.S. marketing in 1981. With the arrival of the VIC 20, a prospective purchaser could get a real computer for only slightly more than the cost of a video game. Commodore made much of that fact and used Star Trek's Captain Kirk (William Shatner) in an advertising campaign that cost millions and sold hundreds of thousands of VIC 20 computers. The VIC 20 can be (and is) used for everything from video games to stock market speculation.

Commodore's VIC 20 computer is a front runner in the race to put computers in the home. It is also a front runner in the mass marketing of computers. Few people buy a VIC 20 at a computer store. They are more likely to get it at J.C. Penney or K-Mart, outlets likely to be long on discounts and short on technical advice and support. We hope this book will help fill the gap and provide you with the information needed to use the computer effectively.

WHAT CAN YOU DO WITH IT?

Many people are downright skeptical about inexpensive computers. We are accustomed to computers that cost millions of dollars and require teams of highly trained specialists to run them. Can the tiny VIC 20 really do much? This section presents an overview of the major areas of application with an emphasis on products and software for the VIC 20 that are available now. Later chapters cover software and accessories in detail.

COMPUTER LITERACY

In spite of its low price, this computer works in essentially the same way as the bigger, more expensive computer systems. It can cost from $100 to $500 to attend a good course or workshop on computer literacy where you learn the fundamentals of computer usage.

The VIC 20 is not a computer to learn and discard, however. You can spend around $250 on the basic unit, a cassette recorder, and a few programs of interest and begin learning how computers operate. Add $50

to $100 for books and training material and you have less than $400 invested. After a few months of work, however, you may decide to begin using the computer for some serious computer applications, several of which will be discussed later in this section. You need not banish the VIC 20 to a dusty corner of your closet. You can add accessories to the computer and, for less than $800, the VIC 20 becomes a powerful computer capable of performing many important tasks in your home, business, or school.

BASIC PROGRAMMING

Many new computer owners want to use their machines to do some useful work. Others use them for recreation, and some want to learn how to "program" the computer. Programming involves typing lines of instructions into the computer. In order to program the VIC 20, you need to know how to speak its language. The VIC 20 uses BASIC as its primary programming language. BASIC stands for Beginners All-Purpose Symbolic Instruction Code. There are many different computer languages in use today, but BASIC is by far the most popular. BASIC is actually a family of computer languages with each computer designed to speak a particular dialect. The BASIC understood by the VIC 20 computer is a fairly standard version which is also used on other computers manufactured by Commodore. If you learn BASIC on the VIC 20, you will also be able to program several other models with little difficulty.

There are at least 20 books currently in print on programming the VIC 20 in BASIC. You will probably be able to browse through at least five or six at a well-stocked bookstore. In addition, Commodore also sells a package of materials on programming the VIC 20 that uses the computer to teach you programming. Their *Introduction to BASIC, Part 1 and Part 2* costs less than $30 and is an effective way to learn BASIC on the computer.

MACHINE LANGUAGE/ASSEMBLY LANGUAGE PROGRAMMING

It is also possible to write programs for the VIC 20 in a computer language called 6502 machine language. This is a versatile language and one that is used by commercial programmers who create video-game cartridges and word-processing programs. It is not, however, an easy language to learn. We would suggest that you start with BASIC and put off tackling machine language until you have more experience.

Human Engineered Software (71 Park Lane, Brisbane, California 94005, phone 415-468-4110) also markets HES MON, a cartridge that makes it easier to write programs in 6502 machine language and in assembly language, a close cousin to machine language. The same company sells the

HES 6502 Professional Development System, a $30 program that lets you write programs in assembly language.

FUN AND GAMES

Set aside eight hours a day to play all the games available for the VIC 20 computer, and it would probably take a year to play each game just once. And by the time you finished playing all that are available now, there would be hundreds of new ones. Our guess is that 50% of VIC 20 owners use the computer primarily for fun and games. Several magazines regularly publish at least one or two game programs for the VIC 20 in each issue (see Chapter 7), and hundreds of small (and not so small) companies sell game programs for the VIC 20.

The commercial programs come in any of four forms: on a cassette, on a diskette, in a plug-in cartridge, or in a book. If you buy a program on cassette or diskette you must "load" the program into the compuer from the cassette player or disk drive each time you use it. A cartridge, on the other hand, need only be plugged into the back of the compuer. Books that contain listings of programs that work on the VIC 20 require you to type in the program you want to use. Once it is typed in, however, you can create a permanent copy of the program on a diskette or tape and use that copy thereafter.

Many of the best recreational programs take full advantage of the color graphics and sound synthesis possible on the VIC 20. Most also let you use the joy stick rather than the keyboard to move players around on the screen. Below is a sample of the recreational software available for the VIC 20. Remember, however, that it is only a sample; there are hundreds of programs available.

Munchmaid. This program is very close to another popular arcade game, PAC-MAN, but runs on the VIC 20. It costs $11 on cassette and $15 on diskette. Wunderware (P.O. Box 1287, Jacksonville, Oregon 97530, phone 503-899-7549).

Hidden Maze. This program puts you in the middle of a large maze from which you must extricate yourself. The screen shows only a small portion of the maze; thus it is necessary to create an image in your head of where you are, where you've been, and likely avenues of escape. The VIC 20 version uses a joystick to control movement through the maze. The program was written by Gary Boden and was published in *Compute!* Magazine (December, 1982, pp. 152-159). If you have a copy of the magazine, the program is free for the trouble of typing it into your computer. This magazine regularly publishes VIC 20 programs.

Exterminator. Centipedes and spiders are all over the screen in this game. It is your job to shoot all of them down as quickly as possible—at least before they get you. The game makes good use of VIC 20's color graphics; it is very fast, and it has sound effects. At $25, it is a popular game for the VIC 20. Nufekop Software (P.O. Box 158, Shady Cove, Oregon 97539).

Spiders of Mars. This $60 game comes in a cartridge and is very similar to an arcade game called Defenders. You are a fly on the planet of Mars and there are all sorts of insects out to get you, including spiders, bats, and dragonflies. You must shoot down your adversaries to score points. The game has very good color graphics and sound. It is available from UMI (3503 Temple Avenue, Pomona, California 91768).

HOME AND SCHOOL APPLICATIONS

The VIC 20 can perform many important jobs around the home and serve as an electronic tutor as well. Some of the programs that put the computer to work are described below:

Tax Helper. This program helps prepare your federal income tax return. It requires extra memory and costs around $25. Magreeable Software, Inc. (5925 Magnolia Lane, Plymouth, Minnesota, phone 612-559-1108).

Math Tutor Series. This series of four programs for $27.95 is standard "drill and practice" software that teaches children the basics of simple math operations. They are available from Comm*Data Computer House (320 Summit Avenue, Milford, Michigan 48042, phone 313-685-0113).

Alpha-Beci This $17 program is available on cassette and teaches children to recognize the alphabet. Well done screen displays show the upper and lower case letter and a picture. The company also offers a number recognition program in the same format. Boston Educational Computing, Inc. (78 Dartmouth Street, Boston, Massachusetts 02116 617-536-5115).

State Capitals. This program teaches the capitals of states. It is $10 from American Peripherals (122 Bangor Street, Lindenhurst, New York 11757, phone 516-226-5849). This company has a complete line of software for Commodore computers and many educational programs are in their catalog. Another company with lots of educational software for the VIC 20 is Micrograms (P.O. Box 2146, Loves Park, Illinois 61130, phone 815-965-2464).

Typing Tutor. A cassette program that costs $22, this one teaches you how to touch type on the VIC 20 keyboard. Rated highly by several

reviewers it comes on a diskette with another program called Word Invaders. Academy Software (P.O. Box 9403, San Rafael, California 94912, phone 415-499-0850).

BUSINESS AND PROFESSIONAL APPLICATIONS

The VIC 20 computer was not designed as a business computer. Color graphics, sound synthesis, and joystick connections are features that make it a popular home and educational computer, but it lacks some features that are desirable in a business computer. The video display, for example, displays only 22 lines of 23 characters. That is not enough for most business applications. Computers that sell well in the business market generally have a 24 line by 80 character display. The Apple II has 24 lines of 40 characters and is popular in small businesses, but would fare better with a higher capacity display. The ability to see a lot of material at once is important in many business applications and the VIC 20 falls short on video capacity.

Business programs also generally require a lot more memory than the unadorned VIC 20 possesses, but that limitation can be corrected by plugging in extra memory cartridges. You can also add a disk drive to the VIC 20 for data storage. (Few business programs run well with cassette storage systems.) The disk drive for the VIC 20 is, however, relatively slow. All in all, we do not see business applications as an area where the VIC 20 will outperform other computers.

Having said that, however, we must admit that there is a considerable amount of business software, and more than a few brave souls regularly use their VIC 20 in business or professional applications. Several companies, in fact, sell add-on kits that let you convert the video display of the VIC 20 to 24 lines by 40 characters and/or 24 lines by 80 characters. When it comes to using the VIC 20 as a business computer, it is possible if you are willing to spend the money for the job. For most people we recommend another approach—let the VIC 20 do what it does best and buy another computer for the business applications. That advice, however, is not without a price since business-oriented computers tend to cost much more than the VIC 20. If you find yourself in the position of needing an inexpensive computer that will do many things, it will be possible to make do with the VIC 20. Here are some of the business programs that run on the VIC 20:

WORD-PROCESSING PROGRAMS

Several companies offer VIC 20 owners the opportunity to use their computer to compose and print documents that range from simple letters to long reports, even book manuscripts. Virtually all these programs

require extra memory. Several are also compatible with the accessories that expand the video display of the VIC 20 and many will work only if you have the 1540 or 1541 disk drive. Word processors also assume you will have a printer connected to the computer so you can print copies of the material you type.

A good word-processing program lets you compose, edit, and update written material with less effort and more speed than a typewriter. Essentially you create your document on the screen of the video display where changes are easily made. When you are ready, the computer can create a copy of the document on your printer. In addition, you can save the document you have created on a cassette or diskette and load it back into the computer at a later date either for revisions or to print another copy.

As noted above, a word-processing system will require much more equipment than the VIC 20 keyboard. You need a disk drive, extra memory, a printer, and a word-processing program. See Chapter 10 for information on the extra equipment needed. Some of the word-processing programs available for the VIC 20 are listed below:

Quick Brown Fox. This $65 program works on standard VIC 20 computers or those with expanded video display features. It is available from Quick Brown Fox (548 Broadway, New York, New York 10012, phone 212-925-8290).

Word Master. This program is supplied on tape but can be converted to diskette by the user. It is a combination word-processing program and mailing-list program. This program is sold by a company that manufactures enhanced video display circuits for the VIC 20. The program is free when you buy a video display enhancement. Data 20 Corporation (23011 Moulton Parkway, Suite B10, Laguana Hills, California 92653).

Word Wizard. This $35 program runs on a VIC 20 with at least 8K of memory (K stands for 1024). It can be used with cassette tape or disk systems. MicroWave (1342 B Rt. 2, Butler, New Jersey 07405, phone 201-838-9027).

Wordcraft 20. This program is an abbreviated version of a program that runs on PET computers. It offers sophisticated word-processing features to owners of VIC 20 computers. UMI (3503C Temple Avenue, Pomona, California 91768, phone 714-594-1351).

HES Writer. This cartridge program sells for $35 and is considered by many to be an excellent word-processing program for the VIC 20. HES (71 Park Lane, Brisbane, California 94005, phone 415-468-4110).

ELECTRONIC SPREADSHEET SOFTWARE

Many business applications require the calculation of large tables or charts of related figures. Loan amortization tables, salary schedules, insurance fee tables, inventory order records, travel expense forms, and job cost estimates all require many tedious calculations. Electronic spreadsheet programs such as VisiCalc and SuperCalc have earned millions of dollars for their creators because so many business people need a convenient, quick way of doing such jobs. Although generally not as powerful as the spreadsheet software for larger computers, there are several useful spreadsheet programs for the VIC 20. Keep in mind that spreadsheet programs tend to use large amounts of computer memory. You are likely to need all the memory your computer can accept if you use spreadsheet software.

BusiCalc. This spreadsheet program costs $50 and is available from Skyles Electric Works (231G South Whisman Road, Mountain View, California 94041, phone 800-227-9998 or 415-965-1735).

ACCOUNTING SOFTWARE

A few software suppliers offer programs in this area for the VIC 20 computer. One company, Micro Spec (2905 Ports O'Call Court, Plano, Texas 75075, phone 214-867-1333), offers the following disk-based programs:

- Payroll Systems (including check printing)—$60 for VIC 20, $80 for the Commodore 64.
- Mailing List—$45 for VIC 20, $100 for CBM 64.
- Inventory Package—$80 for VIC 20, $100 for CBM 64.
- General Ledger—Handles up to 75 different accounts—$90 for VIC 20, $100 for CBM 64.
- Checkbook Manager—Handles up to 25 different expense categories—$50.
- Disk Data Manager—lets you organize, store, and retrieve information in a convenient format—$60 for VIC 20, $80 for the CBM 64.

Other companies that offer accounting and/or word-processing software for the VIC 20 include:

Totl Software (1555 Third Street, Walnut Creek, California 94596, phone 415-943-7877).

Powerbyte Software (2 Chipley Run, West Berlin, New Jersey 08091, phone 609-346-3063).

Programmer's Institute (P.O. Box 3470, Chapel Hill, North Carolina 27514, phone 919-967-0861).

TELECOMMUNICATIONS

A growing area of interest for home computer owners is telecommunications. Telecomputing involves connecting your home computer to other computers via phone lines. Your phone then becomes a link to hundreds, perhaps thousands, of other computers programmed to provide all sorts of services to personal computer owners. You can sit back in your easy chair, get a comfortable grip on the keyboard and do everything from pay bills to read weather reports for Colorado ski resorts. Telecomputing is, in fact, one of the fastest-growing areas of personal computer use. Telecomputing lets you dial up local area networks and community bulletin boards which are run by colleges, computer clubs, amateur radio clubs, and other special interest groups.

You can also use services provided by several national "information utilities" that let you see on the screen everything from stock market information to the current AP or UPI newswire. You can use your computer to purchase products, check on airline schedules, get reviews of current movies, and look for research or articles on topics of interest to you. Access to the *World Book Encyclopedia* is even available on one popular information utility called The Source. Here are the addresses of the most active information utilities:

The Source (1616 Anderson Road, McLean, Virginia 22102, phone 703-821-6660).

CompuServe, Inc. (5000 Arlington Centre Boulevard, Columbus, Ohio 43220, phone 614-457-8600).

Bibliographic Retrieval Services (1200 Route 7, Latham, New York 12110, phone 518-783-1161).

DIALOG Information Services (3460 Hillview Avenue, Palo Alto, California 94304, phone 415-858-3810).

If you need access to a particular type of information, the book *Directory of Online Databases* (Cuadra Associates) or *Directory of Online Information Resources* (CSG Press) lists hundreds of information utilities you can contact via computer.

One of the most popular information utilities, CompuServe, is also the host for the Commodore Information Network. This service gives you access to up-to-date information on Commodore products and programs.

You can even type in a question and get an answer from a Commodore technician. The March, 1983, issue of *Commodore Magazine* has an excellent article by Jeff Hand on using the Commodore Information Network.

If you want to use the VIC 20 for telecomputing, you will need a modem, software and an account number. You get an account number from the information utility you plan to use. There is usually a one-time registration fee and a monthly bill for the time used each month. The other item, a modem, is a device that converts signals from the computer to signals that can be transmitted over telephone lines. Modems cost between $100 and $600 depending on features and the profit margin of the seller. VIC 20 owners, however, have a bonus in this area. Commodore sells the VICmodem for $100. It plugs into your VIC 20 and comes with the software needed to let the VIC 20 communicate with most popular information utilities. You even get some free time on CompuServe.

The VIC 20 and the VICmodem are the only equipment you will need for telecomputing, but many VIC 20 users find the 22 character lines on the VIC 20 a bit limiting. With 22 lines of 23 characters you really don't get that much information on the screen at one time, and several services are not set up to format the lines of data coming to your computer so that they are no more than 23 characters long. To deal with this problem Midwest Micro Associates (P.O. Box 6148, Kansas City, Missouri 64110, phone 816-254-9600) created Terminal-40. It is a cassette-based program that requires 8K of memory. It replaces the software that comes with the VICmodem (you still need the modem though) and lets the VIC 20 operate with 24 lines of 40 characters. It is an excellent value at $30.

The same company sells SuperTerm-40 for the VIC 20 that adds some of the features of a sophisticated computer terminal to your VIC 20.

Few people are likely to be interested in all the applications areas described in this chapter, but most readers will find at least one or two appealing. The popular computer magazines, several of which are described in the final chapter of this book, frequently carry articles on the applications areas described thus far. In addition, Chapter 11 lists book publishers who have hundreds of current titles. You can even use The Source or CompuServe to search for articles on topics of interest.

Chapter 2

Setup and Installation

Even the basic VIC 20 computer comes with accessories. There is the power supply, the cable that connects the television to the computer, an "RF modulator," and a switch box that lets you switch between regular TV reception and the computer display. This chapter will show you how to connect everything together and how to adjust the television set. If you have purchased a cassette recorder for your VIC 20 or a disk drive, instructions for installing these accessories also are provided.

CONNECTING THE COMPUTER TO THE TELEVISION

The VIC 20 can be connected to two different types of video displays. First, it has all the equipment needed to display information on an ordinary television set. It will work on either a color or black-and-white set. Most people attach the VIC 20 to a color set to take advantage of the computer's color graphics features.

The VIC 20 also can be connected to a video monitor. A color video monitor will produce much sharper color images than an ordinary color television but generally costs much more than a television. Few homes have an unused color monitor laying around just waiting for a VIC 20, but most homes do have a color TV. We will thus concentrate on connecting your VIC 20 to a television.

There are, however, many video monitors in schools since they are often used with video tape recorders. In addition, many of the new "component" television systems for the home have color monitors. The final part of this section offers advice on how to hook up the VIC 20 to a video monitor.

If you peek at the back of your VIC 20 computer, you will see five different connectors spread across the back. The wide, recessed slot on the back is where you can plug in all sorts of cartridges. It is called the Expansion Port. Next to it is a round connector called the Video Port. This connector has five holes in a semicircle at the bottom and an alignment slot

at the top of the circle. Round connectors like these are frequently used on European stereo systems and accept a "5 pin DIN connector."

You should have a cable about three feet long with a DIN plug on one end and a black box on the other. This black box is an RF modulator. It takes the sound and video signals produced by the VIC 20 and converts them to a signal the televison can receive on either Channel 2 or 3. There is a switch on the Radio Frequency (RF) modulator that lets you select the channel. If your city does not have a television station of Channel 2 or 3, the switch can be set to either 2 or 3. It should work fine on either one. If you have a Channel 2 or 3 in your area, use the switch to select an unused channel to minimize the chances of interference. If you have stations on both 2 and 3, select the channel with the weaker station. The switch on our modulator was not labelled so we had to guess which way to set the switch for Channel 2 or 3. You may have to experiment, too.

Plug the cord from the RF modulator into the Video Port on the VIC 20. Note that there is another DIN connector next to the Video Port, but it has six rather than five holes in it.

Now attach the TV switch box to the VHF connector on the back of the television. If you have an antenna lead or cable line connected to the VHF

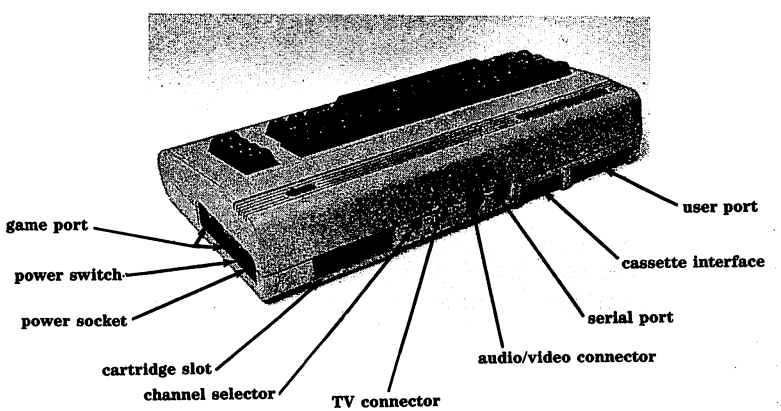


Figure 2.1 The back of the VIC 20.

terminals, remove it and attach it to the switch box. Use the terminals marked "antenna." Now, take the video cable that came with your VIC 20 and connect one end to the RF modulator. Connect the other to the TV switch box.

With the television connected to the RF modulator, you can now use the slide switch lever on the TV switch box to select the signal from the computer or from the antenna/cable lead. Set the switch box at the back of the television to "computer" for now.

Some of the earlier VIC 20 computers were less than agreeable with many models of televisions. The VIC 20, like many computers, produces a lot of electrical "noise" when it is on. Some of that noise may show up as picture interference. Commodore made some design changes in the VIC 20 to reduce the interference, but you may still have problems if your television is particularly sensitive. When you get your VIC 20 up and running, try moving the cords around to see if one position reduces interference. If you plan to buy a small television to use with the VIC 20, it is advisable to take the computer to the store and try it out before making the purchase (or get the store to agree that you can return it if it doesn't work well).

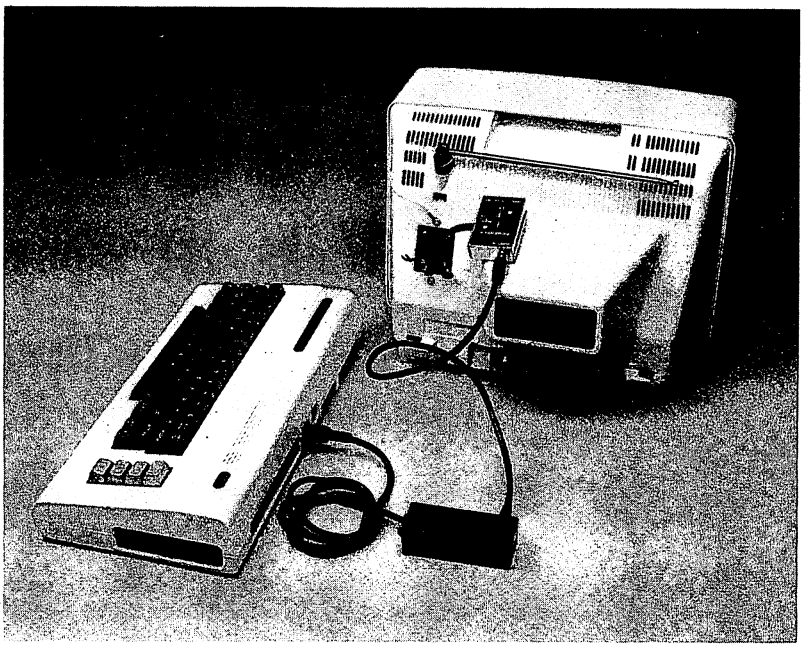


Figure 2.2 Connecting the VIC 20 to a television.

If you have a major problem with the quality of the display on your television, there are two tricks that may or may not help. First, you can replace the cable between the RF modulator and the switch box on the television with a high quality shielded cable that is available in video stores. The standard cable has RCA phone plug jacks. Better cables may use threaded "coax" connectors. This means you will need to buy two converters that attach to coax connectors and allow you to connect to RCA phone plugs. Both the cables and the converters are available at Radio Shack stores and at many video stores. If interference is "leaking" into the cable, switching to a higher quality, well-shielded cable may help.

Another little trick involves creating a loop in the cable that runs from the computer to the television. Coil four to six loops of the cable around your hand at the knuckles (as close to the TV end of the cable as possible) and increase or decrease the size of the loop until you see an improvement in the display. When you find a loop size that works best when your hand is removed, tape the loop so that it retains that size.

The Tape Recorder Connection

It is not absolutely necessary to buy a tape recorder to use the VIC 20, but if you plan to load programs stored on cassette tapes into the computer's memory or save on tape programs you write, you might as well set up the tape recorder now. (The final section of this chapter will deal with the issue of whether a cassette or disk drive is preferred.)

Commodore manufactures a special tape recorder that works with all of its computers. Although there are several different versions, the current

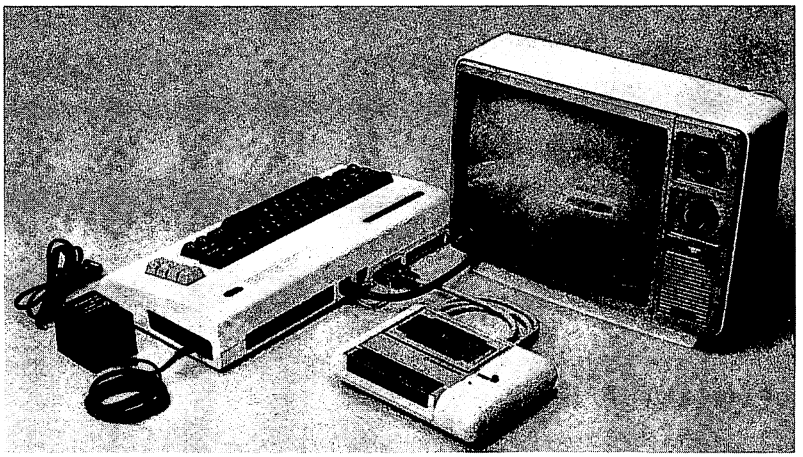


Figure 2.3 Hooking up the VIC 20.

model is white with a sculpted case. Its retail price is around $75. Some people complain about the price of the unit and about the fact that Commodore does not let you use a regular tape recorder with the computer while Radio Shack and Timex Sinclair do. Standard tape recorders tend to be very unreliable. The Commodore recorder has worked quite well for us. We won't defend the price of the C2N, but we will argue that computer systems that use standard tape recorders are usually far less reliable than computers such as the VIC 20 and the ATARI 800/400 that use a special recorder. (Note: If you plan on using a standard recorder see Chapter 9 for information on where to buy a converter.)

Attaching the C2N recorder to the VIC 20 is very straightforward. Look at the grey cable attached to the recorder. The flat plug on the end has six metal connector traces in it. Between the second and third connector is a tiny sliver of white plastic. If you look on the back of the VIC 20, the connector next to the unused DIN plug has six copper traces on it with a slot between the second and third trace. Orient the cable so that you can insert the plug into the VIC 20 with the white piece of plastic neatly fitting into the slot between trace 2 and 3. It should seat firmly. On our recorder that means the side of the plug with the little Commodore corporate logo

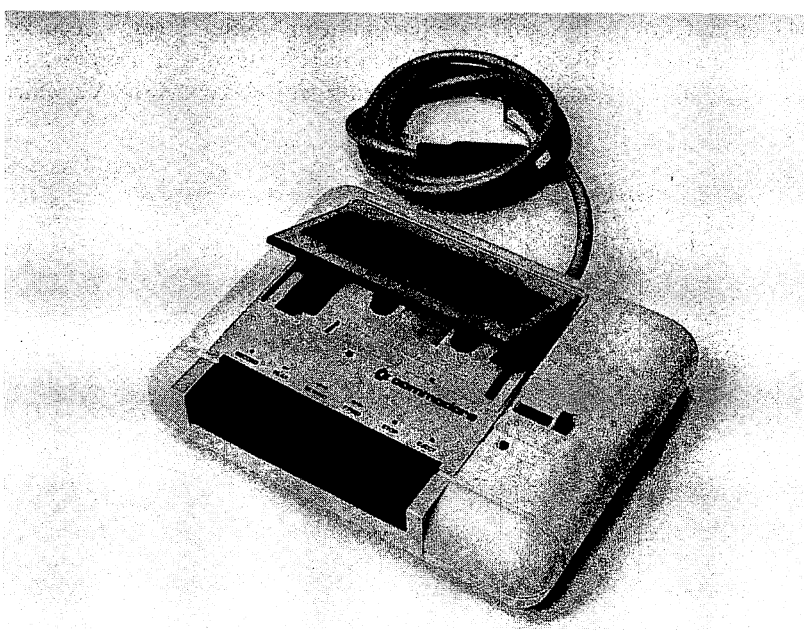


Figure 2.4 The C2N Cassette recorder.

embossed on it is up. Since the white plastic slivers tend to get mangled or lost with use, now would be a good time to scratch "top" on the cable end so that you will know how it should be oriented even if the sliver is missing.

With that, your recorder is properly connected. It gets its power from the computer and thus does not even need to be plugged into a wall outlet.

Installing a Disk Drive

See Chapter 4 for information on installing the disk drive.

The Power Connection

There is one more cable to be attached to the VIC 20. Take the power supply cable and plug one end into the connector on the back, right side of the VIC 20. Then plug the other end into a suitable wall outlet. Now, if you haven't done it already, plug in the television set and arrange the computer so that it is convenient for you to use while viewing the television screen. That's it! You are in business.

Take It For A Spin

All the connections now have been made (unless you splurged and bought a printer too). Turn on the television and turn the selector to either Channel 2 or 3 (whichever is unused or least used in your area). Now turn on the VIC 20 (the ON/OFF switch is on the right side). If you were lucky, the screen looked like the photo of the initial screen display (Figure 2.6). That is, it told you that you are working with CBM BASIC V2 which

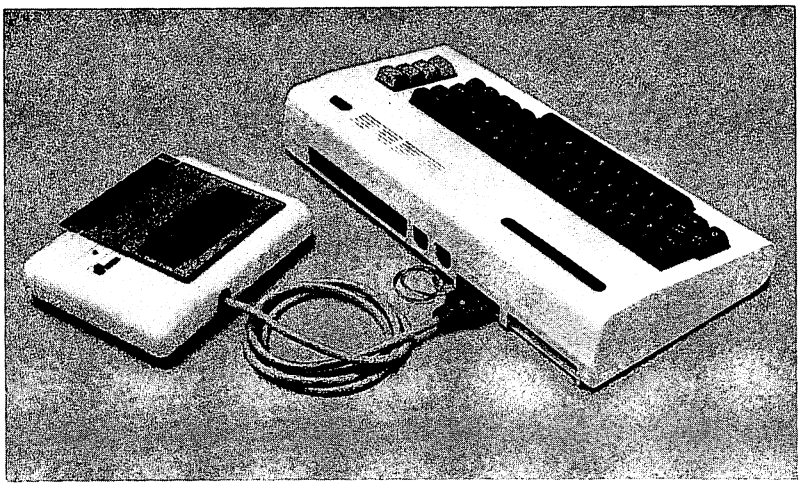


Figure 2.5 The C2N Cassette recorder plugged into the VIC 20.

translates to "Commodore Business Machines BASIC, version 2.0." A second line will indicate "3,583 BYTES FREE" (or more if extra Random Access Memory is installed) which means you have room in the computer's memory to type in 3,583 characters. A third line tells you the computer is "READY."

If you don't have a display on the screen move the switch on the RF modulator to its other position. You should get the proper picture. If the picture is wavy or fuzzy, use the fine-tuning controls and the brightness/ contrast controls to get as clear a resolution as possible.

If you still have problems, turn the computer off and check all your connections. Are they correct? If all else fails, try another TV. When you get a display that looks like the photo of an "initial video display," you are ready to begin computing.

Connecting the VIC 20 to a Video Monitor

(Skip this section if you are using a standard television.)

The signals that come from the video port on the VIC 20 are converted by the RF modulator into a signal like that produced by a television station operating on either Channel 2 or 3. If you use the VIC 20 with a video

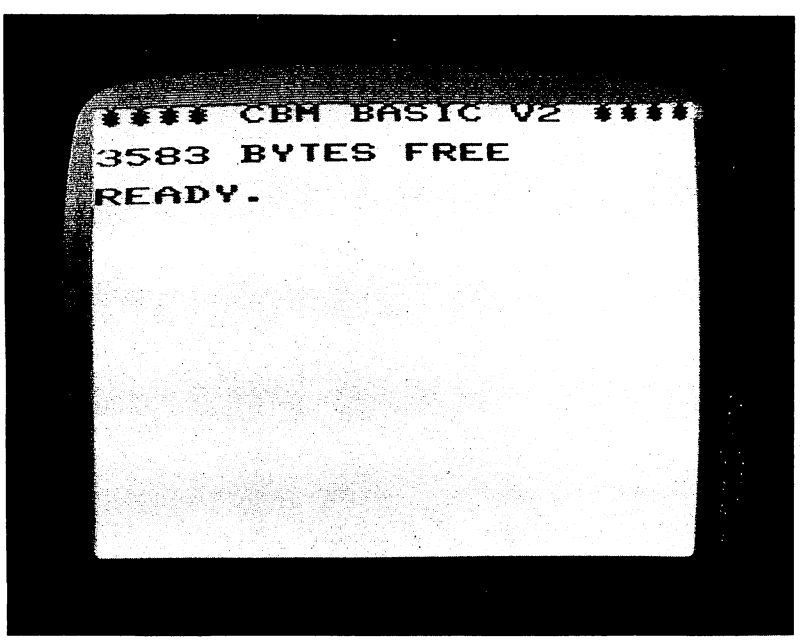


Figure 2.6 The initial screen display.

monitor instead of a television, it will be necessary to buy or build a cable that lets you connect the Video Port to the video monitor. You will need a cable with a 5 pin DIN plug on one end and two connectors on the other end that plug into the video and sound inputs for your video monitor. The monitor end of the cable will vary depending on the type of connectors your monitor uses. There is a video signal compatible with monitors that expect a 75 ohm input (low) and another for monitors with a 300 ohm video input (high). The manual that came with the monitor should tell you which signal to use (the great majority use 75 ohm video input). If you need help wiring this cable, many computer stores and most shops that repair televisions should be able to do it for less than $20.

THE ESSENTIAL VIC 20

Before we begin using the computer, it might be useful to take a quick tour around the VIC 20. It is a small but very powerful machine. This computer, like most small computers, can be viewed as a system which is made up of several basic components. Figure 2.8 is a block diagram of the VIC 20 computer. Everything outside the dotted line, except the keyboard, is an option you must supply yourself.

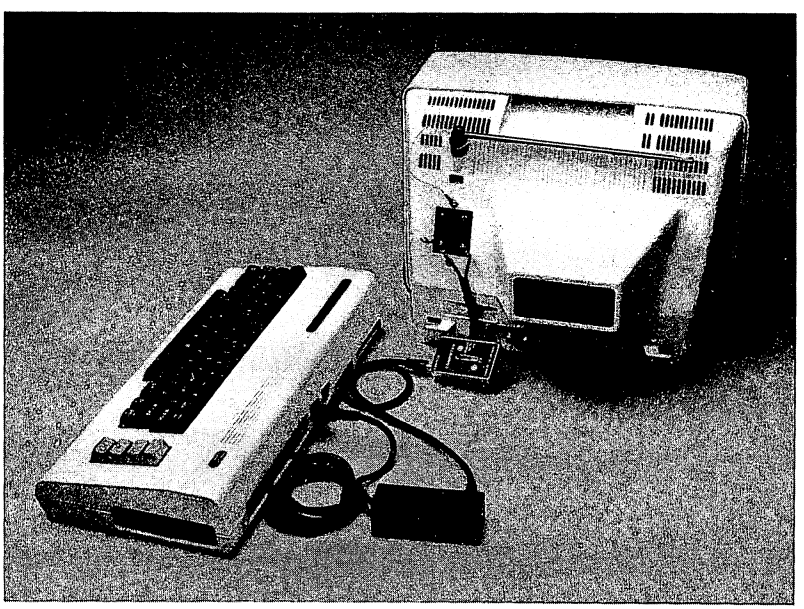


Figure 2.7 The cable connections for a video monitor.

The Power Supply

The VIC 20 runs on 5-volt, direct current power. The black power supply unit takes the 110 volts, alternating current, from your wall outlet and converts it to 9 volts. When you plug the power supply unit into the computer, a voltage regulator inside it changes the 9 volts to 5 volts DC which is then routed to all the circuits on the board that need it. It is a simple power supply, but it does the job.

I/O Ports

I/O is an abbreviation for input/output. If a computer is to be of any use to you, it must be able to receive information and communicate its response back to you. This basic function is called Input/Output. The places on the computer circuit board where I/O occurs are often called "ports." The VIC 20 has several ports that allow you to communicate with one another. The keyboard is your primary means of inputting information and instructions to the computer. It is attached to the main circuit board by a connector at the top left of the board. When the VIC 20 talks back to you, it is usually via the television. It converts its messages to a video (and audio) signal that exits the computer via the 5 pin DIN plug on the back. The VIC 20 has a very good keyboard, particularly for its price. The video display, however, has both positive and negative points. The VIC 20 can create very good color graphics but it is limited to 22 lines of 23 characters. A display of 22 lines by 23 characters is rather sparce even in an inexpensive computer. This is the VIC 20's most serious fault, but one that can be corrected (see Chapter 7).

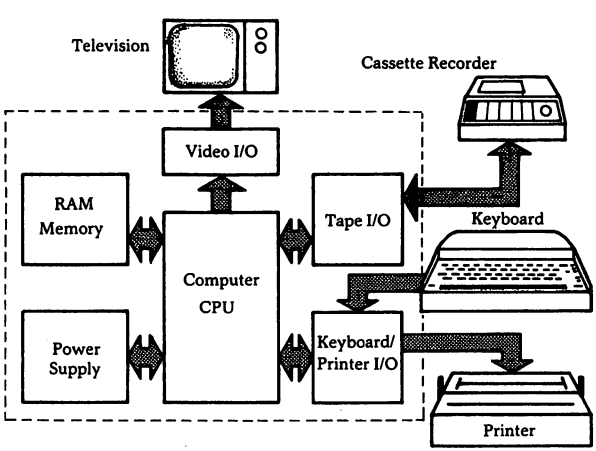


Figure 2.8 A block diagram of the VIC 20.

In addition to I/O ports for the keyboard and monitor, the VIC 20 has provisions for connecting a tape recorder (back panel 6-trace edge connector), a disk drive and/or printer (6 pin DIN plug on back), and a joystick or light pen (9 pin "D" connector on right side).

The cassette (or disk drive) I/O circuits allow you to store programs on a standard audio cassette tape (or "floppy diskette") and then "load" the program back into the computer when you want to use it again.

A "program" is a set of instructions the computer follows to accomplish a particular task (e.g., balance your checkbook or play a game of chess). The computer reads the instructions in a program and does what the instructions tell it to do. As you might expect, the instructions are not given orally. The VIC 20 will not understand spoken instructions. The instructions are stored on the cassette in a code that is a series of tones. If you listen to a tape, all you hear is a buzzing sound. That buzzing is music (instructions, actually) to the ear of the computer. If you use diskettes to store program instructions, they are stored as a series of electrical signals on the magnetic surface of the diskette.

There are two more I/O ports on the VIC 20 computer. One is the Expansion Port which is on the back. This port lets you insert cartridges that work just like cartridges on video games. These cartridges contain additional memory. Some cartridges contain a type of memory that can be used to store programs written by users. Other cartridges have "preprogrammed" memory that contains instructions for the computer. Several VIC 20 video games, for example, are available in cartridge form. To use one, you plug it into the Expansion Port on the VIC 20 and turn on the machine. The instructions in the memory of the cartridge then control how the VIC 20 operates.

Finally, VIC 20 has a "user port" which is on the back next to the cassette port. The user port is called an "expansion buss" or "expansion port" on some computers. It is a convenient way to connect several types of accessories to the computer.

Memory

When you type something on the keyboard or load a program into the computer from a cassette, there must be somewhere to put that information. Each letter or number you type on the keyboard is converted to a code and stored in the memory of the computer. Each character you type has its own code, a series of ones and zeros.

All computers convert characters into codes of ones and zeros. (Ones and zeros, however, are really electrical signals with a zero voltage signifying a zero and a positive voltage—often 5 volts—signifying a one.) The code

that is virtually universal today is the American Standard Code for Information Interchange (ASCII). Most small computers use this code for translating characters and symbols into signals that can be stored in their memory.

Each one and zero is called a "bit." Seven of those bits are used to define the code for each letter or character. Another bit, the eighth, is usually added to the code for each character so the computer can check for errors. Those eight bits make up a "byte." Bytes are the fundamental code units for the VIC 20 and for most small computers. Memory inside the computer is also divided into bytes. One byte of memory can hold the electrical impulses that represent eight ones and zeros.

The VIC 20 computer contains two types of memory, RAM and ROM. ROM stands for Read Only Memory. This type of memory is generally programmed at the factory. The contents of ROM cannot be changed by the user. The VIC 20 ROM contains thousands of instructions for the computer. If there were no preprogrammed ROM in the computer, you might have to provide instructions in patterns of ones and zeros, a fate we would not wish on anyone. Fortunately, when you plug the computer in, it automatically begins to follow the instructions in its ROM. Instead of ones and zeros, you can use English-like words of the computer language BASIC because the Commodore programmers wrote instructions to understand BASIC.

All computer memory cannot be ROM, however. Much of the memory in the VIC 20 is RAM, or Random Access Memory. The standard VIC 20 has a little over 5,000 bytes of RAM. Since each byte can store the code for one character, the 5,000 bytes of RAM can hold up to 5,000 characters. The VIC 20 uses almost 1,500 bytes of RAM for its own housekeeping work. This means you have around 3,583 bytes of memory available for use. You can add extra RAM memory (and ROM) by plugging in optional cartridges. The VIC 20 can use a maximum of slightly over 32,000 bytes of RAM.

RAM is also known as "volatile" memory. It is general-purpose memory. You can store data or instructions in RAM, tell the computer to use the information you've stored there, and then replace the material in RAM with something new. You can put data in RAM (write to memory) and you can take it out (read from memory). You can only read ROM. The biggest problem with RAM is the fact that whatever is in it disappears when the computer is turned off. If you need to save something in RAM for use later, it is necessary to store it on a cassette (or diskette) before turning off the computer. Material in ROM, on the other hand, remains there forever but cannot be changed or modified.

You will see ads for "3K RAM" or "8K RAM," and "16K RAM" for the VIC 20. Memory is often referred to in terms of "K." Each K of memory is 1,024 bytes. Thus 16K would be 1,024 times 16 or 16,384 bytes. Just multiply the number of K by 1,024 to determine the number of bytes of memory.

The CPU

The CPU or central processing unit is the heart of a computer system. Although most CPU chips are smaller than an Oreo cookie, the electronic components they contain would have filled a room a few decades ago. Using advanced microelectronic techniques, manufacturers can cram thousands of circuits into tiny silicon chips that not only work dependably but also use less power than an electric razor. There are several popular CPU chips today with names like Z80, 6502, 8088, 68000. There are real differences among the cips mentioned above, but the differences are mainly of interest to computer designers, experienced programmers; if you need the special capabilities of some of the chips (e.g., the ability to use large amounts of memory or work very rapidly), the different chip numbers may be important to you.

Regardless of the chip, the CPU does most of the actual processing inside the computer. It also sends signals to the other parts of the computer that synchronize the operation of each component. This synchronization permits the system to perform thousands of operations a second. These operations are fairly simple (e.g., adding two numbers together) but the speed at which they are done allows the computer to do some very complex things by breaking complicated jobs down into hundreds of simple steps.

The VIC 20 uses the 6502 microprocessor chip which is manufactured by MOS Technology, a company Commodore owns. This "computer on a chip" is the same one used in the Apple II, PET, and ATARI 800XL/1400XL/1450XLD computers. A modified version of the 6502, called the 6510, is used in the Commdore Model 64 computer. In the VIC 20 computer, the 6502 chip has some help from another chip, the VIC 6560 chip from which the computer derives its name. VIC is short for "Video Interface Chip." The VIC 6560 chip is a specially designed integrated circuit that does most of the work required to create and produce the computer's color video display and sound. Between them, the 6502 and the VIC 6560 chip do most of the actual computing work. The rest of the circuits inside the VIC 20 play supporting roles.

Chapter 3

Up And Running — VIC 20

At this point, you should have all your cables correctly connected. The computer and television should be plugged in. If they are not on, switch on the VIC 20 and the television. The VIC 20 on/off switch is on the right side.

If everything is operating correctly, you should now have the initial message described in Chapter 2 on the screen. If it is not on the screen return to Chapter 2 for instructions and suggestions. At this point we assume you have the initial display on your screen.

The Keyboard/Display Connection

The keyboard on the VIC 20 uses the QWERTY layout found on standard typewriters. That is, the top row of letters begins with Q on the left and has WERTYUIOP across. Figure 3.1 is an illustration of the VIC 20 keyboard. If you touch type, the location of letters and numbers will be familiar to you.

The VIC 20 keyboard is more than a copy of a typewriter keyboard, however. It is, in fact, a very busy keyboard. Look at the A key, for example. Like most of the keys it has a letter printed in the middle of the keypad. There are also two graphic symbols on the front of the A key. This one key can produce four different symbols on the screen — a lowercase a, a capital A, and two graphic symbols.

The VIC 20 keyboard also contains a number of keypads with symbols that may be unfamiliar to you. There are keys with arrows on them, keys with words or phrases such as RUN/STOP, CLR/HOME, INST/DEL, and RESTORE, and there are keys with symbols such as the Commodore logo (bottom left) and arrows (top left and bottom right). Finally there are four keys on the right labelled F1 through F8. You will learn the functions of all the VIC 20 keys in the next few chapters.

Figure 3.1 The VIC 20 keyboard.

YOUR FIRST PROGRAM

Let's try a little work with the keyboard. With everything plugged in and turned on, type in the following after reading the Note just below the material to be typed:

```
10 INPUT N$
```

Note: The VIC 20 normally produces capital letters without holding down the shift key. The line above can be typed correctly without using the shift key until you need the $ symbol.

Now press the RETURN key. This key is used to tell the computer you are finished typing material on a particular line. The VIC 20 will move the blue square, which is called a cursor, down one line. It should be flashing now over on the left side of the screen. The VIC 20 is now waiting for you to type more information.

If You Make a Mistake

Unless you are a much better typist than we are, you will make some errors as you type this first program. There are several things you can do to correct errors.

Before Pressing Return. If you spot an error before pressing the RETURN key, you can use the RIGHT ARROW and LEFT ARROW keys to move the cursor around on the line you are writing. The cursor is the blue square that tells you where you are on the screen. The RIGHT ARROW and LEFT ARROW keys are on the bottom right of the keyboard. Note that one key has arrows that point to the right and to the left. If you press this key, the cursor will move to the right. If you hold down the SHIFT key and press the same key, the cursor will move to the left. Use the SHIFT and Right/Left Arrow key to move the cursor to the point in your line where the error occurred. For example, if you typed 10 INPUR N and then noticed that you pressed the R key instead of the T, it is an easy error to correct. If the cursor is just past the N, hold down the SHIFT key and then press the Right/Left Arrow key until the cursor is in the space just to the right of the offending R. Now find the INST/DEL key which is at the top right of the keyboard. Press this key once. DEL is short for delete. Pressing it with the cursor to the right of the R should cause the R to disappear. Now hold down the SHIFT key and press the INST/DEL key again. INST is short for INSERT. When you deleted the R the computer moved material on the right over one space. Pressing the INST key once (actually SHIFT INST/DEL) told the computer to make room for a character. Now type a T, use the Right/Left Arrow key to move the cursor to just past N and type a $.

You can use the Right/Left Arrow key and the INST/DEL key to correct many errors. It will take a little practice to get used to how these keys work. If you hold down the Right/Left Arrow key, the cursor will continue moving in the direction of the arrow. If it gets to the edge of a line the cursor will wrap around to the next line. Try it and see how the cursor travels.

After Pressing RETURN. Often you will not notice an error until you have already pressed the RETURN key. Again there are several things you can do.

Retype the Line. Suppose you typed in 10 IBPUT N$ and pressed the RETURN key before noticing the error. If you type in 10 INPUT N$ and press the RETURN key again, the new version of line 10 will replace the old version in the memory of the computer. This is often the simplest and easiest approach to correcting an error. If you accidentally type in a line that shouldn't be there at all, just type the number of the line and press RETURN. That will cause the entire line to be deleted from the program you are writing (but not from the screen). For example, if you accidentally put a line 99 into the program, typing 99 and pressing RETURN will delete the line. There is another process called "editing" that we will deal with later. You can correct any mistakes you make, however, with the methods described above.

(**Note:** Press the RETURN key when you finish typing each line and remember that you get capital letters without pressing the shift key.) Now type in the following:

```
 20 PRINT N$;" CAN YOU WRITE"
 25 PRINT "COMPUTER PROGRAMS?"
 30 INPUT A$
 40 IF$ = "YES"THEN GOTO 100
 50 IF A$ = "NO" THEN GOTO 110
100 PRINT "GREAT, WE'LL HAVE FUN."
105 GOTO 200
110 PRINT "NOT TO WORRY, YOU WILL LEARN"
200 END
  5 PRINT "HI WHAT IS YOUR NAME?"
```

Use the Right/Left Arrow key and the INST/DEL key to correct any errors you make while typing the lines above. Because the VIC 20 has a limited display capacity (no more than 23 characters on each line), your program will not look as neat on the screen as it does here. Line 20, for example, is split into two lines on the screen with 20 **PRINT** N$" CAN YOU W" on one line and "RITE" on another. Unless you add accessories that expand the video display capacity of the VIC 20 to 40 or 80 characters per line, you simply will have to live with this limitation. A "program" line can be as long as 88 characters and may fill several lines on the screen.

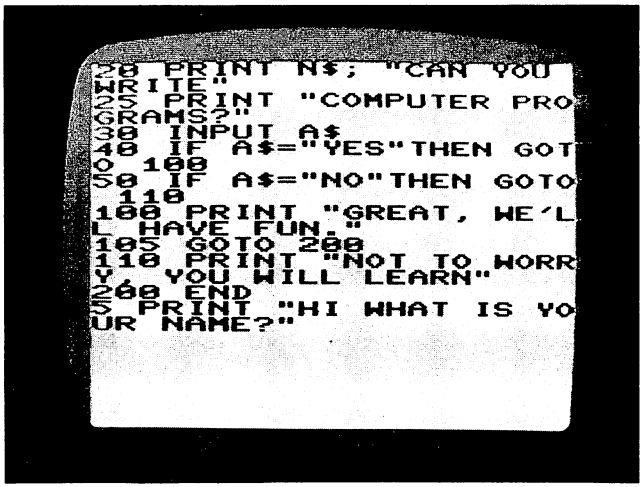


Figure 3.2 The screen display after the program is typed in.

YOU HAVE A PROGRAM!

At this point you have your first program in the computer. Now hold down the SHIFT key and press the HOME/CLEAR key. That will CLEAR the screen. You should have a completely blank screen with a flashing cursor in the upper lefthand corner. The key word **CLEAR** (SHIFT HOME/CLEAR) tells the computer you want the material on the screen erased and the cursor moved to the HOME position (top left corner). If you press CLEAR/HOME without holding down the SHIFT key the cursor will move to the home position but the screen will not be erased.

Now type the word LIST and press RETURN. Your program should be listed on the screen in numerical order. Even though you typed line 5 after all the other lines, it is placed first on the screen since it has the smallest line number. Lines 5, 20, 25, 40, 50, 100, and 110 are too long to be displayed on one line on your screen.

The material you have just typed is a computer program which was written in BASIC, a very popular computer language. The VIC 20 uses a relatively powerful dialect of BASIC which has much in common with the versions of BASIC used in other personal computers such as the Apple II, the ATARI computers, and TRS-80 Color Computer. The VIC 20's version of BASIC is very close to the BASIC used on other Commodore computers. If you learn BASIC on the VIC 20 it will not be difficult to learn the version used on other popular computers should you move on to another model.

A program is a set of instructions you want the computer to follow. The instructions are organized into program lines, and each line begins with a line number. When the computer is told to follow or "execute" the instructions in a program, it begins with the instructions in the line with the smallest number. It executes the instructions in that line first; then it moves onto the line with the next smallest number and executes the instructions there. It thus moves sequentially through the program, from lowest line number to highest line number. If you do not want the computer to follow this standard or "default" pattern, it is necessary to tell the computer what line you want it to do next. If you don't, it will move sequentially through the program.

How do you get the computer to execute your program? It is really very simple. The VIC 20 BASIC has a keyword, **RUN**, that tells the computer to follow the instructions in a program. To run your first program hold down the SHIFT key and press the CLR/HOME key. That will clear the screen and put the cursor in the home position. Now type RUN and press RETURN. The sentence HI WHAT IS YOUR NAME? appears at the top of the screen. On the line below the question a ? appears along with the cursor. Type in your first name and press RETURN. If you type in JERRY

and press RETURN, the computer will reply, JERRY CAN YOU WRITE COMPUTER PROGRAMS? Another ? indicates the computer is waiting for you to answer the question. Type in either YES or NO and press RETURN. Let's say you type in NO. The computer should respond with NOT TO WORRY, YOU WILL LEARN.

You have now typed in a program and it has been RUN or executed by the computer. If it didn't work as expected, look back over the lines you typed in and try to find the problem. For example, if the number 0 was typed for the letter O in line 50, the program won't work correctly when you type NO. When the program runs correctly, read the explanation of each line in the section that follows.

A Line by Line Explanation of the Program

Chapter 6 discusses some of the popular languages available for the VIC 20. This section explains what happens as each line of your first BASIC program is executed.

5 **PRINT** "HI, WHAT IS YOUR NAME?

This line contains the key word **PRINT** and the sentence HI, WHAT IS YOUR NAME? The key word **PRINT** instructs the computer to print whatever appears on the line just after **PRINT**. The quotation marks before and after the sentence tell the computer to print the material just as it appears in the program.

10 **INPUT** N$

The key word **INPUT** tells the computer to look for something to be typed on the keyboard. If you type in AMY and press RETURN the computer sets aside a section of its memory and puts the characters AMY there. It then makes N$ equal to AMY. N$ is a "variable name" and AMY is the content of that variable. Later in the program if you tell the computer to PRINT N$, the computer will print AMY rather than N$. On the other hand, if you tell the computer to PRINT "N$" the computer would actually print N$ because you enclosed it in quotation marks.

20 **PRINT** N$" CAN YOU WRITE"
25 **PRINT** "COMPUTER PROGRAMS?"

The key word in both lines is **PRINT**. Since N$ is not enclosed in quotation marks the computer checks to see what N$ stands for. If it is AMY, that is what is printed on the screen. The rest of the phrase in line 20 is between quotation marks so it will be printed as is. All of the material after PRINT in line 25 is within quotation marks so the sentence AMY

CAN YOU WRITE COMPUTER PROGRAMS? will appear on the screen with AMY CAN YOU WRITE appearing on one line and COMPUTER PROGRAMS? on the next.

30 **INPUT** A$

This line is just like line 10 except whatever you type in will be assigned to the variable name A$ rather than N$. We chose the letter A because this is an answer to a question. The N in N$ is short for name.

40 **IF** A$ = "YES" **THEN GOTO** 100

There are three key words in this line: **IF**, **THEN**, and **GOTO**. The line tells the computer to check what A$ stands for. If A$ equals YES then the instruction after THEN is executed. In line 40 the key word **GOTO** instructs the computer to skip to line 100 and begin executing the instructions there **IF** and only if A$ equals YES.

50 **IF** A$ = "NO" **THEN GOTO** 110

This line, like line 40, tells the computer to check what A$ equals. If A$ equals NO, the computer jumps to line 110 for instructions. If A$ equals YES, the computer never reads or executes the instructions in line 50 since it jumps from line 40 to line 100. If A$ equals NO, the computer never reads or executes the instructions in lines 100 and 105 since it jumps from 50 to 110.

100 **PRINT** "GREAT, WE'LL HAVE FUN."

The key word in line 100 is **PRINT**. The line causes the sentence in quotation marks to be printed on your screen.

105 **GOTO** 200

If A$ equals YES and line 100 prints its cheery comment, the program has accomplished its purpose. You don't want the other comment in line 110 to be printed, so **GOTO** 200 sends the computer from line 105 to line 200.

110 **PRINT** "NOT TO WORRY, YOU WILL LEARN."

If A$ equals NO, the computer jumps from line 50 to line 110 where an encouraging comment is printed on the screen.

200 **END**

The key word **END** tells the computer that, once it gets to line 200, it has finished its work. It stops executing instructions and prints READY which indicates it is waiting for additional instructions. **END** is optional in many

programs, because the computer will also stop and tell you it is READY when it has executed the instructions in the line with the largest line number.

At this point you may be thinking that this example program was too simple, that writing programs for a computer is far more complicated than this. It is true that there are some key words that are more difficult to understand than the ones used so far. And there are complex, sophisticated procedures used by professional programmers that can't be picked up easily in a weekend while keeping one eye on the football games. But, having said all that, it also is true that you can quickly and easily learn to write your own programs in a language like BASIC. The skills you learn are some of the same ones used by programmers who spend most of their waking hours at the keyboard of a computer. In addition, if you learn the key words in BASIC and how programs are written, you will be able to make use of programs you find in magazines on your computer, and you will be able to adapt and personalize programs. There are a number of good BASIC programming books available that will help you expand your programming ability.

Chapter 4

Loading Information From Cassette or Disk

The process of getting information that has been stored on either a cassette or diskette back into the computer's memory is called *loading*. This chapter explains how to **LOAD** programs using cassettes or diskettes. Chapter 5 describes how to **SAVE** programs using a cassette or diskette and how to use the printer.

CASSETTE STORAGE

Making the Connection

In order to use a cassette player with the VIC 20, the cable from the recorder must be connected to the back of the computer. See Chapter 2 if your recorder is not connected to the VIC 20.

Let's look at the recorder and its parts for a minute. Two very helpful features of the Commodore recorder are the memory counter (discussed later in this chapter) and the **SAVE** indicator light (discussed in Chapter 5). The Commodore recorder has built in another valuable feature. Notice that the recorder does not have a volume and tone control. Commodore wisely preset the correct volume and tone levels in its specially designed recorder so that adjustment is not an issue with this machine. Commodore and Atari (which uses the same technique) are the two companies that have reliable cassette systems. Many other computers have reputations for unreliable cassette systems.

When you turn on the TV and the VIC 20, the screen looks like Figure 4.2. The number of bytes may be different depending on the amount of memory you have.

Once the cassette is connected and the screen looks like Figure 4.2, you are ready to **LOAD** programs into the memory of your VIC 20.

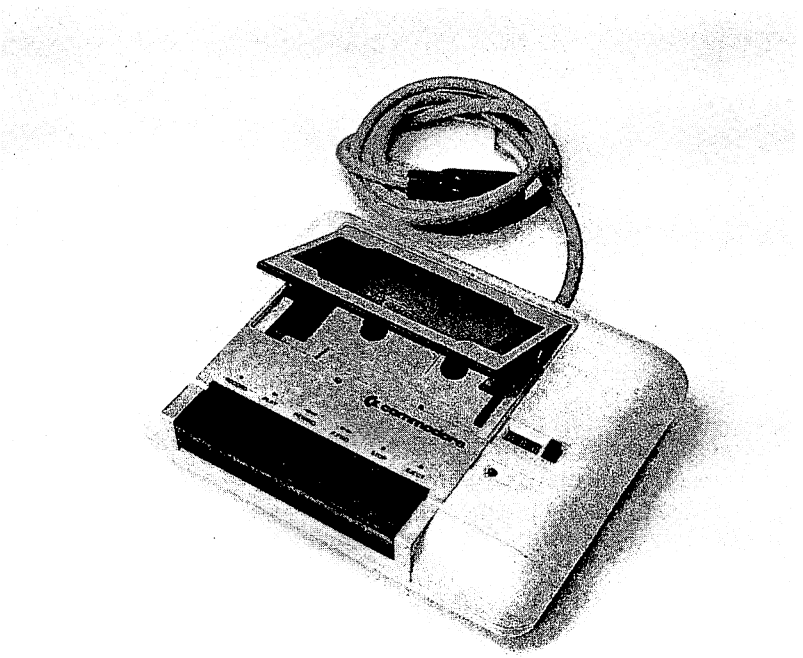


Figure 4.1 The Commodore cassette recorder.

LOADing Steps

The easiest way to **LOAD** a program from cassette uses the SHIFT and RUN/STOP keys on the VIC 20 keyboard. Follow these guidelines to **LOAD** the first program on a tape:

- rewind the tape completely
- hold down the SHIFT key and press the RUN/STOP key

The screen reports:

PRESS PLAY ON TAPE

When the PLAY button on the recorder is pressed, the screen displays:

OK
SEARCHING

and when the first program is located (say it's name is SUM), the message on the screen is:

```
FOUND SUM
LOADING
READY
RUN
```

Using this method to **LOAD** does two things. First, the program is **LOAD**ed into the memory of the computer. Second, the program is automatically **RUN** or executed. In the instructions that are explained next, it is necessary to type **RUN** to execute the program. There may be times when you don't want the program to execute automatically (you might want to get a list of the program line by line for example), but most of the time SHIFT and RUN/STOP will be a quick and easy way to get a program up and running.

Suppose you have a cassette that contains several programs. The first program, called TEST, is the one you want to transfer from tape into the

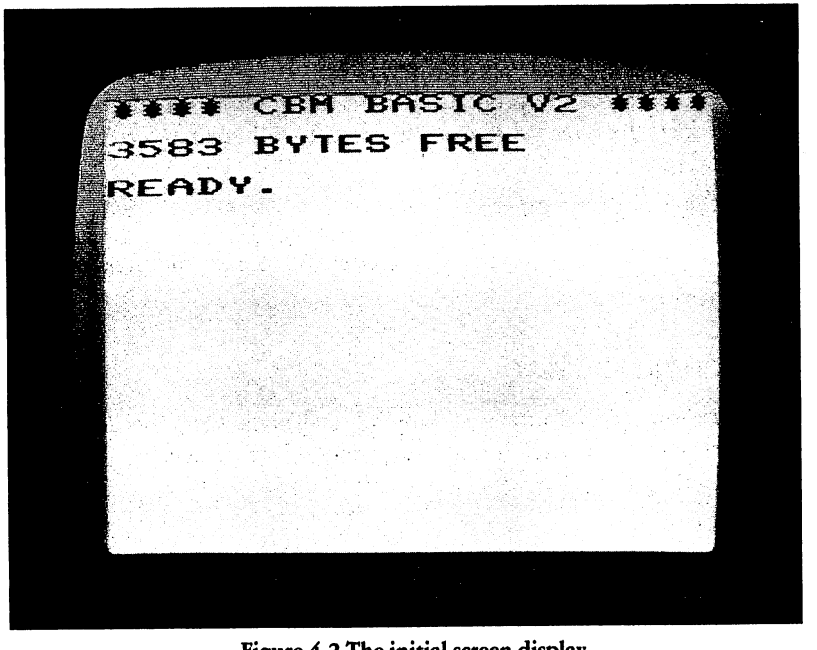


Figure 4.2 The initial screen display.

computer. The following steps show you how to **LOAD** the program into memory:

1. Rewind the tape in the recorder to the beginning of the tape.

2. Type

   LOAD "TEST"

and press the RETURN key on the keyboard. Be sure to type quotation marks (" ") before and after the name of the program. If these are left out, the following message appears:

```
?TYPE MISMATCH
ERROR
READY
```

To correct this mistake, retype **LOAD**, the quotation mark, name of the program, and the last quotation mark. Then press RETURN.

Also, it is important to spell the name of the program correctly. When the computer reads information on the tape, it looks specifically for the name of the program you typed. For example, if TRST is typed instead of TEST, the VIC 20 will search the entire tape looking for a program named TRST that doesn't exist.

If you discover an error before pressing ENTER, use the DELETE key to backup and type the correction.

After the name of the program is correctly entered and you press the RETURN key, the screen displays a message to tell you what to do next. That message is:

```
PRESS PLAY ON TAPE
```

3. Press the PLAY button on the recorder. As soon as you press PLAY, the screen displays:

```
OK
SEARCHING FOR TEST
```

to let you know the computer is looking for the program you specified. The VIC 20 lets you know when it finds the program by displaying:

```
FOUND TEST
LOADING
READY
```

This means the program has been read, or transferred, from cassette into the computer's memory. The READY message appears when the entire

program is in memory, so it may be a few seconds before it appears.

4. To execute the program, type

RUN

press the RETURN key, and that's all there is to it! The program is ready for you to use.

As with just about anything, there are a couple of other ways to type instructions to **LOAD** programs. The instructions above are probably the safest, but what if you forget the name of the program you want to **LOAD**? Can you LOAD a program without knowing the exact name? Yes, you can. If the program you want is the first one on the tape, the format is as follows:

LOAD "

(notice only one set of quotation marks) or:

LOAD

with no quotation marks. This only works, though, if the program you are looking for is first on the tape. The instructions tell the computer to **LOAD** the first program it finds. You could go through this procedure several times to locate your program, but it gets to be very time-consuming.

What if you don't remember the name of the program or it isn't first? Suppose, for example, that you have a cassette with the programs COLOR, DEMO, TEST, and SAMPLE on it and TEST is the one you want to **LOAD**. There are two ways to deal with this. If you know the name of the program, you can follow steps 1-3 above. As the computer reads the tape and finds the first program, it lets you know what is happening by displaying:

SEARCHING FOR TEST
FOUND COLOR

As the tape progresses and the computer locates the next program, the display is:

FOUND DEMO

When TEST is found, it is **LOAD**ed and you can go to step 4 (type RUN) to use the program.

That method is fine if COLOR and DEMO are short programs. Some programs, however, can take five minutes or longer to be read. If you are sitting at the keyboard waiting for TEST to be located, that can be a long time. For lengthy programs, you may want to use another method.

Using steps 1 and 2 above, you will need to go through the entire tape at least once working with the counter on the recorder.

The counter allows you to locate where each program begins and ends on the tape. For instance, the program COLOR might begin at location 009. When the screen display reports that COLOR has been found, write down the numbers on the memory counter (e.g., COLOR-009). Each program on the tape should be found and a note made of the ending location. Use the following guidelines for noting tape locations.

- rewind tape completely
- press the counter button so that there are all zeros in the display
- in order to locate all the programs on the tape, tell the computer to **LOAD** the last program (that way the VIC 20 searches the entire tape, telling you when each program is found), or you could instruct it to search for a program that doesn't exist and the computer searches the entire cassette for that nonexistent program.
- when the screen display indicates a program has been found, make a note of the program name and its location.

When all of the programs have been found and you have a list of where each one begins and ends, you have a valuable tool for **LOAD**ing programs later. For example, say TEST ends at counter location 57 and SAMPLE begins at 60. You can rewind the tape to the beginning and set the counter to 0. Next, use the fast forward key to advance the tape and counter to about 58. Now you are ready to **LOAD** SAMPLE using steps 2-4 above.

DISKETTE STORAGE

Making the Connection

Just as the cassette player must be connected to the VIC 20, so must the disk drive. The cable from the disk drive is connected by a 6 pin socket immediatley to the left of the cassette connector. The VIC 20 printer plugs into the computer there or it can be connected to the disk drive.

The power up sequence, or order in which the computer and disk drive are turned on, is very important with the VIC 20. When the drive is connected, turn it on first. Always make sure the computer is turned on *after* the disk drive.

The first part of this section assumes that if you have a disk drive it is the model 1541, and that you have at least the demo diskette that comes with the drive. Figure 4.3 illustrates the diskette and its various parts as well as the correct way to insert a diskette into a disk drive.

The material you see through the oval window of the diskette is where information is stored. Treat diskettes with care and *never* touch the plastic platter that is visible in the oval window. The oil from your finger can prevent the computer from reading information stored there. Diskettes are sensitive and don't respond well to dust, cigarette smoke, dirt, chocolate chip cookie crumbs, or strawberry milkshakes. Because the platter inside the protective envelope is covered with a magnetic film, diskettes also are highly sensitive to magnetic fields. It is a good idea to try to keep them from the tops of television sets and disk drives because they contain transformers that generate a magnetic field.

The notch on the right side of the diskette is called the "write protect" notch. If this notch is as it appears in the picture, the diskette is not write protected. That is, the computer can store or change information on it as well as read the information already stored there. If you have a diskette that contains a valuable program and you don't want anyone to change it, you can place a piece of tape (most diskettes come with tabs for this purpose) over the notch. It's enough here to say that information cannot be put on a diskette while the write protect notch is in place. Chapter 5 discusses this further.

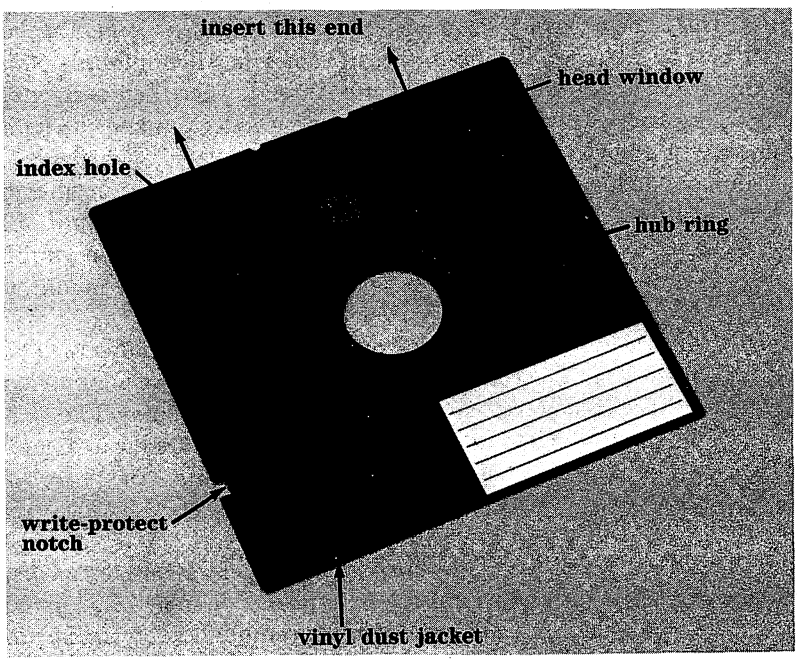


Figure 4.3 A floppy diskette.

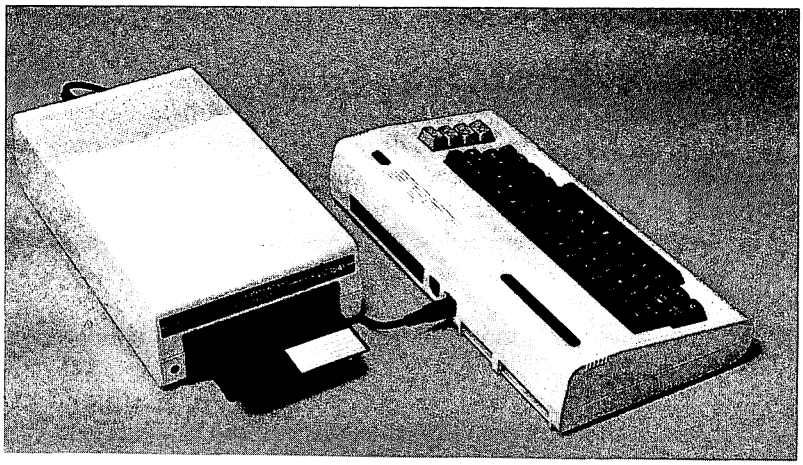


Figure 4.4 The correct method for inserting a diskette into the VIC 20.

When inserting a diskette into the drive, always put it in window-edge first (see Figure 4.4). This means the label is always on top. Figure 4.4 also labels the parts of the disk drive. Notice the green light on the left of the drive. This is the power light and it is lit when the disk drive is on. The red light next to it has two functions: first, it comes on when information is being written to or read from the diskette. Second, when there is an error (e.g., program not found), it flashes off and on to let you know there is some sort of problem.

The Commodore 1541 disk drive door can be a hassle, but generally, to open the drive door press the lip gently in and up with your thumb. It's easiest to place the thumb on the lip with the rest of the fingers on the top of the drive and to press up with the thumb. Slide the diskette in until it seats. To close the door, press down from the same position.

The 1541 disk drive is a versatile unit and can be used with either the VIC 20 or the Commodore 64 computers. The two systems receive information at different speeds, however, and the 1541 assumes the Commodore 64 is the computer that is being used. For that reason, it is necessary to tell the drive that it is communicating with the VIC 20. To do this type:

OPEN 15,8,15,"UI-": **CLOSE** 15

and press the RETURN key. The ominous **OPEN** and **CLOSE** statements, as well as the accompanying numbers, are explained in more detail in the next chapter. Briefly, the statement **OPEN**s a communication channel to the disk drive with a user command and then **CLOSE**s the channel. It is

possible to **LOAD** some programs without setting the 1541 to the VIC 20's speed. But it's probably better to take the extra minute to give the instruction at the beginning of each computing session. It is only necessary to give the instruction once when you turn the computer on. If you have a 1540 disk drive the instruction is not required.

LOADing a Program

The process of **LOAD**ing programs from diskette is somewhat like using a cassette, only much faster! The VIC 20 assumes you are using a cassette. For example, even with the drive connected to the VIC 20, if you type **LOAD** "TEST" the VIC 20 responds with the PRESS PLAY ON TAPE message because it expects a cassette to be used. It is necessary to tell the computer to expect information from the serial port where the drive is connected. Anything that is attached to the VIC 20 computer is called a "device" and has a "device number." The cassette player, for example, has a device number of 1. The disk drive device number is 8 and the printer is 4. Each time you want to get information from a device, you must include the number of the device.

But you didn't do that with the cassette you say? Right. The VIC 20 has a default value of 1; that is, it assumes you are using a cassette and doesn't require a device number with the recorder. It is necessary, however, to include device numbers when using disk drives and printers (more on printers in the next chapter).

LOADing Instructions. There are really only 2 steps for **LOAD**ing programs from diskette. Suppose you have inserted a disk that contains the program you want and have closed the disk drive door securely. To **LOAD** the program you want (say it's name is SAMPLE):

1. Type:

   LOAD "SAMPLE",8

and press RETURN. The screen displays

```
SEARCHING FOR SAMPLE
LOADING
READY
```

and although the message is the same as for the cassette, the disk performs much quicker than the cassette. Notice that the comma comes *after* the quotation mark and is followed by the device number of the disk drive (8).

What happens if the comma is placed inside the quotation marks? The display reads:

```
? SYNTAX
ERROR
READY
```

What if a number other than 8 is entered for the device? The following message appears:

```
SEARCHING FOR TEST
? DEVICE NOT PRESENT
ERROR
READY
```

Because the VIC 20 usually tells you what kind of mistake was made and when it is ready for you to take control, all you have to do is retype the instruction. But what happens if the name is misspelled or a file is requested that isn't on the diskette? Then this message appears:

```
? FILE NOT FOUND
ERROR
READY
```

At this point, the red light on the disk drive blinks on and off. This happens any time there is an error related to the drive. It is a handy little feature to let you know it has happened (or isn't happening) in this case.

2. After the program has been successfully LOADed into the computer type:

```
RUN
```

and press RETURN. The program executes. When you are finished with a program and want to empty the VIC 20's memory, type:

```
NEW
```

and press RETURN. Now you can **LOAD** and RUN another program. If you get the message:

```
? OUT OF MEMORY
```

there is not enough room to **LOAD** and/or RUN the program you requested. If this happens, you can type NEW and clear the memory of any programs already there. In fact, we suggest that you always clear the memory between programs to keep programs from accidentally overlapping. Try **LOAD**ing the program again. If you still get the message that

there isn't enough memory, then the program requires more memory than your VIC 20 has. It is always a good idea to make sure the programs you purchased do not require more memory than you have *before* buying the program.

Some of the features of the disk drive that make it preferable to the cassette are evident when it comes to **LOAD**ing programs. For example, it is necessary to know where on tape a program is located to speed up the **LOAD**ing process when using a cassette. On a disk, however, you don't have to know where on the disk a program is—the computer takes care of that for you. Another advantage of a disk drive is the ability to get a list of the programs on a diskette. This list, usually called a directory, lets you know what programs are on a diskette and approximately how much room each one will take up in the memory of the computer. Suppose you have a diskette that hasn't been used in a while and you don't remember what's on it. To find out, type:

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

and press RETURN. This doesn't mean to **LOAD** money into the computer. The dollar sign means "directory" to the computer. The instruction then means to **LOAD** the directory from device number 8 (the disk drive). When you press RETURN, the screen displays:

```
SEARCHING FOR $
LOADING
READY
```

The VIC 20 disk drive sets aside space on each diskette for the directory. When the screen indicates the computer is ready, the VIC 20 has read the directory into its memory. In order to see the list of program names, type:

```
LIST
```

and press RETURN. Quicky the names roll by on the screen. Sometimes the program names go by so quickly they can't be read. The speed can't be changed, but you can control how many flash by. If you want to interrupt the **LIST**ing, press the RUN/STOP key. Pressing the RUN/STOP key causes this message to be displayed at the bottom of the screen:

```
BREAK
READY
```

When you're ready to see the rest of the directory, type **LIST** and press RETURN again. The directory starts over from the beginning. You can do this as many times as you want.

Take just a minute to look at your directory. While the numbers and names might be different, suppose your directory looks something like this:

```
5      "TINYMON1 FOR VIC"
  PRG
8      "TINYMON INST"
  PRG
7      "PROGRAMBLE CHAR"
  PRG
9      "VIC CHAR GENR"
```

On the left side of a line is a number. This number represents the amount of "blocks" the program uses. A block is 254 bytes. The first program, then, is five blocks or 1,270 (5*254) bytes long. If a program is 255 bytes long it takes two blocks, so the block concept is approximately how much room a program needs.

The first file, again, is about 1,270 bytes long. You also can tell that this file is a program by the designation PRG. If it were a data file, the letters SEQ would have followed the block numbers. Data files are found most often when you have written a program that requires data. The last piece of information about the file is its name—TINYMON1 FOR VIC. This is the title you use to **LOAD** the program. If you forget the name or spelling of a program, the information within quotation marks in the directory is what you need to know.

Now that you have **LOAD**ed a few programs, you are ready to **SAVE** the programs you create. Chapter 5 explains how to **SAVE** programs for future use. The first section of the chapter deals with cassette storage while the second section explains how to **SAVE** your programs on diskette. The last section explains how to use the printer for storage.

Chapter 5

Output to Cassette, Diskette or Printer

If you have a program in the memory of the computer and want to store it for later use, in the language of computers, what you want to do is **SAVE** the program. That simply means that you want some way of keeping the program intact when the computer is turned off and the memory is erased. There are basically two devices you can use for this purpose: a cassette or a diskette. There is another device that is less often considered a storage medium and that is a printer. The cassette and diskette store information electronically in codes that make sense primarily to the computer. The printer, on the other hand, provides what is called a "hard copy" of the program. A "hard copy" is the program transferred to paper via the printer. If you are using a cassette for storage, all you need to do is insert a good quality tape in the cassette recorder. If, however, you are using a diskette, you need a 5¼-inch, soft-sectored diskette that has been prepared (formatted) before a program can be stored on it. Information on how to format a diskette is in the section on saving with diskettes. This chapter describes first the steps for saving on cassette, then diskette, and finally on a printer.

Suppose you have typed in a program (like the one in Chapter 3, for example) and you'd like to show your skeptical friends the new programming talents you've acquired. But you won't see them for a week or so. Other than leaving the computer on all that time, what can you do? This chapter explains how to solve that problem by saving your program. (If you don't have a program in memory, please type in one. Again, the one in Chapter 3 will do nicely.)

SAVE TO CASSETTE

First of all, make sure the recorder is properly connected to the computer. Next, decide what name you're going to give the program. The

name can be up to 16 characters long (including spaces). It is not absolutely necessary to have a name for the program if you're using a cassette, but it makes **LOAD**ing simpler if you have several programs on one cassette. For the sake of simplicity, let's call the program DEMO.

Make sure the tape is rewound to the beginning, set the memory counter to zero, and follow these steps to **SAVE** the program called DEMO:

1. Type:

   SAVE "DEMO"

and press RETURN. The screen displays:

   PRESS RECORD & PLAY ON TAPE

2. Press RECORD and PLAY and the recorder starts automatically. It is crucial that both RECORD and PLAY be pressed. The computer can't sense if only one of them is pressed. As with anything that is recorded on a tape player, if both RECORD and PLAY aren't used, nothing is recorded (in this case, **SAVE**d). If both are pressed, however, the **SAVE** light on the recorder (just below the memory counter) comes on. The screen displays:

   SAVING DEMO
   READY

When the program is **SAVE**d, the light goes off and the tape stops automatically.

3. Make a note of the program name and where it stops according to the memory counter. For example, if DEMO starts at location 5 and ends at 12, write the beginning and ending numbers on the cassette label. This information is useful in two ways: one, when **LOAD**ing the program later, if the memory counter goes past 12 and the program isn't ready, you'll know there is a problem. Second, when you put another program on the tape you'll know where DEMO ends without **LOAD**ing it. For example, **SAVE** DEMO again but this time call it SAMPLE. Where on the tape should it be **SAVE**d? Well, you could rewind the tape and **LOAD** DEMO again, find where it ends and **SAVE** SAMPLE there. Or, because you know the location on the memory counter, you can rewind the tape, reset the couner to zero and fastforward the tape to about 15 (a safe distance after DEMO ends at 12). This process if faster than **LOAD**ing each time.

Because strange things can happen when programs are **SAVE**d, Commodore included a **VERIFY** command that lets you check the program just **SAVE**d on tape with the program still in memory. Why is this a nice feature? Suppose you have a very important program you have just **SAVE**d

because you'll need it next week when your income tax return is audited. Although you're in a hurry, you decide to **VERIFY** the program anyway. To your horror you discover that the program, indeed, was not **SAVE**d correctly. You get a new tape, **SAVE** and **VERIFY** again and everything is fine. If you hadn't verified the program when you **SAVE**d it, you wouldn't have known that the program wouldn't work until you tried to **RUN** it later—when the original was erased from the computer's memory. It is better to discover a problem while the original is still in memory! The crucial element here is that **VERIFY** checks the **SAVE**d version with the program that's in memory. The time to **VERIFY** is while the original is right there in front of you.

To use the **VERIFY** command, rewind the tape to the beginning of the program (in this case, it's the first one on the tape). Type:

```
VERIFY "DEMO"
```

and press RETURN. (When the program you want to **VERIFY** is first on the tape, it is not necessary to include the name of the program.) After typing **VERIFY**, you can press RETURN and the first program is checked. The screen replies:

```
PRESS PLAY ON TAPE
```

Press PLAY and the screen displays:

```
SEARCHING FOR DEMO
FOUND DEMO
VERIFYING
OK
READY
```

with a few seconds between each message. When the **VERIFY**ing is completed, the tape recorder automatically stops.

As long as the **SAVE**d version matches the version in memory, the above message appears. But what if there's a problem? If the name is misspelled, the computer searches the entire tape and can't complete the **VERIFY** command because the program isn't there. If, however, a problem occurred during the **SAVE** procedure itself, the following message appears:

```
?VERIFYING
ERROR
READY
```

It may be difficult to determine exactly what the problem is, but you may

want to **SAVE** the program on another section of the tape or on the other side. It's an even better idea to use another tape.

With the tape that has DEMO and SAMPLE on it, rewind to the beginning and **VERIFY** SAMPLE. After typing **VERIFY** "SAMPLE" and pressing RETURN, the screen displays:

```
SEARCHING FOR SAMPLE
FOUND DEMO
FOUND SAMPLE
VERIFYING
OK
READY
```

You can **VERIFY** a program regardless of its location on the tape. If you **SAVE**d a program last week and don't currently have it in memory, though, it can't be **VERIFY**ed. Use **VERIFY** on all your important programs before erasing them from memory.

SAVE TO DISKETTE

In Chapter 4, the advantage of a disk system in terms of time was mentioned in relation to **LOAD**ing programs. You have the same time advantage when you **SAVE** and **VERIFY** programs on diskettes.

Formatting Diskettes

In order to **SAVE** a program on a diskette, you need a diskette that has been formatted. When you buy a diskette that has a program on it, that diskette had to be prepared before the program could be **SAVE**d on it. This preparation is called "formatting" and involves writing a sort of electronic map or grid on the diskette. Later, when the computer wants to store information on the diskette, it uses the electronic map as a guide to tell it where to place the information.

Before you can use a new diskette to store programs or data, you must also use the format procedure. It's best to use blank diskettes (disks that don't have programs on them), because the formatting process erases anything that may be on the diskette. If you put an unformatted diskette into the drive and ask for a directory, the message

```
SEARCHING FOR $
```

appears on the screen. The red light on the disk drive comes on (indicating it is trying to read information on the diskette) for a few seconds and then goes off. After a few more seconds, the light comes back on and then goes off again. This process goes on forever if you let it. The problem is that the

computer is looking for instructions on the diskette and doesn't find any. Assuming it has made a mistake, the computer goes back and tries to read the diskette again. There isn't any information on a blank diskette (but the computer doesn't know this) so it continues trying to do what it was instructed to do. How do you stop the cycle? The only thing you can do is pull the diskette out of the drive. Try to pull the diskette out when the red light is off. Why? As mentioned earlier, every time the red light is on, the computer is trying either to get information from or to the diskette. When the diskette has been removed from the drive, turn the computer off, wait a few seconds, and then turn the computer on again.

To format a diskette, follow these steps:

1. Insert a diskette in the drive according to Figure 4.5.
2. Close the drive door securely.
3. Type

   OPEN 15,8,15

and press RETURN. The screen displays

   READY

OPEN 15 tells the disk drive to prepare to communicate with you. This is called **OPEN**ing a "channel" and is necessary before you can write or read from a device other than the computer. In this case, the number of the other device is 8 which stands for the disk drive. The last number in the **OPEN** statement (15) is called the secondary address and this establishes limits for the device that was specified for writing or reading.

4. Type

   PRINT #15,"N:DISKNAME,DI"

and press RETURN. The diskette is being formatted and the identifiers DISKNAME (which can be any name with a combination of characters and/or spaces such as: MY PROGRAMS) and DI (any two characters like DW). DI is short for Disk Identifier. These identifiers are useful when you are using more than one diskette and need to differentiate between disks when files are being read to or from them. The red light on the drive comes on and stays on for about 85 seconds during the formatting process.

5. When the red light on the drive goes off, type:

   CLOSE 15

and press RETURN. This tells the computer you have completed your communication with the disk drive and the communication channel can be closed. That's all there is to formatting or preparing a diskette for use. It's a good idea to have several formatted diskettes available so that when you have a program in the memory of the computer and want to store it on a diskette there is a formatted diskette ready to use. It can be very discouraging to want to **SAVE** something only to find that there isn't a formatted diskette around.

SAVING INSTRUCTIONS

To **SAVE** to diskette, we can use the same program examples as used in the cassette explanation. If you don't have this program, it is the one in Chapter 3. Any program will work, but we will continue to use the names DEMO and SAMPLE. When you have a program typed or loaded, follow these steps to **SAVE** it:

1. Insert a formatted diskette into the drive and close the door securely (if you are unsure how to insert the diskette, see Chapter 4).

2. Type:

```
SAVE "DEMO",8
```

and press RETURN. The red light on the drive comes on while the program is being **SAVE**d. When the light goes off the program has been **SAVE**d and the screen displays:

```
READY
```

If you get a message like:

```
PRESS RECORD & PLAY ON TAPE
```

that means one set of quotation marks was omitted or the device number 8 was missing. Remember to put the name of the program inside the quotation marks and the comma outside the quotation marks. The 8 represents the device number for the disk drive and must always be used when **LOAD**ing or saving with diskettes (see Chapter 4 for more on device numbers). If the device number is left out, the VIC 20 assumes you are using a cassette and the keyboard locks.

If this happens, press the RUN/STOP key to regain control and retype the instruction.

Another frequent message is:

```
?DEVICE NOT PRESENT
```

This appears if a number other than 8 (for the disk drive) is entered.

Although a file name is not required when using cassettes, it is impossible to **SAVE** a program on a disk without one. If you were to type **SAVE** " ",8 to try not to give a name, the following appears on the screen.

```
?MISSING FILE NAME
ERROR
READY
```

As you might expect (or hope), just as you can **VERIFY** cassette **SAVE**s, you also can **VERIFY** disk programs in much the same way. Again the purpose is to compare a program just **SAVE**d with the original still in memory. To **VERIFY** DEMO type:

```
VERIFY "DEMO",8
```

and press RETURN. The computer checks to make sure the **SAVE**d version is exactly like the original. The following message is displayed:

```
SEARCHING FOR DEMO
FOUND DEMO
VERIFYING
OK
READY
```

if everything is as it should be. If, for some reason, the **SAVE**d version is different, the message is:

```
SEARCHING FOR DEMO
FOUND DEMO
?VERIFY
ERROR
READY
```

and you can be sure there is a problem. If the program isn't a crucial one, you may want to try to **SAVE** it again on the same diskette. If you get the message again, or the program is important, the best suggestion is to try another diskette (this is a time when having several formatted diskettes available is very helpful) and go through the **SAVE** and **VERIFY** routines again.

Now, **SAVE** the program in memory again and call it SAMPLE. Type

```
SAVE "SAMPLE",8
```

and press RETURN. You actually have two copies of the same program now, one called DEMO and one called SAMPLE. Next, **VERIFY** SAMPLE.

The screen will tell you when SAMPLE is found and how the **VERIFY** procedure goes (an error message appears if there is a problem, "OK" is reported if all is well).

OUTPUT TO A PRINTER

The last method of saving information to be discussed is using a printer to get a "hard copy" or printed version of the program. The VIC 20 printer connects to the back of the computer in the serial port (see Chapter 4 for a diagram of the installation). Make certain the printer is properly connected and turned on before the VIC 20 is turned on.

You can print one or more lines of information in a program, or you can have the lines of a program printed out. When using a printer, the word to get a permanent copy of your program is **LIST**. The example in this chapter explains how to get a line-by-line **LIST**ing of the program as well as how to print material from within a program.

Suppose you have a program like the one below. When the program is keyboarded, the word **RUN** typed and the RETURN key pressed, the program executes. In the case of the program below, the phrase "THIS IS A TEST" should appear on the screen and then the READY message. Type in the following program to use in the examples for making use of the printer as a device to **LIST** your programs.

```
20 PRINT "THIS IS A TEST"
30 END
```

Press RETURN, type **RUN** and press RETURN again. The screen should look similar to this:

```
20 PRINT "THIS IS A TEST"
30 END
READY
RUN
THIS IS A TEST
READY
```

The computer sends the output to the screen. That is, when you type **RUN** and press RETURN, the execution of the program is displayed on the screen. What you want, though, is to have what is inside the quotation marks sent to the printer. To get that, a few pieces of information must be given to the computer by adding three lines to the program. As each new line is typed, we'll discuss its purpose. To begin, type

```
5 OPEN 9,4
```

and press RETURN. This tells the computer that you want to **OPEN** a file numbered 9 (this is often called the logical file and can be any number from 0 up to 255) and that you want to access the device number 4 (the printer can be either 4 or 5, your VIC 20 manual lists all the devices and their corresponding numbers). This line tells the computer to **OPEN** file number 9 and to send the information for logical file 9 to a printer (device 4). Next, it is necessary to use the **CMD** instruction to tell the VIC 20 that the printer is to be the primary output device instead of the screen. Type

```
10 CMD9
```

and press RETURN. **CMD** initializes the command channel 9 and tells the computer that file number 9 is the output channel. The file number always follows the **CMD** instruction. If, by mistake **CMD** is followed by the device number, the following message appears:

```
?FILE NOT OPEN
ERROR
```

and the program won't execute. You won't know this, however, until you type **RUN**. If the FILE NOT **OPEN** message appears, you know that you have entered a file number other than the number specified in the **OPEN** statement. This can be corrected by either retyping the line or moving the cursor up to the **CMD** line and typing over the wrong number with the correct one.

After a file is **OPEN**ed and you are finished with it, the file must also be **CLOSE**d. If a file isn't **CLOSE**d, an error message may result later. To accomplish the closure of a file in our example program, type

```
25 CLOSE9
```

and press RETURN. As it stands now, the program seems to be a mess of line numbers in the wrong order. Fortunately, the VIC 20 can straighten out the jumble and put the program in proper numerical sequence. To see the program correctly, type

```
LIST
```

and press RETURN. The screen looks like the following:

```
 5 OPEN9,4
10 CMD9
20 PRINT "THIS IS A TEST"
25 CLOSE9
30 END
```

and in a couple of seconds adds

READY

By using the **LIST** command, you have a line-by-line copy of your program. Notice that the program line numbers are in the correct order now.

To get what's inside the quotation marks in the program to go to the printer, type **RUN** and press RETURN. The printer types

THIS IS A TEST

and a few seconds later responds with

READY

Everything that normally appears on the screen is displayed by the printer when **CMD** is used. Now type

LIST

and press RETURN. Notice that the **LIST**ing of the program lines that earlier appeared on the screen now are displayed on the printer. This is the way the printer is used as a saving device. When the computer is turned off, the program and all the hard work that went into it is gone. With **LIST**ing to the printer, however, the creative work of developing the program is not lost when the computer is turned off because you have a duplicate of the program on paper.

What happens if you take out line 10 (where the **CMD** instruction is given)? Try it by typing

10

and press RETURN. Now use **LIST** and press RETURN. The program **LIST**ing is only on the screen. Type **RUN** and notice that THIS IS A TEST only is displayed on the screen. It is the use of **CMD**, then, that provides the opportunity to use the printer as a saving device.

In any program that you have, the following additions are necessary to send a **LIST**ing or **RUN** a program to the printer:

```
 5 OPEN 1,4
10 CMD1
50 CLOSE 1
```

where 5, 10, and 50 are program line numbers (the line numbers in your program might be different). The file number (1 here) can be any number from 0 to 255, and the 4 represents the device number of the printer. **CMD**1 opens the command channel to file number 1 (this file number

must be the same as the file number in the **OPEN** statement), and **CLOSE** is used to close the file number 1.

There is a way to get information to the printer without using the **CMD** statement. This method lets you send some information to the screen and some to the printer. For example, if you are working on a home budget program you may want some questions, such as "How much did you spend on groceries this week?" to appear on the screen and the summary chart of expenditures to be printed. The use of **PRINT#** allows this distinction. Type in the following short program to illustrate how this works:

```
10 OPEN 1,4
20 PRINT "THIS MATERIAL IS TO BE DISPLAYED ON THE
   SCREEN"
30 PRINT#1, "THIS MATERIAL IS TO BE PRINTED BY THE
   PRINTER"
35 CLOSE1
40 END
```

Line 10 **OPEN**s file number 1 and sends it to device number 4 (the printer). Line 20 is a standard **PRINT** statement. Line 30 tells the computer that the information in quotation marks is file number 1 and that it is to be sent to the printer. Line 35 **CLOSE**s the file and Line 40 ENDs the program. **PRINT#** always transmits information to the device number specified in the **OPEN** statement.

To see this work, type **RUN** and press RETURN. The message in Line 20 is displayed on the screen and Line 30 is printed out for you.

If Line 20 is displayed and then the following message appears:

```
?SYNTAX ERROR IN 30
```

the most likely culprit is the comma after PRINT#1. Remember to always include a comma after the file number in this statement. If this message appears, you can retype the line or use the cursor key S and the INST/DEL key to edit the line.

Using a cassette or diskette, the program can be **LOAD**ed (using directions from Chapter 4) at a later time. With a printer, the line-by-line **LIST** of the program is available but, to use the program again when only a printed copy is available, it must be typed in again on the VIC 20.

Chapter 6

Programming in BASIC

This is the first of two chapters that will deal specifically with the popular computer language called BASIC. We assume at this point you have already read Chapter 3 and that you are familiar with the procedures described there for correcting and editing programs as you type them into the computer. A review of the material in Chapter 3 would be helpful before proceeding.

There are at least two hundred books currently in print that teach how to write programs in BASIC. At least ten books deal specifically with programming the VIC 20 in BASIC. We will not try to teach you how to write BASIC programs on your computer in two chapters. Instead, the goal of Chapters 6 through 8 is to introduce the concept of BASIC and its use on the VIC 20. In addition, the chapters provide you with enough knowledge to use BASIC programs written for the VIC 20. There are many books of BASIC programs for the VIC 20. Several magazines regularly publish programs that run on the VIC 20 (see Chapter 1 for sources of programs).

After reading the next two chapters, you should be able to understand how BASIC programs work and you should be able to take a VIC 20 BASIC program from a book or magazine, type it on your computer, and use it. You will know enough BASIC to "debug" a program that isn't working as it should and even make changes or enhancements that interest you.

How to Use This Section

It would be possible for a reader to plow through these three chapters without a computer and still learn some BASIC. Our advice, however, is to read the BASIC chapters seated in front of the computer. Learn what all the new terms mean, and use your computer to do the examples scattered through the reading. Don't try to memorize definitions and instructions. Practice enough so that working in BASIC is comfortable. It is not necessary to be able to say off the top of your head what **GOSUB** 140 means; you

only need to know where to look for that information. After working with BASIC for a while it will become a familiar second language through experience, not memorization.

This introduction to BASIC is not intended to be a comprehensive tutorial on all aspects of programming in this versatile language. If the material presented here intrigues you, there are several good books on the topic of programming the VIC 20 in BASIC.

INTRODUCTION TO BASIC

There are several important differences between the language spoken by a computer and regular English. In English, the meaning will usually be clear even if a word or two is mispronounced or misspelled. BASIC and other computer languages are not so forgiving. You must say precisely what you intend to, and say it exactly as the computer expects.

Another difference is the punctuation. Commas, semicolons, and periods clarify the meaning of written English, but the rules for their use vary. You would be able to understand a letter from Uncle Harry even if he managed to write a whole page without a comma or period. In BASIC, these punctuation marks are often as important as the letters and numbers. Leaving out one comma can prevent an entire program from running properly.

Finally, English is a very rich language. There are usually several words that have similar meanings. A 1956 Chevrolet could appropriately be called a car, auto, automobile, or vehicle. BASIC is not so well endowed. Often there is only one way of saying something.

The words BASIC understands are called "key words." To make the computer do something, these key words are used to make up STATEMENTS. Statements correspond to sentences in English. The term "statement" actually has two uses in computer programming. It can mean "program line," and it is used often to mean "key word." Three examples of statements (i.e., program lines) are given below:

```
10 LET A = 2
20 LET B = 4
30 PRINT A + B
RUN
```

Each statement is preceded by a line number. When a computer is told to **RUN** a program, it begins with the statement that has the lowest line number and follows the instructions given in that statement. It then proceeds through the program until all the statements have been executed. When it reaches line 30, it will do the arithmetic requested (2 + 4) and

print the result on the screen. Type in this program on your VIC 20. If some of the details of typing in programs are fuzzy you may want to review Chapter 3.

The three statements you typed in make up a simple PROGRAM. A program is a set of instructions or statements that tell the computer how to solve a particular problem. When the three-statement program is typed in, type **RUN** and press the RETURN key. If the computer responds by printing 6, you are off and running.

At this point, we have several new terms. BASIC is a popular computer language that enables the computer to understand a number of special or "key words" such as **LET** and **RUN**. BASIC also has a set of rules about how punctuation marks are used. Using these rules and the key words BASIC understands, a person can write programs the computer can execute or carry out. Programs are composed of lines or statements, each of which begins with a number (the line number). Statements contain key words, used according to the rules of BASIC.

You tell a computer you want it to execute a program (follow the instructions in the program) by giving it the **RUN** key word. This is the normal way a computer is used. There is, however, another way. If you simply type the following line:

```
PRINT 2 + 2
```

the computer will add 2 and 2 immediately after you press the RETURN key. If you do not put a line number in front of a statement, the computer assumes you want the instructions carried out immediately. This method of using a computer is sometimes called the "calculator mode" or the "immediate mode." It is handy since it lets you use the computer as you would a pocket calculator. The key word **PRINT**, if it is used to tell the computer to do something immediately, is called a "Command." If **PRINT** is part of a program it is called a "Statement" in most textbooks. Most of the key words in BASIC can function as either commands or statements. That is, they will work in either the calculator mode or the program mode.

In the next section you will learn more about some of the most frequently used key words in the VIC 20 version of BASIC.

JOB CONTROL COMMANDS

Several of the key words available in BASIC are used to tell the computer how to manage its work. They are listed below with a description of the job each performs:

RETURN. This command has its own key on the keyboard. **RETURN** is used to tell the computer you have finished typing a line of instructions.

When you press the RETURN key, the computer enters the line of instructions that begins with a line number in its memory. If there is no line number, the instructions are executed immediately. **Note:** There is another **RETURN** key word which will be explained later when the key word **GOSUB** is discussed.

RUN. This command has already been used several times. When you give the **RUN** command, it tells the computer you want it to execute the instructions contained in the program currently in the computer's memory.

Normally, **RUN** tells the computer to begin with the instructions in the line with the lowest number. There is a variation of **RUN**, however, that specifies where to begin. If you tell the computer **RUN** 70, the computer will start executing instructions in line 70 and above.

NEW. There are times when you want to stop what you are doing and start over. One way to accomplish that is to pull the plug on the computer. Another way is to use the command **NEW**. **NEW** erases any program currently in the computer's memory. You can then start fresh. Before you use **NEW**, be sure you really do want to erase everything in memory. Once it is done, it's done.

HOME/CLEAR. The instructions **HOME** and **CLEAR** have their own key at the top right of the keyboard. The use of that key was discussed in Chapter 3. In essence, when you use **HOME** or **CLEAR** by pressing the key, you are using these key words in the calculator mode. The instruction is carried out immediately. You do not even have to press the RETURN key. Can you clear the screen and/or move the cursor to its home position by using these key words in a program? Yes, you can, but not by using **HOME** or **CLEAR**. Type the following program into your computer:

```
10 PRINT "THIS IS THE TOP LINE."
20 PRINT "THIS IS THE SECOND LINE."
30 PRINT "SHIFT(CLR/HOME)"
```

Line 30 should be read as an instruction to type 30 PRINT " and then hold down the shift key and press the **CLR/HOME** key. Finally, type the last quotation mark and press RETURN. Holding down the shift key and pressing the **CLR/HOME** key produced an odd graphic character on the screen which looks like the heart on the 5 key except the heart is white and the border around it is filled in with color. This is a "reverse" graphic character.

Most of the keys on the VIC 20 that perform a particular function such as clearing the screen or changing the color of the border or background will produce a particular reverse graphic character when that key is pressed

within quotation marks. When the computer comes to one of these characters as it executes a program, it acts as if the key that produced the character had been pressed. **RUN** the program now and watch what happens.

The two lines of print produced by lines 10 and 20 seem to flash on and off rapidly. After 10 and 20 produce their message the computer reads line 30 (**PRINT** "SHIFT(**CLR/HOME**"). If you held down the shift key and pressed the **CLR/HOME** key, the computer would execute a **CLEAR** command which erases the screen and moves the cursor to the home position. Line 30 causes it to do just that. The screen is erased and the cursor goes to the top of the screen. Now line 40 tells the computer to go back to line 10 and start executing the instructions again. The two lines of print are produced again; line 30 erases them again, and the process begins again. What would happen if line 30 were deleted? Delete 30 by typing 30 and pressing RETURN. Now run the program again. This time, since the lines are not erased and the cursor does not go to the home position, the screen fills up with the print message from lines 10 and 20. As each new line is added to the bottom of the screen, a line at the top is pushed off the screen.

BREAK. On most computers this command has a key of its own that is labelled BREAK. On the VIC 20 you can tell the computer to stop executing a program (BREAK the program) by pressing the RUN/STOP key. If your computer is still running the example program that illustrated how CLR/HOME works in a program, press the RUN/STOP key now. The program will stop executing and the computer will display a message that says BREAK IN 20. (It may give the number 10, 20, 30, or 40). That means the computer stopped while it was working on line 20.

END. This key word tells the computer it has reached the end of the program. It causes the computer to stop executing instructions.

SAVE and **LOAD**. These key words are used to load and save material on cassettes. They are dealt with in detail in Chapters 4 and 5.

LIST. This command tells the computer that you wish to see the lines of the current programs displayed on the video display. This is usually called a "listing" of the program. **LIST** will cause the entire program to be listed. If there are no more than 21 lines, all the program will be shown on the screen. If your program is more than 21 lines long, the computer can't get all of it on the screen at once. **LIST** will cause the lines to fly by on the screen with the final 21 being the only ones you can actually read. There are several other formats for **LIST**, however:

LIST #– The listing begins at the line number specified. **LIST** 30– will begin listing lines at line 30.

LIST # Without the – after the number, all you will get is the line specified. **LIST** 30 will cause only line 30 to be printed.

LIST –# If the – precedes the line number, the computer will list lines from the beginning of the program to the line specified. **LIST** –80 will cause all the lines with numbers up to 80 to be listed.

LIST #–# This version will print a section of the program. **LIST** 30–95 will cause all the lines from 30 to 90 to be listed.

EDIT. This is not a key word in VIC 20 BASIC, but it is a useful feature of the VIC 20. You have already learned how to edit a program line using the Right/Left Arrow Key and the INST/DEL key. Up to this point, if you typed a line incorrectly and pressed RETURN the only thing you could do is retype the entire line.

The VIC 20 actually has another method of editing lines after the RETURN key has been pressed. Find the Up/Down Key on the bottom, right of the keyboard. You can use that key to move the cursor anywhere you want on the screen. Pressing that key alone will move the cursor down. Holding down the SHIFT key and pressing the Up/Down key will cause the cursor to move up. If you move the cursor up to a line that has already been entered in a program (by pressing RETURN), you can still use the Right/Left Arrow key and the INST/DEL key to correct an error. Then press the RETURN key. Use the Down Arrow key to move the cursor to a line with nothing printed on it and type **LIST**. You will see that the line you corrected on the screen is now listed in the corrected form.

You can even change the line number of a set of instructions. Suppose there is a line 20:

```
20 PRINT "THIS IS A DEMO LINE"
```

If you move the cursor over the 2 in 20 and press the 4 key, the line number will change to 40. Now if you press RETURN, move the cursor to a clear line, and type **LIST** you will have a line 20 and a line 40:

```
20 PRINT "THIS IS A DEMO LINE"
40 PRINT "THIS IS A DEMO LINE"
```

If the program already had a line 40, the one you created will replace the original line 40. This ability to create a new line from an old one is handy when you are typing in many lines that are very similar. You can "create" a new line by changing the number of an old line and then editing the new one where necessary.

CONT. This is short for **CONT**inue. If you interrupt execution of a program by pressing the **RUN/STOP** key you can tell the computer to pick up where it left off by typing **CONT** and pressing the RETURN key. A **STOP** statement in a program can also be overridden by typing **CONT** and pressing RETURN.

STOP. This key word is a statement that works a lot like the command BREAK. A program line like 45 **STOP** will cause program execution to stop at line 45. If you want the computer to continue executing the program after it gets to the line containing the **STOP** instruction, just use the **CONT** key word. The computer will pick up where it left off and continue execution.

CLERICAL INSTRUCTIONS

PRINT, Comma, Semicolon, SPC, TAB

Now that you've gotten the hang of it, we'll pick up the pace a bit. In the following sections additional key words and programming rules will be presented and explained. Generally, the explanation will be accompanied by a short demonstration program. If this is your first foray into programming, we advise you to read the explanations carefully and **RUN** each of the example programs.

PRINT. This statement causes the computer to print out or display whatever follows **PRINT**. You have used it several times already. If material following **PRINT** is enclosed in quotation marks, the computer will print it exactly as typed. The material inside the quotation marks is called a STRING or a literal string. The instruction **PRINT** "THIS IS A TEST" would cause the computer to print the string THIS IS A TEST on the screen.

If there are no quotation marks around the material to be printed, the computer behaves a little differently. Consider the program below:

```
10 LET THIS = -30
20 PRINT THIS
```

Type in and **RUN** the program. Line 10 tells the computer that you want to use a variable named THIS and that you want to make THIS equal to −30. Line 20 does not print the name of the variable (THIS). Instead, it prints the value of THIS which is −30. Thus, whether you use quotation marks or not will make a big difference in what the computer does.

COMMAS and SEMICOLONS. Please add the following lines to the two lines you typed in to see how **PRINT** works:

```
30 PRINT "THIS IS";THIS
40 PRINT "THIS IS ";THIS
50 PRINT "THIS IS ",THIS
60 PRINT "THIS IS",THIS
```

Note that the only difference between line 30 and line 40 is the space between the S in IS and the quotation mark. Now **RUN** the program. Your display should look like this:

```
 -30
THIS IS -30
THIS IS  -30
THIS IS     -30
THIS IS     -30
```

The -30 on the first line of the display came from line 20. Then lines 30 and 40 produced almost identical displays. The literal string THIS IS was printed first by both lines. Then the value of the variable THIS was printed. In line 40, a space inside the quotation marks makes the line neater and easier to read because the number 30 is not directly up against the S in IS. The semicolon that follows the literal string tells the computer to print whatever comes immediately after the last item printed. If you use a semicolon to separate two items that are to be printed, you will need to add any spaces needed yourself (i.e., inside the quotation marks of the literal string).

Lines 50 and 60 produced exactly the same results even though one has a space inside the quotation marks and one does not. Why? The key lies in using the comma rather than the semicolon as a separator. The VIC 20 divides the screen into two "areas" or zones. The second zone begins at the 11th position on the screen. A comma instructs the computer to begin printing whatever follows in the next print zone rather than right up against the material already printed. Thus, a comma generally separates material while a semicolon puts it together.

Both commas and semicolons show no respect for the end of a line. If you tell the computer to print material that is too much to fit on a line, the computer will type all it can on one line and put the rest on the next line below. The program below illustrates the point. Use the key word **NEW** and then type it in:

```
 5 FOR X=1 TO 20
10 PRINT "AND";
20 PRINT "YET";
30 PRINT "ANOTHER";
40 NEXT X
```

RUN the program and you will see that it fills up 11 lines on the display and spills over a bit to the twelfth line. The three literal strings AND, YET, and ANOTHER are printed end to end until the program loop has been executed 20 times. Now add the following lines:

```
 50 PRINT
 60 PRINT
 70 FOR B = 1 TO 7
 80 PRINT "EVEN",
 90 PRINT "MORE",
100 NEXT B
```

RUN the expanded program. You get the same letters at the top of the screen. Then lines 50 and 60 give us a blank line (50 gets us off the twelfth line, 60 skips one line). Then the second **FOR NEXT** loop prints a pattern of EVEN MORE on the screen seven times. The comma on lines 80 and 90 causes the computer to space over and print material at the beginning of each of the two print zones.

You may want to experiment with the **PRINT** instruction and with commas and semicolons a bit to become more familiar with them.

A final note on **PRINT**. If you tell the computer to print something and don't put a comma or semicolon after it (e.g., **PRINT** "HELLO") the computer will automatically go to the beginning of the next line when it encounters the next **PRINT** instruction.

Another final note on **PRINT**. Since **PRINT** is used so often, VIC 20 BASIC lets you substitute a question mark for **PRINT**. Typing 50 **PRINT** and 50 ? will produce the same thing. The VIC 20 will change 50 ? into 50 **PRINT** when you **LIST** the program. Most of the key words used in VIC 20 BASIC can be abbreviated. The abbreviation for most key words is the first letter of the word plus the next letter typed with the SHIFT key depressed. **CONT**, for example, can be typed by typing a C, then holding down the shift key and pressing the O key. Appendix A lists the shortened versions of all the key words in VIC 20 BASIC.

SPC. This key word is used with **PRINT** to tell the computer to move over a specified number of spaces before printing something on the screen. The general format for its use is:

PRINT SPC(#);material to be printed

The number following **SPC** indicates how many spaces to move over before printing the material that follows. **PRINT SPC(8);**"THIS WAY OUT" will cause the literal string THIS WAY OUT to be printed eight spaces over from the left side of the screen.

TAB. This key word works much like the tab key on a typewriter except that the computer sets the location of the tabs and you cannot modify them. Here is a demonstration program:

```
20 FOR X = 1 TO 12
30 PRINT 3;TAB(18);16
40 NEXT X
```

Be sure to type **NEW** and press RETURN before typing in this program. When you have it typed correctly, **RUN** the program. You should get 12 lines of material on the screen. Line 20 begins a "**FOR NEXT** loop," something that will be explained later, while line 40 is the end of the loop. In essence, lines 20 and 40 will cause line 30 to be executed 12 times. Line 30 tells the computer to print the number 3. Then the instruction **TAB**(18) tells the computer to move the cursor to position 19 on the screen. Why 19? Because the programmers who wrote the directions for BASIC on the VIC 20 set it up that way. The cursor moves to the position that is one more than that specified by **TAB**. For other computers, the **TAB** instruction would cause the cursor to move to the location specified. At position 19 the computer prints the number 16. The **TAB** key word tells the computer to move to a particular "location" on the screen while the **SPC** key word tells the computer to move a certain number of spaces.

Chapter 7

Getting Information Into the Computer

There are three major ways VIC 20 BASIC allows the programmer to assign "values" to "variables." You can do this with the **LET**, **GET**, and **INPUT** statements.

LET and Types of Variables

You have already used **LET** several times in the example programs. The general format for **LET** is:

LET variable name = value

The **LET** key word tells the computer what value to associate with a particular variable name. For example, **LET** B = 22 tells the computer that until further notice the variable named B will equal (have the value of) 22.

There are two basic types of variables, numeric and string. B, in the example above, is a numeric variable. **LET** B$ = "HELLO" tells the computer that the string variable B$ is to equal HELLO. You must enclose string variables in quotation marks (e.g., **LET** B$ = "FINE", not **LET** B$ = NOT FINE). The $ after B designates the variable as a string variable. If the variable name does not end with a $ the computer assumes it is dealing with a numeric rather than a string variable. The VIC 20 will not accept **LET** A$ = 22 or **LET** A = "HELLO" because there is a mismatch between the type of variable specified (numeric or string) and the type of value assigned. If you type **LET** A$ = 22 and press the RETURN key, the computer will respond with:

```
?TYPE MISMATCH
ERROR
```

This is the VIC 20's way of telling you it does not understand the instructions given. This error message is only one of many the VIC 20 may

provide. See Appendix B for an explanation of VIC 20 error messages and suggestions for how to correct your program.

If you assign a value to B (or B$) early in a program (e.g., 10 **LET** B = 22) and then assign another value to B later (e.g., 30 **LET** B = 66), the value of B will be the last one assigned (i.e., 66 in this example).

Up to this point the value assigned with the **LET** statement has been a simple numeric value (22) or a literal string value ("HELLO"). **LET** is much more versatile, however. **LET** can use another variable to assign the value to a new variable (**LET** B = A or **LET** B$ = A$) and it can use an "expression" (**LET** B = 2 + 5). These **LET** instructions also are acceptable:

```
LET B = B + 2
LET B = B + A
```

The only restriction in the above examples is the requirement that any variable name on the right side of the equal sign must have been defined previously in the program. The program line 30 **LET** B = B + A is fine if an earlier line defined the value of A (e.g., 10 **LET** A = 95).

We should note here that the computer can use variable names that are more complex than A or B$. Standard numeric variables can have names of up to two characters long (e.g., BG, K1, R9). Both numeric and string variable names must begin with a letter. String variables must end with a $ (e.g., TS$, BG$, B1$, A8$).

There are, however, two additional limitations on the way you name your variable. VIC 20 BASIC has a number of key words which it treats as instructions. You have already used key words such as **PRINT** and **LET**. You cannot use a key word to name a variable. **ABS**, for example, is not an acceptable name since it is a key word. The same is true for **ATN**, **CLR**, **COS**, **SQR**, **SPC**, and many more. Table 7.1 lists the key words used in VIC 20 BASIC.

We mentioned that you can use only two characters in a numeric name and two characters plus a $ in a string variable name. That is not exactly true. You can actually use an instruction such as **LET** TEST = 24 or **LET** INVENTORY$ = "CAR POLISH". The VIC 20 will accept these instructions but it will use only the first two characters in the name. The variables in the last examples thus have the name TE and IN$ as far as the VIC 20 is concerned. That means TEST and TEXT are the same variable to the VIC 20 since they both begin with TE. You will avoid problems if you select names that meet the VIC 20's expectations.

There is one more twist in the use of **LET**. You really do not have to use it. The instructions **LET** A = 10 and A = 10 do the same thing. The **LET** is understood in the second instruction just as the "you" is understood in the

sentence "Be careful!" **LET** is not actually typed out in many programs because the computer will execute a **LET** instruction without it.

ABS	INPUT	RETURN
AND	INPUT#	RGR
ASC	INT	RIGHT$
ATN	KEY	RJOY
AUTO	KILL	RND
CHANGE	LEFT$	RPEN
CHAR	LEN	RPOT
CHR$	LET	RSND
CIRCLE	LIST	RUN
CLOSE	LOAD	SAVE
CLR	LOG	SCNCLR
CMD	MERGE	SGN
COLOR	MID$	SIN
CONT	NEW	SOUND
COS	NEXT	SPC
DATA	NOT	SQR
DEF	OFF	ST
DELETE	ON	STATUS
DIM	OPEN	STEP
DRAW	OR	STOP
DUMP	PAINT	STR$
EDIT	PEEK	SYS
END	POINT	TAB
EXP	POKE	TAN
FIND	POS	THEN
FN	PRINT	TI
FOR	PRINT#	TIME
FRE	PROG	TI$
GET	RCOLR	TIME$
GET#	RDOT	TO
GOSUB	READ	TRACE
GOTO	REGION	USR
GRAPHIC	REM	VAL
HELP	RENUMBER	VERIFY
IF	RESTORE	WAIT

Table 7.1 Reserved keyword names in VIC 20 BASIC.

INPUT

When the computer comes to an **INPUT** statement, it stops and waits for the operator to provide it with some data. If **INPUT** is followed by a numeric variable name (e.g., **INPUT** B or **INPUT** BALANCE), the computer expects you to type in a numeric value and press ENTER. If **INPUT** is followed by a string variable name (e.g., **INPUT** B$ or **INPUT** NAME$), the computer expects you to type a string (don't enclose it in quotation marks) and press ENTER. If you give a numeric value when the computer expects a string value or vice versa the computer will reject your input. Here is a short sample program:

```
10 PRINT "TYPE IN A NUMBER"
20 INPUT A
30 PRINT A
40 PRINT "TYPE IN A STRING"
50 INPUT A$
60 PRINT A$
```

RUN this program. The screen will display TYPE IN A NUMBER on one line and a ? will appear on the next line. The computer is waiting for you to type in a numeric variable. If you type a number and press ENTER, the computer will set A to equal the number you typed. Then it will print that value (line 30). Line 40 tells you to TYPE IN A STRING and line 50 tells the computer to look for a string. Type in TESTING and press RETURN. The computer will print TESTING and then print READY.

Now what happens if your input is not what the computer was expecting? **RUN** the program again and type in TESTING when the computer expects a numeric value. When you press RETURN the computer prints the message ?REDO FROM START. That means it did not expect you to type in a string and would like you to try again. The problem is a mismatch between the variable name (A) and the type of data input (TESTING). Type the number 34 and the program will continue. The computer accepts the number when you press RETURN and prints it on the screen. It goes on to lines 40 and 50. Now it is looking for a string variable. Type in 34 and press ENTER. Surprised by the results? The computer accepted 34 as a string with no problems and printed 34. That means A equals 34 and so does A$. There is a difference in the two variables, however. Add the following two lines to the program:

```
70 PRINT A + 12
80 PRINT A$ + 12
```

RUN the program and give both A and A$ the value of 34. When the

computer gets to line 70 it will compute 34 + 12 and print 46. Then it will print the following:

```
?TYPE MISMATCH
ERROR IN 80
```

You cannot add a string variable to a numeric variable. Thus, even though A$ equals 34 you will not be permitted to use it in math operations. To the computer, the 34 value of A$ is simply a two-character string—not a number. The variable A can be manipulated mathematically while the variable A$ cannot.

Up to this point we have used simple **INPUT** statements in which the value of one variable was requested. An **INPUT** statement can be used to assign values to more than one variable. INPUT A,B,C would cause the computer to print a ? on the screen. If you type in 44 and press RETURN, the computer will type a double question mark (??) to indicate it expected more. Type in 66, press RETURN, and another ?? will appear since there are three variables to be assigned values. Type in 77, press RETURN, and the computer will be happy. A will equal 44, B will equal 66, and C will equal 77. You can also input all three values at once. If you had typed 44, 66, 77 and pressed RETURN the computer would have accepted all three values at once. You can even mix numeric and string variables (**INPUT A$**, B, C$) but you must be careful to enter the values in the correct order (e.g., STRING,NUMBER,STRING) and separate each with a comma.

In the examples used thus far, the message associated with an **INPUT** statement has been placed in a **PRINT** line just before the **INPUT** line. It is also possible to put the message in the **INPUT** line. Here are some examples:

```
10 INPUT "TYPE IN A STRING";A$
20 INPUT "TYPE IN A NUMER";A
```

This type of **INPUT** statement requires your message to be enclosed in quotation marks. The ; after the final quotation mark serves as a separator and tells the computer that the variable name comes next. Here is another example:

```
10 INPUT "TYPE IN A STRING AND A NUMBER";A$,A
```

This version lets you type in two variables (which must be separated by a comma).

There is one final point about **INPUT** statements. The RUN/STOP key has already been mentioned as a means of telling the computer to stop executing a program. **RUN** your program one more time. When the

computer asks you to input a value for A, press the RUN/STOP key. Normally, you would be able to stop execution of a program by pressing the RUN/STOP key. That will not work when the computer is waiting for input. You can stop execution of the program during execution on an **INPUT** statement by holding down the RUN/STOP key and pressing the RETORE key. This causes the computer to execute a "reset" instruction. It clears the screen and prints READY. The program you are executing is not erased from memory but the value of all variables in the program is set to 0. **LIST** and **RUN** now will allow you to view the program lines on the screen or execute the program again.

GET and GOTO

One of the problems with using **INPUT** is the requirement that you press RETURN after typing the necessary data. That is usually no great burden, but there are times when pressing RETURN creates problems. Many games, for example, begin with instructions. You tell the computer whether you want instructions or not by pressing one of two keys. Then you tell the computer you have finished reading the instructions by pressing another key. Novices often forget to press the RETURN key in such situations and end up wondering why the program doesn't work. This sort of frustration can be avoided if you use **GET** instead of **INPUT**. Type **NEW** to eliminate any program in the memory of the computer and then type the program below. It will illustrate the use of **GET**:

```
10 PRINT "NEED INSTRUCTIONS?"
20 PRINT "PRESS Y FOR YES"
25 PRINT "PRESS N FOR NO"
30 GET A$
35 IF A$ = "Y" THEN GOTO 60
40 IF A$ = "N" THEN GOTO 100
50 GOTO 30
60 PRINT "INSTRUCTIONS PROVIDED HERE"
70 PRINT "PRESS G TO BEGIN GAME"
80 GET B$
85 IF B$ = "  "GOTO 80
100 PRINT "THE GAME BEGINS HERE"
```

The **GET** statement in line 30 lets you define the value of the string A$ simply by pressing a key on the keyboard. Line 10 in the example program asks if you need instructions. Lines 20 and 25 tell you to press the Y key for instructions, and the N key if you don't want them (e.g., you've played the game before and are ready to begin). Lines 35 and 40 check to see if the string A$ equals Y or N. If it equals Y, the computer obeys the **GOTO**

instruction that tells it to go to line 60 (where the instructions would be printed). If A$ equals N the **GOTO** instruction in line 40 tells the computer to jump to line 100 and begin executing instructions from that point. As soon as you press Y or N, the computer follows one of the **GOTO** instructions in lines 35 and 40. The computer will execute lines 35 and 40 before you press Y or N, however. When it does, A$ will equal nothing since you have not pressed a key. Line 50 sends the computer back to line 30. The program will thus "loop" through lines 20 through 40 until a key is pressed.

If you press the Y key, the computer prints the material in lines 60 and 70. Normally that might be a screen of instructions. You are also told to press G when you finish reading the instructions and are ready to play the game. Line 80 gets the value of B$. Then we come to line 85:

```
85 IF B$ = "  " THEN GOTO 70
```

As long as you do not press a key on the keyboard, the string variable B$ will equal nothing. Nothing is indicted by " " which is also called a "null string" since there is nothing between the quotation marks. So unless you press a key, the computer will execute the instructions in lines 70 and 80 over and over. When you do press a key, the **GOTO** 80 instruction in line 80 will not be exeucted since B$ does not equal " ". The computer goes on to the next line of instructions which is line 100. Note, however, that although the computer tells the person to use the G key, any key will cause the computer to move on. This was done for several reasons. You don't want to tell the user to press "any key" because **GET** accepts the RUN/STOP key as a signal to BREAK and stops execution. In addition, if the user happens to miss the G key and presses another nearby key we can be fairly sure that he or she is ready to begin the game even though G was not pressed.

GET does have its limitations. It can only have a one character value (**GET** A$ can equal A, or B or 1 and so on, but it cannot equal AVERAGE). In addition, the computer quickly scans the keyboard to see what key is pressed when it encounters **GET**. That means you must write the program in such a way that the computer will check the keyboard over and over until a key is pressed. Consider the example below:

```
 5 GET A$
10 PRINT A$
20 GOTO 5
```

If you **RUN** this program, nothing will appear to happen until you press a key. Then the character for that key will appear on the left side of the screen and hurry off the top of the screen. **GET** scans the keyboard rapidly and

assigns whatever key is pressed to the string A$. If no key is pressed when **GET** tells the computer to look at the keyboard, A$ is assigned a value of " " or nothing. The computer then prints a nothing on the screen and executes **GET** again. When you press a key, its character is assigned to A$ and is printed. It runs off the top of the screen because **GET** is executed continuously and the **PRINT** statement in line 10 causes the computer to move down one line even if A$ is a null string. Press the RUN/STOP key and type in a new line 10:

```
10 PRINT A$;
```

Now **RUN** the program. The ; after A$ tells the computer not to move down a line when the **PRINT** instruction in line 10 is executed. Now the character associated with each key is printed as the key is pressed. What would happen if you replace the ; with a comma?

MATH EXPRESSIONS AND ORDER OF CALCULATION

BASIC uses the regular symbols to indicate addition (+) and subtraction (−). Multiplication is indicated by an asterisk (*) and division by a slash (/). The expression:

```
LET A = 5*4 + 1
```

means "A equals five times four plus one." The expression:

```
LET B = 10/4 − 1
```

means "B equals ten divided by four minus one." The variable A thus equals 21 and B equals 4. There is one more math symbol that is frequently seen in programs. Look at the two examples below:

```
LET R = 5*5*5        LET R = 5↑3
```

Both these expressions equal 125. The first uses a series of multiplication symbols to obtain the answer. The second expression is read as "five raised to the third power" or "five cubed." The symbol for powers is an up arrow which is just above the RETURN key on the right side of the keyboard.

When BASIC is instructed to do a series of math operations, it follows a standard sequence in doing them. All the power computations are done first, then the multiplication and division is finished. Finally, the addition and subtraction is computed. A handy mnemonic to remember the standard sequence is "Please My Dear Aunt Sally" for Power, Multiply, Divide, Add and Subtract. When there are several of the same types of math to be done, the computer will do the work on the left first and then work across to the right.

Sometimes the standard order of computing is not the order needed to solve the problem correctly. It can be changed by the use of parentheses. The computer will do all the computations inside parentheses, regardless of its type, before it does the work outside the parentheses. Consider the two expressions below:

PRINT 6*4 – 3 **PRINT** 6*(4 – 3)

The first expression produces 21 because the multiplication is done first, then the three is subtracted from the result. The second expression produces 6 because the subtraction is done first (4 – 3), then the multiplication (6*1). If an expression has several sets of parentheses, the computer will do everything inside the inner-most set of parentheses first, then work inside the next and so on, and finally do the work outside any parentheses. When using parentheses be sure there is one right parenthesis for every left parenthesis, otherwise the computer will report an error.

Types of Numbers

The BASIC used in the VIC 20 can deal with several types of numbers. Here are examples of the types of numbers it accepts:

5	positive integer or "whole" number
– 5	negative integer or "whole" number
345.55	positive decimal number
– 345.55	negative decimal number
4.33E + 5	positive number in scientific notation
4.33E – 5	negative number in scientific notation

The last two numbers may require some explaining. Scientific notation is a means of writing very large and very small numbers in a way that conserves space. The number 4.33E + 5 is the same as 433000. You make the converson by moving the decimal five places to the right. The number 4.33E-5 is the same as .0000433 because you move the decimal point five places to the left. If the VIC 20 encounters a number so large that it cannot display it in normal fashion, it will convert it to scientific notation. In your programming you can also use scientific notation when it is convenient.

If you use positive or negative integers in a program the variable name could end in a %. That is, if you ask the user to **INPUT** an integer number which will be assigned to the variable A, the line might look like this:

10 **INPUT** "TYPE A NUMBER";A%

Integer variables such as A% cannot handle decimal points; and if you type in 9.9 when line 10 above is executed the computer will ignore the .9. A%

will equal 9. Standard numeric variables such as A can handle whole numbers as well as numbers with decimals in them. Regular numeric variables are often called "floating point" numbers since you can use values that have the decimal point in a variety of locations. In response to **INPUT** A you could type in 9.9, 9.99, or 456.890. The computer can handle any of these with no problem.

Chapter 8

More BASIC

BASIC has several ways of comparing one variable to another and of making decisions. The most important of these are **IF...THEN**, **GOSUB**, and **FOR...NEXT**.

IF...THEN

Remember the explanation of the key word **GET** in the preceding chapter? The example program used several **IF...THEN** statements to make decisions about what to do next. The key words **IF** and **THEN** are used in combination to enable the computer to make many different types of decisions.

IF...THEN is more a family of instructions than just one statement. Here is a typical example of an **IF...THEN** statement:

```
50 IF X<N THEN PRINT X$,X
```

Line 50 above looks at the value of X. **IF** X is less than (<) N, **THEN** the computer will **PRINT** the string X$ and the value of the numeric variable X. The **IF** in line 50 is used to specify a "condition." If that condition is true, the action specified in the **THEN** part is performed. There are several types of **IF...THEN** statements.

The **IF...THEN** statement can involve any combination of the conditions and actions listed, and it can make comparisons that involve values (**IF** 5 = 5), variables (**IF** X = G), strings (**IF** P$ = "YES"), or expressions (**IF** X/B = Z/A). When the condition is true, the computer will proceed across the line and take whatever action is specified. If the condition is false, the computer moves on to the next line in the program without taking the action called for in the **IF...THEN** statement. **IF...THEN** allows a programmer to perform "conditional branching." That is, the computer will do something if and only if certain conditions exist.

IF The Condition	THEN The Action
< less than	**GOTO**
> greater than	**PRINT**
= equals	**LET**
< = less than or equal	**INPUT**
> = greater than or equal	**GOSUB**
< > not equal	**RETURN**
NOT	**STOP**
AND	**CLS**
OR	**PAUSE**
	PLOT
	UNPLOT

Here are some examples of typical **IF . . . THEN** statements that use various combinations of "conditions" and "actions."

IF B > A **THEN PRINT** A—If B is greater than A then print the value of A on the video screen.

IF A$ < > B$ **THEN LET** A$ = " "—If the string A$ is not equal to the string B$ then let A$ equal a null string.

IF 5*45/K < = B/K **THEN INPUT** A—If 5 times 45 divided by K is less than or equal to B divided by K then input the value of A.

Comparisons Involving Strings. Several of the examples above compare strings (e.g., A$ = "YES" or A$ < > B$). Comparisons involving the = sign and the not equal to sign (< >) are relatively easy to understand. The two strings must be exactly the same to meet the equal condition, and a single difference (e.g., CAN NOT versus CANNOT) will cause the not-equal-to condition to be met.

What about comparisons involving the greater than or less than symbols? What does

IF A$ < B$ **THEN GOTO** 120

mean? Suppose A$ equals THESE and B$ equals THIS? Would the action after **THEN** be executed? Logically, you might assume that since there are five letters in A$ and only four in B$, then the condition would not be met since A$ is not less than B$. That, however, is not how the comparison works. When a greater-than or less-than comparison is to be made, the computer does it alphabetically. The letter A is less than the letter B because A comes first in the alphabet. Thus when the comparison of A$

and B$ is made, the computer will find the first two letters of the string to be equal, but the third letter in A$ is E while it is I in B$. E comes before I in the alphabet and that makes A$ less than B$.

The Use of Logical Operators. All of the examples cited above used standard arithmetic operators such as equal-to and greater-than. Most readers are probably familiar with arithmetic operators. There are three "conditional" operators, however, that may not be familiar—NOT, AND, and OR. These are logical operators which come from a branch of mathematics called Boolean algebra after the mathematician Boole who developed the concepts. Here are some examples using AND and OR:

```
IF A$ = "YES" AND A<34 THEN GOTO 134
```

If the string A$ equals YES and, in addition, the numeric variable A is less than 34, then go to line 134 and begin executing the instructions found there.

The AND operator allows you to set up two "conditions" for execution of the instructions after **THEN**. If you use AND it tells the computer that both conditions must be met before the action or instructions after **THEN** are executed.

It also is perfectly acceptable to set up more than two conditions using AND. For example:

```
IF A=1 AND B=2 AND C=3 THEN PRINT "EVERYTHING
CHECKS OUT"
```

If A equals 1, B equals 2, and C equals 3, print the string. In this example all three conditions must be met. If one of the variables has a value other than the one specified after the equal sign, the action after **THEN** will not be taken.

Now let's look at OR. Here is an example:

```
IF A$ = "YES" OR A<35 THEN GOTO 135
```

If the string A$ equals YES or if A<35 then go to line 135 and begin executing the instructions stored there.

The OR operator tells the computer to execute the instructions after THEN if either of the conditions are met. Thus if A$ equals YES and A equals 121, the action will be taken. If A$ equals NO and A equals 34, the action will be taken. If A$ equals YES and A equals 34, the action also will be taken since the condition specified by OR is more than fulfilled. The only way the action after **THEN** will not be taken is if A$ does not equal YES and A is not less than 34.

You might like to type in the program below and **RUN** it several times to see how different values of A and A$ determine what happens:

```
10 INPUT A
20 INPUT A$
30 IF A$ = "YES" OR A<35 THEN GOTO 200
75 PRINT "CONDITION NOT MET"
100 STOP
200 PRINT "CONDITION MET"
```

Then change line 30 to

```
30 IF A$ = "YES" AND A<35 THEN GOTO 200
```

RUN the program several more times using different values for A$ and A. Do you see how AND and OR work? It should be noted that OR can be used with more than two conditions.

```
IF A = 1 OR B = 2 OR C = 3 THEN PRINT "EVERYTHING CHECKS
OUT"
```

If at least one of the conditions (A = 1 or B = 2 or C = 3) is met, the action after **THEN** will be taken.

There is one more twist to the use of logical conditions OR and AND in an **IF...THEN** statement. You can combine AND and OR conditions in one **IF...THEN** statement. Here is an example.

IF A = 1 AND B = 2 OR C = 3 **THEN PRINT** "SOMETHING WORKS"— If A equals 1 and B equals 2, execute the action after **THEN**; if C equals 3 (regardless of what A and B equal), execute the action after **THEN**.

When you use AND and OR together it lets you specify that two or more conditions must be met to get the action (AND) or if one of another set of conditions is met the computer is to take the action (OR) even if the AND specifications were not matched. You can create a variety of very complicated conditions if you need to with AND and OR.

The "NOT" operator is much less interesting or useful than AND and OR. NOT simply negates the conditional:

```
IF NOT A = 1 THEN GOTO 120
```

If A is not equal to 1, then go to line 120 and begin executing the instructions there.

The instruction above is the same as **IF** A< >1 **THEN GOTO** 120. Thus, NOT generally duplicates an instruction that can be given in a different way.

There are several advanced programming concepts which involve the use of logical operators that have not been covered. They are encountered less frequently than the concepts presented here, and they are best approached after you clearly understand the basic uses of logical operators.

FOR...NEXT TO and STEP

You have already used several simple **FOR...NEXT** loops in Chapter 6. The program listed below will be used to illustrate some of the finer points of **FOR...NEXT** loops:

```
10 PRINT "DOLLARS TO INVEST"
15 PRINT "EACH YEAR"
20 INPUT D
30 PRINT "YEARS TO INVEST"
40 INPUT Y
50 PRINT "RATE OF INTEREST"
60 INPUT R
70 LET R = R/100
80 PRINT "YEAR   INVEST   TOTAL"
```

Note: Type in 3 spaces between YEAR and INVEST and 3 spaces between INVEST and TOTAL.

```
 90 LET B = 1 + R
100 PRINT 1;TAB(7);D;TAB(15);D
110 FOR L = 1 TO Y
120 LET B = B*(1 + R)
130 LET 5 = (D*(B - 1))/R
140 IF L < Y THEN PRINT L + 1;TAB(7);(L + 1)*D;TAB(15);INT(5)
200 NEXT L
```

Type in the program exactly as it is listed above. Most of the key words used in the program will be familiar to you.

This program lets you indicate how much money you will invest per year, how many years you will invest that amount, and the interest rate you will receive. The computer takes that information and computes the accumulated investment per year (under INVEST) and the total value (including accumulated interest) of your investment (under TOTAL). The amount under TOTAL is rounded to whole dollars by the INT key word (you'll learn how INT works later in this chapter).

RUN the program using 600 for the Dollars Invested per Year, 10 for the Years to Invest, and 9 for the Rate of Interest. Here is what you should get as a printout:

Table 8.1 A sample run of the Investment Program.

```
DOLLARS TO INVEST
?600
YEARS TO INVEST
10
RATE OF INTEREST
9
YEAR          INVEST        TOTAL
1             600           600
2             1200          1253
3             1800          1966
4             2400          2743
5             3000          3590
6             3600          4514
7             4200          5520
8             4800          6617
9             5400          7812
10            6000          9115
```

Thus, if you invest $600 a year for ten years, you will end up with $9115 of accumulated interest and principal. (This program assumes you do not invest the money until the end of the year; thus no interest is earned during the first year.) Lines 110 to 200 define a LOOP. It is the heart of the program since it computes the amount of money that will be earned and prints the results. It is called a loop because it is used several times, once for each year you plan to invest. The **FOR...NEXT** statement controls how many times the computer moves through the loop. The **NEXT** in line 200 defines the lower boundary of the loop. When the computer comes to a **NEXT**, it returns to the line where **FOR** occurs and goes through the loop again. The way **FOR** operates is a little complicated. In this case, L is the "control" variable in the loop (there is no significance to the label L, it could be any other acceptable variable name). The expression on the other side of the equal sign tells the computer where to start and how many times to run through the loop. Line 110 says "Set L equal to 1 (the initial value for the first loop). Each time the computer runs through the loop, increase the value of L by one. When L is equal to Y (the final value), go through the loop one more time and move on to the line immediately after the **NEXT** statement." Y is the number of years you plan to invest your money, so there will be one line of results for each year you invest. Since in this

program there is no line after line 200 where **NEXT** appears, the computer stops when it finishes the **FOR**. . .**NEXT** loop.

As usual there are a few fine points that have not yet been covered. Sometimes you do not want to increase the control variable by one each time the **FOR**. . .**NEXT** loop is executed. Here is an example:

```
FOR I = 9 TO 14 STEP .25
```

The **STEP** .25 at the end of the line above tells the computer to increase the value of I only .25 each time the loop is executed. Thus, I would equal 9 the first time, 9.25 the second time, 9.50 the third time and so on. If this loop was the beginning of a section of the program that calculated and printed the rate of return on investments that drew different rates of interest, the I variable might be used to designate the interest rate. Thus, the instruction would give data on interest rates from 9 to 14 in steps of .25. The line **FOR** I = 9 **TO** 15 **STEP** 2 would give the same information but data would be provided only on interest rates of 9, 11, 13, and 15. The line **FOR** I = 15 **TO** 9 **STEP** -2 accomplishes the same thing. In this case the data for an interest rate of 15 will be printed first, then 13, then 11, then 9. The **STEP** -2 will cause the computer to subtract 2 from the control variable instead of increasing it each time the loop is executed.

Another common variation for **FOR**. . .**NEXT** loops is to put one inside another. The details of how this works are provided in most books on BASIC. If you are using a program that has "nested loops," remember to be sure the **NEXT** statements are in the proper order. In essence, a loop inside another loop will run through its entire range every time the outside loop runs through one step.

GOSUB RETURN

FOR. . .**NEXT** loops let you use the instructions inside a loop over and over with the number of times they are used determined by the control values provided. This is a very handy procedure. You will find **FOR**-. . .**NEXT** loops used in many programs.

Subroutines. Another handy procedure that is common in BASIC programs is the "subroutine," a set of instructions used at several points in a program.

The example program below will make the concept of a subroutine clearer:

```
10 GOSUB 100
20 PRINT "GUESS MY NUMBER"
25 PRINT
```

```
 30 INPUT G
 40 GOSUB 200
 50 GOTO 20
100 REM RANDOM NUMBER GENERATOR
120 LET N = INT(RND(0)*100)
130 RETURN
200 REM GUESS CHECKING ROUTINE
210 IF G>N THEN PRINT "TOO HIGH TRY AGAIN"
220 IF G<N THEN PRINT "TOO LOW TRY AGAIN"
230 IF G=N THEN GOTO 280
240 GOTO 300
280 PRINT "GREAT, YOU GOT IT!"
285 PRINT "I WILL THINK OF A NEW NUMBER"
290 GOSUB 100
300 RETURN
```

If you **RUN** this program it will print GUESS MY NUMBER at the top of the screen. Line 25 is a **PRINT** instruction with nothing to print. It causes the computer to skip a line each time line 25 is executed. You then type in a number between 0 and 100 and press ENTER. The computer compares your guess with the number it has generated and tells you if your guess is high, low, or correct. If it is high or low, the computer tells you to try again and asks you to make another guess (don't forget to press ENTER). When you guess the number, the computer tells you GREAT, YOU GOT IT! Then it says I WILL THINK OF A NEW NUMBER and asks you to input another guess. To stop playing this game you should hold down the RUN/STOP key and press RESTORE.

The program begins with the instruction **GOSUB** 100 in line 10. The key word **GOSUB** tells the computer to stop executing instructions sequentially. Instead it is to go to a subroutine which begins at the line number which follows **GOSUB**. In this case the subroutine begins on line 100. Actually line 100 contains a **REM**ark that tells us what the subroutine does. If you begin a line with the key word **REM**, the computer will ignore the line but you can type a message after **REM** that provides information to the programmer. The subroutine that begins at line 100 generates a random number. The subroutine includes lines 100 to 130. We know line 130 is the end of the subroutine because it contains the key word **RETURN**, which tells the computer that the subroutine has been executed. The computer then returns to the instruction just after the **GOSUB** key word. In this case, it is line 20. The computer has branched to a subroutine that did a particular task and then returned to continue normal execution of the program. Line 20 prints GUESS MY NUMBER; then line 30 looks for a

number which you will type. When you press ENTER, the computer takes the number you typed and assigns it to the variable named G (for guess).

Now we come to line 40 where there is another **GOSUB**, this time to line 200. The subroutine includes lines 200 through 300. This subroutine compares the number generated by the computer (variable N) with your current guess (variable G) and gives you either a hint or a congratulatory message. If you are incorrect, the **GOTO** in line 240 sends the computer to line 300 which has a **RETURN** statement. The computer returns to the instruction just beyond the **GOSUB** that brought it to the subroutine. That is line 50 where there is a **GOTO** statement which sends the computer back to line 20. The computer tells you GUESS MY NUMBER and looks for another input. When you type in another number, the computer comes to the **GOSUB** 200 instruction in line 40 once more, and things start all over again.

Now what happens if you guess correctly? Line 230 in the subroutine indentifies the guess as a correct one and the instruction **GOTO** 280 is executed. The computer offers praise in 280 and tells you it will think of a new number to guess in line 285. Then line 290 does something interesting. We are already in a subroutine, but there is another **GOSUB** in 290. The **GOSUB** there sends the computer to the random number generator subroutine beginning in line 100. When the computer has a new number for variable N, it returns to the instruction just past the **GOSUB** instruction in 290. That is line 300 which is another **RETURN**. This time the computer goes back to line 50 because line 40 sent it to this subroutine.

There are several advantages to the use of subroutines. You can write all the instructions to do a particular task in one place in the program and then use the **GOSUB** instruction anytime you need that task done. For many people, writing programs is easier if you break the job down into a series of subroutines. (One theory of programming, however, criticizes the use of subroutines. The theory suggests a program should be written so that it executes sequentially, from top to bottom, with no jumping from one place to another.) In addition, when you must do a particular task at several points in the program, it is very easy to put a **GOSUB** at the appropriate place and avoid writing the same set of instructions at several points in the program.

Subscripted Variables

We now come to one of the last major concepts in BASIC, subscripted variables. As noted earlier, variables have names (A, A4, A$, AA, etc.), and they have values. There are two major types of variables—string and numeric, and BASIC has several instructions that work only on either numeric or string variables.

Up to this point, we have dealt with discrete variables with names that are not, as far as the computer is concerned, related. Variables such as A1 and A2 or TEST1 and TEST2 are just variables to the computer; the similarity of the names means nothing to the computer. There are no special instructions, for example, that will let you deal with all the variables that have TEST in their name (e.g., TEST1, TEST2, TEST3 and so on). BASIC does, however, provide a means of dealing with groups of variables. You do it through the use of "subscripted" variables.

Variables with names like A, TEST, and TOTAL3 are just plain, garden-variety variables. Variables with names like A(1), TEST(1), and TOTAL(3) are a different matter. The variable A4 is simply a convenient name. A(4), on the other hand, is a subscripted variable; A is the variable name and 4 is the subscript. If you think of the data in a computer as a parade, then A4 might be one lone clown who marches by. A(4), on the other hand, might be one rider in a row of riders. The rider on the far right of the row of riders might be A(1), the next one would be A(2), the next A(3), and the one on the far left would be A(4). If these riders were data in a computer they would be called a "list" or "one-dimensional array" since they are an associated group arranged along one dimension.

Now suppose a band comes marching by. Instead of one row, a band is made up of a "matrix" or "array" of people arranged in rows and columns. The person in the first row and first column might be given the variable name B(1,1) while a tuba player in the fifth row and eighth column would be B(5,8). To the computer, the "double subscripts" in the variable names for the band represent the column and row placement of the variables. This is a handy procedure when you have to deal with a large number of variables that are related to one another in some systematic way. In BASIC there are ways of dealing with subscripted variables that cannot be used with regular variables. (It is important, however, to keep in mind that a subscripted variable such as A(2) is an entirely different variable from A2 and A$(2). A$(2) is, of course, a string variable with a single subscript.) Here is an example that illustrates the use of subscripted variables:

```
10 PRINT "HOW MANY TOOK THE TEST"
20 INPUT T
25 DIM S(T)
30 FOR X = 1 TO T
40 PRINT "TYPE IN SCORE   ";X;
50 PRINT "   THEN PRESS RETURN"
50 INPUT S(X)
60 NEXT X
65 LET TL = 0
```

```
 70 FOR X = 1 TO T
 80 LET TL = TL + S(X)
 90 NEXT Y
100 LET MN = TL/T
110 PRINT "THE MEAN SCORE ON THE"
120 PRINT "TEST WAS "; MN
```

This little program lets you type in the scores students make on a test. It uses the subscripted variable S(#) to store the scores. The first thing the program does is ask you how many scores there are. You input that as the value for variable T in line 20. If T equals 3, that means there must be room in the computer for three scores which, in the program, will be stored as S(1), S(2), and S(3). Line 25 uses a new key word, **DIM**, to tell the computer that you plan to use a subscripted variable named S later in the program and that it will have three values. (If you tell the computer that there will be 24 scores, then line 25 will set variable S up for 24 different subscripts—S(1) through S(24).)

If you use subscripted variables in a program, you will generally want to use the **DIM** instruction to tell the computer what to expect before the subscripted variable is used. **DIM** is short for **DIM**ension. It tells the computer to set aside a section of memory that will be used to store the values of each element of the subscripted variable. If you do not execute a **DIM** statement before using a subscripted variable, the VIC 20 will set aside space for 11 elements in each dimension (0 through 10). If you use less than that, the only result of not using a **DIM** statement will be that a little memory is set aside for variables that will not be used. If you have more than 11 elements, the result is more serious. You will get an error such as BAD SUBSCRIPT. A **DIM** statement will have to be added to the program to make it work correctly. If you inadvertently try to **DIM** the same subscripted variable twice in a program, however, the computer will balk and give you a REDIM'D ARRAY ERROR. All arrays with more than one dimension must be set up with a **DIM** statement before use.

There is a **FOR**...**NEXT** loop in lines 30 through 60 of the program. The first time through this loop the computer asks you to type in the first test score and it assigns that score as the value of S(1). The second time through the loop it gets the value for S(2), and so on. The other **FOR**...**NEXT** loop in the program is lines 70 through 90. That loop adds up all the scores on the test and puts the total in the variable named TL.

Finally, the computer divides TL by the number of scores and prints out the average or mean score on the test. This program illustrates one of the major advantages of subscripted variables, you can use **FOR**...**NEXT** loops to get the values assigned to the correct variables.

The subscripted variable in the test scoring program is a "one-dimensional" variable. The VIC 20 will let you define or dimension (**DIM**) as many one-dimensional variables as you like, each with whatever number of subscripts needed. **DIM** R(345), for example, sets aside enough memory for 345 variables named R(1) through R(345). There is, however, one limitation. You must have the memory available to set aside. If you don't, the computer will give you an OUT OF MEMORY error. In addition, even if you do have enough memory at the beginning of a program for many **DIM**ension instructions you may not have enough memory to do all that **DIM**ensioning, to store the program, to run the video display,and to execute the program. Keep in mind how much memory you have available as you work with **DIM**.

Remember, also, that **DIM** can be used to set up multiple-dimensional arrays or matrixes as well as one-dimensional subscripted variables. DIM R(2,4) sets aside enough memory for a variable that has 2 rows and 4 columns:

R(1,1)	R(1,2)	R(1,3)	R(1,4)
R(2,1)	R(2,2)	R(2,3)	R(2,4)

Actually, we have not been totally correct in describing how **DIM** works up to this point. **DIM** R(3) sets aside space for four subscripted variables—R(0), R(1), R(2), and R(3). Many programmers ignore the 0 subscript because some versions of BASIC do not accept it and because most of us are not accustomed to using the digit 0 to indicate something other than nothing. If you are working on a program that takes up most of the memory you have available, however, you can save some memory by remembering that subscripted variables begin with 0 rather than one (e.g., R(0) or R(0,0)).

DIM and String Variables. Thus far we have been dealing with numeric subscripted variables. Does it work the same way with string variables? Almost. If you give the instruction **DIM** A$(2), the computer will be ready to accept three different strings which can have one or more characters in them. A$(0) might have the value CATS; A$(1) might be AND; A$(2) might be DOGS. Now if you tell the computer to **PRINT** A$(2) it will print DOGS. What would happen if you told the computer to **PRINT** A$(0)+A$(1)+A$(2)? If these were numeric variables the computer would add up the three numbers and print the sum. With string variables the computer "concatenates" the strings. That means it puts them together. The instruction to "add" the three strings above would produce:

CATSANDDOGS

Adding strings together is thus a very acceptable thing to do in VIC 20 BASIC. You cannot, however, subtract strings using the minus sign.

Functions in VIC 20 BASIC

Another aspect of BASIC to be discussed in this chapter is that of functions. A function is a lot like canned laughter on a TV program. Whenever the producer needs laughter, pushing the right button will produce it. Functions work the same way. If you need the square root of a number, for example, you could write a subroutine that uses several BASIC key words to do the job. There is, however, a function in VIC 20 BASIC that does the job for you. The instruction **LET** M = SQR(A) will take the square root of A and assign that value to variable M. If A equals 81, then M will equal 9. If you type **PRINT** SQR(81) and press RETURN the computer will print the answer 9 on the display.

There are three families of functions, one that works with numeric variables, one for string variables, and a set of general purpose functions. You already have used several numeric functions in example programs (e.g., INT and RND). The functions SPC and TAB were explained in Chapter 6 and PEEK will be explained in the chapter on graphics. Here is a summary of the numeric functions available in VIC 20 BASIC:

SQR (A) —gives the square root of A. A must be a positive number, but can be a variable—**SQR**(A); a number—SQR(56); or an expression such as **SQR**(X/2*3.5).

ABS (A) —gives the absolute value of A. That is, whether A is positive or negative, **ABS** (A) is always positive. If A equals -5, **ABS** (A) will equal +5. If A equals +5, **ABS** (A) will again equal +5. A can be a variable such as **PRINT ABS** (A), a number like **LET** A = **ABS**(-5), or an expression such as LET B = **ABS**(A-4*6).

INT (A) —finds the integer or whole number part of a number. In VIC 20 BASIC, regardless of whether the number is positive or negative, **INT** chops off everything to the right of the decimal. **PRINT INT** (12.3) would produce 12 while **PRINT INT** (12.9) also produces 12. **PRINT INT** (-7.1) and **PRINT INT** (-7.7) both produce 7.

SGN (A) —If A is greater than 0, then **SGN** (A) will equal 1; if A equals 0 then **SGN** (A) will equal 0, and if A is a negative number, then **SGN** (A) will equal -1. The instruction **LET** M = **SGN** (A) will set M equal to 0 if A equals 0, -1 if A is a negative number, and +1 if A is a positive number. **LET** N = **SGN** (B-C+X) will set N equal to 0, +1, or -1 depending on the value of the expression (B-C+X).

RND (A) —This function is used to generate a random number between 0 and 1. The numbers generated will never be 0 or 1. Instead, they will be decimal numbers between 0 and 1 such as .0011291504 or .78868103. You can use other key words to get numbers in any range you need. **LET M = INT(RND(1)*11)**, for example, will set M equal to numbers between 0 and 10. **LET M = (INT(RND(1)*10)) + 1** will set M equal to numbers between 1 and 10.

FRE(A) —The instruction **PRINT FRE(1)** tells the computer to print amount of memory currently available for use. If the number 3670 were printed, that would mean you have 3670 bytes of RAM currently unused. The number in parentheses is called a "dummy variable" since you can use any number and the result will be the same.

POS(A) —The instruction **PRINT POS(1)** tells the computer to print the number of the column where the cursor currently resides. The number in parenthesis is a dummy variable.

Trigometric Functions. VIC 20 BASIC has a set of functions that deal with trigometric values in radians. **PRINT COS** (34), for example, produces -.848570273. Here are the trig functions available:

COS—cosine
ATN—arctangent
SIN—sine
TAN—tangent

The **EXP** function returns the natural antilog of the value which follows **EXP**. **PRINT EXP** (4), for example, produces 54.5981501. The **LOG** function returns the natural log of the value which follows **LOG**. **PRINT LOG**(4), for example, produces 1.38629436.

NOTE: In Chapter 7, the priority sysem used by VIC 20 BASIC was described. The computer, for example, does multiplication before addition unless you have changed the priority order by adding parentheses to math expressions. The math functions described above have the highest priority. Functions are calculated even before multiplication and division.

String Functions

CHR$(#) —To explain this function we must first deal with the character code the computer uses. The computer cannot store the letter A or the key word **PRINT** in memory. It is only capable of storing patterns of electrical charges that represent numbers. The codes used by the VIC 20 are explained in more detail in Chapter 9. The code for D is 41. The **CHR$** function lets you give the computer the code for a character. Type in

PRINT CHR$**(68) and press RETURN. A "D" appears on the screen. Type **PRINT CHR$ (65) and press RETURN. You should get an A. The **CHR$** function lets you use the code numbers normally used internally by the computer. Table 8.1 shows the codes used by the **CHR$** function. Note that there are codes for all the graphics, symbols, letters, numbers, punctuation, colors, cursor control keys, and some instructions. The instruction **PRINT CHR$**(14), for example, will cause the computer to shift from uppercase/graphics mode to upper/lowercase letters.

ASC(X$) —This string function returns the code value of the first character in the string enclosed in parentheses. For example, if A$ equals CAT, then **PRINT ASC**(A$) would produce 67 since 67 is the code for C.

LEN(X$) —This string function lets you determine how many characters (including spaces) there are in a particular string. If A$ equals CAT, then **PRINT LEN** (A$) will produce 3 since there are three letters in CAT.

STR$ and **VAL** . In one of the example programs earlier in this chapter the point was made that you cannot perform math operations on strings nor can you perform string functions on numeric variables. Though generally true, there is a way of getting around these limitations using the **STR$** and **VAL** functions. **STR$** can be used to convert a numeric variable into a string variable, and **VAL** will convert a string variable into a numeric variable. Here is an example that shows how it works:

```
 5 PRINT "INPUT A"
10 INPUT A
15 PRINT "INPUT A$"
20 INPUT A$
30 PRINT 2*A
40 PRINT 2*A$
```

Now **RUN** the program and input the number 48 for variable A and 1234 for the string A$. Your screen should first display **INPUT** A (respond with 48 and press RETURN), then the display will reply **INPUT** A$ (respond with 1234 and press RETURN). Then the computer will multiply A by 2 and print 96 on the screen. When it comes to line 40 it will try to multiply the string A$ by 2. It cannot multiply strings so it prints ?TYPE MISMATCH ERROR IN 40.

Now add the following lines to the program:

```
40 LET B$ = STR$(A)
50 PRINT B$ + "STRING NOW"
60 LET B = VAL A$
70 PRINT B*2
```

Now **RUN** the program again and again give A the value of 48 and A$ the value 1234. As before the computer prints 96 (2 times 48) from line 30. Line 40 converts the numeric variable A (which equals 48) to the string variable B$. Line 50 prints the string B$ along with "STRING NOW." The computer treats B$ as it does any other string, even though B$ was created from a numeric variable.

In line 60 the string A$ which equals 1234 is converted to the numeric variable B. Line 70 multiplies B by 2 and prints the result which is 2468. Using **VAL** and **STR$** you can convert string variables to numeric variables and vice versa.

LEFT$(A$,#) and **RIGHT$(A$,#)** —Suppose the string A$ is the word BUSTED. **PRINT LEFT$**(A$,2) will cause the two leftmost letters (BU) to be printed. The **RIGHT$** function works the same way. The **LEFT$** or **RIGHT$** tells the computer where to start (on the right or left), the string of interest is specified in the parentheses, and the number after the comma indicates how many letters are to be used in the operation. These functions will also work with a "literal" string. **PRINT LEFT$**("BUSTED",2) would cause BU to be printed on the screen.

MID$(A$,#,#) —This function works like the two functions above but the first number in parentheses identifies the point in the string where the computer will begin and the second number tells how many letters are to be used. **PRINT MID$**(A$,2,3) produces UST if A$ is BUSTED—begin with the second letter and print three letters.

Other Functions

USR(X) —This function is used by programmers who write in assembly language. **USR**(15) will cause the computer to look at memory locations 1 and 2. The numbers stored there tell it where a machine language program begins. The computer goes to that location and begins executing the machine language program. It passes the number 15 to the machine language program where it can be used for a variety of purposes.

TI —Each time the VIC 20 is turned on, it starts an internal "clock" which keeps track of how long the computer has been operating. If you tell the computer to **PRINT TI** and the result is 15710 that would mean the computer had been on for 15710 "jiffies." A jiffy is $^1/_{60}$th of a second. Thus **PRINT TI**/60 would tell you how many seconds since the computer was started.

TI$** —This function has two functions. You can set the internal clock with it. **LET TI$ = 000000 resets the timer **TI** to 0 no matter how long the

computer has been on. **PRINT TI$** will cause a six-digit number to be printed on the screen. That number is the "time" in hours, minutes, and seconds. If **PRINT TI$** produces 023445 that would mean it has been 2 hours 34 minutes and 45 seconds since the computer was turned on (or since the timer was reset to 0).

DEF FN and FN (X) —**DEF FN** is short for "define function." This statement lets you create your own function. **FN (X)** lets you use that function in a program. Suppose, for example, that there is a complicated formula your program must calculate at several different points. The formula might be (A*B)/K-45. First you define your custom function with the **DEF FN** instruction:

```
DEF FNA(X)=(A*B)/K-45
```

This will define the calculations to be performed by function A. If you had used **DEF FNAK**, you would be defining function AK. Now when you want the calculation performed, you might use the following:

```
PRINT FNA(3)
```

The X in parentheses could be any numeric variable and the 3 in parentheses could be any number. The function will work the same regardless of the values provided in parentheses. If you wanted to tell the computer to perform the calculations you assigned to function AK and print the result, you would tell it to **PRINT FNAK(3)**.

Two Final BASIC Key Words

There are two BASIC key words that have not been used in any of our example programs. You may, however, encounter them in programs you use:

SYS —Some BASIC programs perform some operations by jumping to a machine language subroutine. Animated graphics, for example, are often too slow when executed from BASIC, but may run several hundred times faster with machine language. **SYS** 3456 would tell the computer to go to a machine language subroutine which begins at memory location 3456. **SYS** works much like the **GOSUB** instruction, except that the computer goes to a subroutine in machine language instead of a subroutine in BASIC. When the machine language subroutine has finished its work, the machine language equivalent of RETURN sends the computer back to the BASIC program at the point just beyond the **SYS** instruction.

ON —This key word is most often used with **GOTO** or **GOSUB**. If the line below where in a program:

ON N GOTO 120, 130, 140, 150

the computer would check to determine the value of N. If N equals 1 the computer will **GOTO** line 120; if N equals 2 the computer will **GOTO** line 130, and so on. **ON** is a way of selecting alternative destinations for a **GOTO** or **GOSUB** instruction. If N equals 0, is a negative number, or is larger than the number of options available after **GOTO** or **GOSUB** (4 in the example above), the GOTO or GOSUB will be ignored and the computer will proceed to the next line of instructions.

WAIT #,#,# —This instruction is rarely used in programs and requires an understanding of Boolean algebra to use properly. Basically **WAIT** is used to tell the computer to halt execution of the program until the memory location specified by the first number after **WAIT** contains a particular number. The second and third number are used, in a round about way, to specify the number the computer seeks. Generally the "memory location" **WAIT** looks at is really a source of input to the computer rather than an actual location in RAM.

Chapter 9

Graphics, Sound (And More BASIC)

This is the final chapter that discusses progamming your VIC 20. It is, however, a chapter that deals with some of the fancier features of the VIC 20 computer. You will learn how to use some of VIC 20's color graphics capabilities and how to program the sound synthesizer. In addition, some BASIC key words will be explained and used in the example programs.

CONTROLLING BACKGROUND, BORDER AND CHARACTER COLORS

The standard VIC 20 screen is surrounded by a dark blue border. The background field where characters are typed is white, and the characters themselves are dark blue. Some people find this pattern pleasing and come to expect it when using their VIC 20. Others would prefer a different pattern of background, border, and character colors.

It is very easy to change the color of the border and the background (called the playfield or display window in some computer manuals). You can select from 8 different border colors, 16 different screen colors, and 8 different colors of characters. The VIC 20's memory contains a special location which controls the color combinations to be used. That location is 36879. Type the following instruction and note the result:

PRINT PEEK (36879)

Your VIC 20 should print 27. The instruction above tells the VIC 20 to look at memory location 36879, determine what number is stored there, and then print that number on the screen. The number "27" is the code that tells the computer to use a dark blue border and a white background. Now type the following:

```
10 INPUT A
20 POKE 36879,A
3O GOTO 10
```

The program above will allow you to type in a number which becomes the value of variable A. Line 20 uses the instruction **POKE**. **POKE** tells the computer to place the value after the comma in the memory location specified just after **POKE**. Thus, if you **RUN** this program and type in 26, the computer will treat this as an instruction to change the border to red and keep the background white. Try that. The VIC 20 Programmer's Reference Manual lists the colors available and its code number.

Type in a few different combinations and see how the color combinations look on the screen. Here is another program that will let you press the RETURN key and view all the possible combinations from 8 to 255:

```
10 FOR A = 8 TO 255
20 POKE 36879,A
35 PRINT "THE COLOR ON THE SCREEN IS FOR COLOR ";A
37 INPUT R
40 NEXT A
```

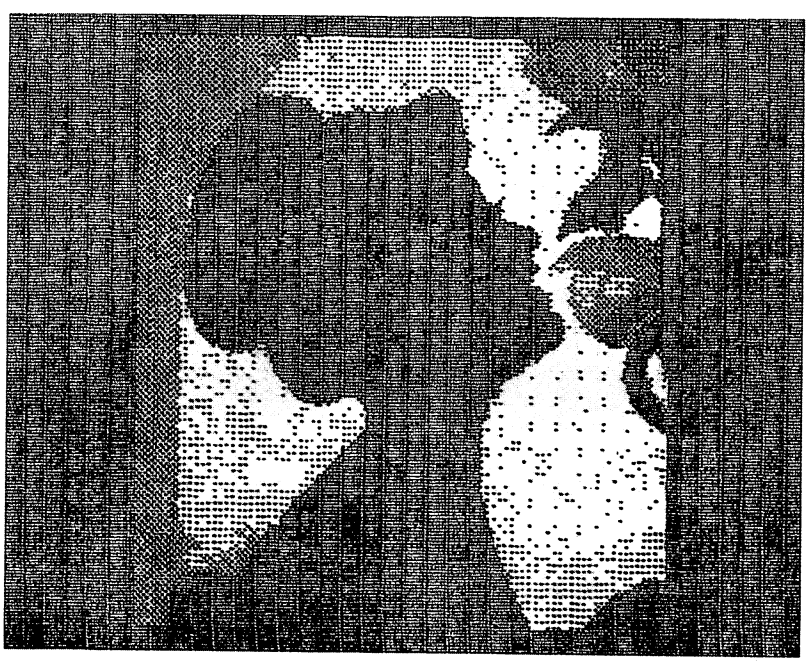


Figure 9.1 VIC PICS.

Line 37 is simply a delaying tactic that causes the computer to stop until you press the RETURN key. You may want to jot down the number of screen combinations that are particularly appealing to you. You may find that some numbers cause the material printed on the screen to "disappear." That is because the color of the characters is the same as the background color.

The VIC 20's ability to change screen colors can be used to create some startling visual effects. Type in the following program (see note about line 10 below):

```
10 PRINT "[CLR]"
20 PRINT "***** ALERT ALERT *****"
30 POKE 36879,42
40 FOR X = 1 TO 100:NEXT X
50 POKE 36879,30
60 FOR X = 1 TO 100:NEXT X
70 GOTO 30
```

Note: Line 10 tells you to type a special sort of print instruction. The CLR in the brackets is obtained by holding down the shift key and pressing the

Figure 9.2 VIC PICS.

CLR/HOME key. For your effort, you will get a small square inside the quotation marks that contains a heart shape (rather than the [CLR] shown above). The heart will be an "inverse" video character to indicate it stands for one of the control keys on the keyboard. Line 10 will cause the computer to clear the screen and move the cursor to the home position. Typing a control key after **PRINT** and inside quotation marks tells the computer to execute an instruction rather than actually print something on the screen. Most of the control keys (e.g., cursor up, down, left, right; backspace, home) can be given as instructions in a program using this format.

If you **RUN** this program, it will print ***** ALERT ALERT ***** on the top of the screen and then alternately change from blue to red. The **FOR...NEXT** loops in lines 40 and 60 are there to slow the computer down. It "counts" to 100 each time it gets to lines 40 and 60. If they were not there, the screen would change colors so quickly the result would be displeasing. Changing the terminal value in the **FOR...NEXT** loop to something other than 100 will change the rate of flashing.

Push the RUN/STOP key to terminate execution of this program and type **POKE** 36879,27 to return the screen to its standard color pattern.

Controlling Character Color

Now let's illustrate another aspect of color on the VIC 20 with another short program:

```
10 PRINT "{BLK}THIS IS IN BLACK{BLU}"
20 PRINT "THIS IS IN BLUE"
30 PRINT "{YEL}THIS IS IN YELLOW"
40 PRINT "{RVS ON BLU}THIS IS IN REFERSE BLUE{RVS OFF}
```

This little program has some odd instructions in it. Line 10 should be interpreted as instructing you to type the number 10 followed by **PRINT**. Then type the quotation mark. The {BLK} should be interpreted as an instruction to hold down the CONTROL key and press the key with BLK embossed on it. That is the 1 key. Now type THIS IS IN BLACK. Finally {BLU} means "hold down the CONTROL key and press the key with BLU embossed on the front" (the 7 key).

Line 30 asks you to type {YEL} which is obtained by holding down CONTROL and pressing the 8 key. Finally, line 40 has {RVS ON BLU}. Hold down CONTROL and press the 9 key, then hold down CONTROL and press the 7 key. At the end of line 40 {RVS OFF} is obtained by holding down CONTROL and pressing the 0 key.

As you type in the CONTROL instructions note that certain graphic symbols are created on the screen. CONTROL-BLK produces a small

square; CONTROL-BLU produces a blue square with a left-arrow in it, and CONTROL – RVS ON produces an inverse PI symbol. You will become accustomed to these symbols and what they designate as you use the VIC 20 graphics instructions. The important point to remember now is that these symbols indicate the computer is to follow a particular instruction when the program is executed. **RUN** the program now and observe the results. There are four lines of text—one with black letters, one with blue letters, and one with yellow letters. The final line of print is in inverse video—that is the letters are surrounded by a dark background. Using the color and RVS ON keys inside quotations after a **PRINT** instruction is one way of controlling how material will be printed on the screen. The {BLK} instruction tells the computer to print characters in black until a new instruction directs otherwise. Characters can be in any of the colors printed on the front of keys 0 through 7 and further variety can be obtained by printing some characters in reverse video via the RVS ON key.

Before moving on to another topic, we should note that the use of control characters inside quotation marks can cause problems under some circumstances. For example, when you are editing or correcting a program, the cursor keys will allow you to move from one point on a line to another and from one line to another. Those same cursor control keys, however, are treated as instructions after a quotation mark. If you type **PRINT** XR + RT when you really meant to type **PRINT** XF + RT you could use the left cursor key to move the cursor back over R and replace it with an F. If you type **PRINT** "THIS AS when you meant to type **PRINT** "THIS IS" the cursor key, if pressed before the end quotation marks, will be treated as an effort to type an instruction into the program. The left-arrow shows up as a rectangle with a narrow line down the left side. Each time you press the left-arrow cursor key, the computer will print one of those symbols. When the program is executed, the symbol is interpreted as an instruction to move the cursor back one space. That means you can program cursor movement by typing cursor control graphic symbols inside quotation marks, but you will not always be able to use the cursor control keys to actually move about inside a string to make corrections. If you must make corrections inside a string the DEL key will work as expected and pressing RETURN will move you off the line where the quotation marks are so you can retype the line correctly. In the example **PRINT** "THIS AS, a correction can be made by adding the second quotation mark—**PRINT** "THIS AS". Now the cursor is outside the quotation marks. You can use the arrow keys to move back inside the marks without adding unwanted symbols.

The points made above can be illustrated with this line:

```
10 PRINT "THIS IS{CRSR DOWN CRSR DOWN}A TEST"
```

After typing THIS IS press the CURSOR DOWN key twice, you should see two inverse-Q's on the screen. Then type A TEST". CLEAR the screen and **RUN** this program. The result should be:

```
THIS IS
        A TEST
```

The computer treated the two inverse-Q's as instructions to move down two lines on the screen. Now **LIST** the program. Use your cursor keys to move the cursor up inside the quotation marks of line 10. At this point the cursor keys let you move freely anywhere on the line. You can edit the string "THIS IS A TEST" any way you want.

Screen Memory and Character Codes

Up to this point, we have used variations of the **PRINT** instruction to tell VIC 20 to display material on its video display. There is another somewhat more complicated way to get material displayed on the screen. The VIC 20 derives its name from one of the integrated circuits it uses. The VIC or video interface chip is a small integrated circuit that controls this computer's color display facilities and its sound generation capability.

The computer can put 22 lines of 23 characters on the screen at one time. That is a total of 506 characters. Each of those 506 locations on the screen (22 lines by 23 characters) has its own byte of memory. The number stored in that location determines what the VIC 20 displays on the screen. The memory location, for example, that controls what is displayed in the top left hand corner of the screen is memory location 7680. Clear the screen (hold down the Shift key and press the CLR/HOME key) and type in the following instruction:

```
POKE 7680,65
```

When you press the RETURN key, the computer should display a graphic symbol in the shape of a spade in the top left corner of the screen. Memory location 7680 controls that screen location and the number 65 is the code for a spade graphics symbol. You can display a symbol at any location on the screen by poking the proper screen memory location with the code for the character you want to display.

Table 9.1 shows the codes for each character the VIC 20 can display. There are two characters associated with each code. The VIC 20 has the patterns for two different sets of characters stored in ROM memory. One set produces capital letters and graphics symbols (Set 1) while the other set foregoes graphics symbols for upper and lower case letters (Set 2). The computer automatically selects the capitals/graphics set when it is switched on. You can select the other set in two ways. If you hold down the Commodore Logo key and press the SHIFT key, the computer will change

from one character set to the other. Do it again and it will flip to the other set. Memory location 36869 actually controls character set selection. If 36869 contains the number 240, the VIC 20 will use the upper/lowercase set. Thus, another way to change between character sets is **POKE** 36869,#. If # is 240 you get upper case letters and graphics, if it is 242 you get upper and lowercase letters.

How does the computer "know" that the code number 65 represents a graphic symbol in the shape of a spade? The VIC 20 computer has a special area of ROM memory that contains patterns for video characters. That part of memory begins at memory location 32768 and continues to location 36863. These locations in ROM contain codes for the video display characters. The pattern for each character is stored in eight bytes of memory. The characters on the video display are actually patterns of dots that create the desired letter or character. The dots for one character are in a pattern of 8 lines of 8 dots. Every character the computer can display has a dot matrix pattern in this section of ROM. One byte of memory is made up of eight bits. A bit of computer memory can store one piece of data, and that data

Table 9.1 VIC 20 screen codes.

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
@		0	U	u	21	*		42
A	a	1	V	v	22	+		43
B	b	2	W	w	23	,		44
C	c	3	X	x	24	—		45
D	d	4	Y	y	25	.		46
E	e	5	Z	z	26	/		47
F	f	6	[		27	ø		48
G	g	7	£		28	1		49
H	h	8	]		29	2		50
I	i	9	↑		30	3		51
J	j	10	←		31	4		52
K	k	11	SPACE		32	5		53
L	l	12	!		33	6		54
M	m	13	"		34	7		55
N	n	14	#		35	8		56
O	o	15	$		36	9		57
P	p	16	%		37	:		58
Q	q	17	&		38	;		59
R	r	18	'		39	<		60
S	s	19	(		40	=		61
T	t	20	)		41	>		62

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
?		63		T	84			106
		64		U	85			107
	A	65		V	86			108
	B	66		W	87			109
	C	67		X	88			110
	D	68		Y	89			111
	E	69		Z	90			112
	F	70			91			113
	G	71			92			114
	H	72			93			115
	I	73			94			116
	J	74			95			117
	K	75	SPACE		96			118
	L	76			97			119
	M	77			98			120
	N	78			99			121
	O	79			100		✓	122
	P	80			101			123
	Q	81			102			124
	R	82			103			125
	S	83			104			126
					105			127

can be either 1 or 0. To be completely accurate, what we call ones and zeros are really the presence of an electrical signal (On equals 1) or the absence of a signal (Off equals 0). Dot patterns for the video characters are stored in memory as eight bytes of data with each byte indicating the pattern of dots to be displayed on one of the eight lines in the dot matrix. When you tell the computer to display a particular character on the screen, it looks in this section of memory to determine just what that character should look like on the screen.

To this point two important sections of memory have been discussed—the area where patterns for each character are stored and the area that controls the character to be displayed on the screen. There is one more area which should be discussed. The VIC 20 not only lets you control what goes in a particular screen location, but also lets you decide what color the character should be. The memory location that controls the color of anything displayed at a particular location is determined by adding 30720 to the screen location number shown in Table 9.2. For example, the top left corner of the screen is memory location 7680. Location 7680 + 30720

or 37400 will control the color of the character in the top left corner of the screen. You can **POKE** a number from 0 to 7 in 37400 and control the color:

0 Black 1 White 2 Red
3 Cyan 4 Purple 5 Green
6 Blue 7 Yellow

Here is a short program that will illustrate how all this works:

```
 5 INPUT A
20 PRINT "{CLR}"
30 POKE 7933,83
40 POKE 7933 + 30720,A
50 GOTO 5
```

Table 9.2 Memory locations for the VIC 20 screen.

Line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	7680	7681	7682	7683	7684	7685	7686	7687	7688	7689	7690	7691	7692	7693	7694	7695	7696	7697	7698	7699	7700	7701
1	7702	7703	7704	7705	7706	7707	7708	7709	7710	7711	7712	7713	7714	7715	7716	7717	7718	7719	7720	7721	7722	7723
2	7724	7725	7726	7727	7728	7729	7730	7731	7732	7733	7734	7735	7736	7737	7738	7739	7740	7741	7742	7743	7744	7745
3	7746	7747	7748	7749	7750	7751	7752	7753	7754	7755	7756	7757	7758	7759	7760	7761	7762	7763	7764	7765	7766	7767
4	7768	7769	7770	7771	7772	7773	7774	7775	7776	7777	7778	7779	7780	7781	7782	7783	7784	7785	7786	7787	7788	7789
5	7790	7791	7792	7793	7794	7795	7796	7797	7798	7799	7800	7801	7802	7803	7804	7805	7806	7807	7808	7809	7810	7811
6	7812	7813	7814	7815	7816	7817	7818	7819	7820	7821	7822	7823	7824	7825	7826	7827	7828	7829	7830	7831	7832	7833
7	7834	7835	7836	7837	7838	7839	7840	7841	7842	7843	7844	7845	7846	7847	7848	7849	7850	7851	7852	7853	7854	7855
8	7856	7857	7858	7859	7860	7861	7862	7863	7864	7865	7866	7867	7868	7869	7870	7871	7872	7873	7874	7875	7876	7877
9	7878	7879	7880	7881	7882	7883	7884	7885	7886	7887	7888	7889	7890	7891	7892	7893	7894	7895	7896	7897	7898	7899
10	7900	7901	7902	7903	7904	7905	7906	7907	7908	7909	7910	7911	7912	7913	7914	7915	7916	7917	7918	7919	7920	7921
11	7922	7923	7924	7925	7926	7927	7928	7929	7930	7931	7932	7933	7934	7935	7936	7937	7938	7939	7940	7941	7942	7943
12	7944	7945	7946	7947	7948	7949	7950	7951	7952	7953	7954	7955	7956	7957	7958	7959	7960	7961	7962	7963	7964	7965
13	7966	7967	7968	7969	7970	7971	7972	7973	7974	7975	7976	7977	7978	7979	7980	7981	7982	7983	7984	7985	7986	7987
14	7988	7989	7990	7991	7992	7993	7994	7995	7996	7997	7998	7999	8000	8001	8002	8003	8004	8005	8006	8007	8008	8009
15	8010	8011	8012	8013	8014	8015	8016	8017	8018	8019	8020	8021	8022	8023	8024	8025	8026	8027	8028	8029	8030	8031
16	8032	8033	8034	8035	8036	8037	8038	8039	8040	8041	8042	8043	8044	8045	8046	8047	8048	8049	8050	8051	8052	8053
17	8054	8055	8056	8057	8058	8059	8060	8061	8062	8063	8064	8065	8066	8067	8068	8069	8070	8071	8072	8073	8074	8075
18	8076	8077	8078	8079	8080	8081	8082	8083	8084	8085	8086	8087	8088	8089	8090	8091	8092	8093	8094	8095	8096	8097
19	8098	8099	8100	8101	8102	8103	8104	8105	8106	8107	8108	8109	8110	8111	8112	8113	8114	8115	8116	8117	8118	8119
20	8120	8121	8122	8123	8124	8125	8126	8127	8128	8129	8130	8131	8132	8133	8134	8135	8136	8137	8138	8139	8140	8141
21	8142	8143	8144	8145	8146	8147	8148	8149	8150	8151	8152	8153	8154	8155	8156	8157	8158	8159	8160	8161	8162	8163
22	8164	8165	8166	8167	8168	8169	8170	8171	8172	8173	8174	8175	8176	8177	8178	8179	8180	8181	8182	8183	8184	8185

This program will display the graphic character that resembles a heart in the middle of the screen. The number 83 in line 30 tells the computer which number is to be poked into memory location 7933. Table 9.2 tells us that 7933 controls the eleventh character position in the eleventh line on the screen. Table 9.1 tells us that code 83 will produce a heart when the computer is using the #1 character set.

Line 30 thus puts the character code in the screen memory location. Line 40 tells the computer to poke the value of variable A into the memory location that controls the color of character location 11 in line 11. When this program is run the computer asks for input for the value of A. Type in 0 and press the RETURN key. Then line 20 clears the screen. Line 30 pokes the character code into screen memory and line 40 pokes the number of the color for the character. Zero is the code for Black, so you should see a black heart in the middle of the screen. Line 50 sends the program back to line 5 where you can input another value for A. Use any number between 0 and 7 to see the color change. Note that since the screen color is white, giving A the value of 1 (white) will make the character "disappear." It is now a white character on a white background.

Thus far, we have used the character patterns noted in the VIC 20's ROM. It is also possible to design your own set of characters. You could, for example, create a set of Chinese, Greek, or Russian characters and assign them to various keys on the keyboard. Or you could design various characters that would be used in a video game you write. Creating a custom set of characters is a rather complicated task that is beyond the scope of this book. There are, however, several sources of information on this topic:

"VIC 20 Programmers Reference Guide." This book, published by Howard W. Sams and Commodore, contains information you may need to create your own character sets.

"A New Character Set for the VIC 20." This article, by Mike Bassman, was published in the August, 1982 issue of *Micro*, a popular computer magazine that frequently publishes articles on the VIC 20 computer.

"Pixelator." This article, by James Calloway, was published in the October, 1982, issue of *Compute!*, another excellent magazine with many VIC 20-related articles. Calloway explains the creation of custom characters and provides a set of programs that will help you design and use custom characters.

"A Day at the Races" by Robert Ferree and **"UFO Pilot: VIC 20 Custom Characters for Game Graphics"** by Bud Banis. Both of these articles appeared in the February, 1983 issue of *Compute!* Each contains a listing

of a very good video game that makes use of the VIC 20's ability to use custom-designed characters. One creates an animated horse race on the screen; the other lets you pilot an "unidentified flying object" on a journey filled with danger.

Grafix Designer. This $15 program comes on cassette from Midwest Micro Associates (P.O. Box 6148, Kansas City, Missouri 64110, phone 816-254-9600). This program makes the task of designing custom characters easy.

It draws a large 8 by 8 matrix on the screen and lets you tell it whether a "dot" will appear in each of the 64 locations on the grid. The program lets you create an entire set of characters for games or foreign language character fonts. There are several convenient ways of editing the characters and even a way of changing the orientation of the character. For example, if you were creating a "tank" figure to turn on the screen, Grafix Designer will let you design the character for the tank once. Then you can tell the program to "rotate" the character that represents the tank so that it faces a different direction. You might have one character for a tank that faces West, one that faces East, one South, and one North. This program makes an otherwise complicated task easy and we recommend it for anyone who wants to work with custom characters on the VIC 20.

High Resolution Graphics on the VIC 20

Each character on the screen is actually made up of an 8 by 8 matrix of dots. If you use the built-in graphics characters on the VIC 20 keyboard, it is possible to create some interesting graphics figures on the screen. Holding down the shift key causes the VIC 20 to print the graphic character on the right front of the key. Holding down the Commodore logo key causes the VIC 20 to print the graphic character on the left front of the key.

These characters are all made up of an 8 by 8 matrix of dots. There are ways to control each of those dots individually rather than as a set of 64 (8 by 8 character matrix) dots. This is usually referred to as high-resolution graphics since you can create a much finer grained display if you can control each dot individually.

There are, however, two problems with high-resolution graphics on the VIC 20. First, most methods use up large amounts of memory and the standard VIC 20 has only 5K of RAM. Individuals who plan to get into high-resolution graphics will need to consider buying extra memory. In addition, the standard VIC 20 BASIC does not have built-in key words that make high-resolution graphics easy to use. Most of the work must be done by poking screen locations. It is hard work on the VIC 20 although other inexpensive computers such as the ATARI 800 and 400 have some built-in

key words in their BASIC that make high-resolution graphics less difficult to accomplish.

Commodore has solved both the problems noted above with a product they call the "Super Expander" cartridge. This $70 cartridge gives your VIC 20 an extra 3K of RAM memory and it adds several new key words to BASIC. These new key words make it easier to use color graphics and sound. Here are some of the key words available:

GRAPHIC —Allows the programmer to set up the screen for a variety of graphics modes. GRAPHIC 2, for example, creates a graphic screen with 160 individually controllable dots on each of 160 lines.

COLOR —Lets you set the screen, border and character colors without poking numbers in several memory locations.

DRAW —Lets you draw a line from one location on the screen to another.

CIRCLE —This key word can be used to draw a circle, ellipse or arc at any location on the screen.

PAINT —You can "fill in" or add color to a section of the screen such as a circle.

CHAR —Puts a graphic symbol on the screen at the location you specify.

If you plan on using the VIC 20 to create high-resolution graphics, we would strongly recommend you consider purchasing the Super Expander cartridge.

An alternative to Super Expander is a program called VIC-PICS, a $20 program that comes on cassette from Midwest Micro Associates (address provided above). This program has a set of remarkable pictures that were created by converting the images from a television camera into a pattern of codes the VIC 20 uses to create screen displays. VIC-PICS has everything from pictures of a pixie to American Eagles and mountain landscapes. You can even tell your VIC 20 to print copies of these pictures on a compatible printer. In addition, VIC-PICS contains a program that lets you draw high-resolution color pictures yourself (or modify the ones provided) using the keyboard and a joystick. These can also be printed or saved on a diskette or cassette. For the price, VIC-PICS provides good value. Midwest Micro Associates also markets "Grafix Managerie" a $15 program that has a number of demonstration displays that show you what the VIC 20 can do with high-resolution color graphics. There are screens of arcade critters, a musical staff, and custom character fonts. Grafix Managerie is not a

program that lets you create your own graphics. It simply demonstrates what can be done by an experienced programmer.

SOUND ON THE VIC 20

The final topic of this chapter is sound synthesis, a strong point of the VIC 20. As noted earlier, the Super Expander cartridge adds a few key words that make it easier to create all sorts of sounds on the VIC 20. The task of creating those sounds on the standard, unadorned VIC 20 is not all that difficult, however.

As you might guess, the VIC 20 uses the key word **POKE** to create sounds. There are five important memory locations that deal with sound generation:

36878 —This location controls the volume of the sound being generated. The values poked into 36878 can be anything from 0 (no sound) to 15 (highest volume). Any number outside that range also results in the elminiation of sound. **POKE** 36878,7 would cause sounds to be generated at moderate volume. You can also control the volume of the sounds by adjusting the volume on your television or video monitor.

36874, 36875, 36876 —These three memory locations are all capable of generating musical notes. Each can be poked with a number from 128 to 255. The higher the number the higher the pitch of the sound.

There is one more voice the VIC 20 can use to create sound. It is memory location 36877. The sound created here is not really musical. It is more like some of the noises used to add sound effects to video games. That is the most common application of this voice. It creates a variety of sound affects.

Figure 9.3 shows, approximately, the number you must poke into one of the sound memory locations in order to get a particular musical note. The VIC 20 has three voices: a low or alto voice, a middle or tenor voice, and a high or soprano voice. Table 9.4 shows how the sounds produced by the three musical voices in the VIC 20 overlap. Note that each voice can produce sounds across three octaves, but they are not the same octaves. Thus 238 will produce an A# note in the lowest octave the VIC 20 can reproduce if the number is poked into memory location 36874. That same number will produce an A# in the second octave if it is in memory location 36875 or an A# in the third octave if it is in 36876.

The availability of three different voices with different ranges means you can actually cover five octaves of sound. It also means you need not limit your musical compositions to those that play one note at a time. It is

possible to play three different notes simultaneously and independently control when each note begins, how long it lasts, and when it ends.

Music synthesis on the VIC 20 involves controlling four variables—the volume of the sounds produced, the note to be played, and the duration a note is played. Volume is controlled by memory location 36878; notes signified by numbers from 128 to 255 can be produced by memory locations 36874 through 36876, and duration is often controlled by a "do nothing" **FOR...NEXT** loop.

Music synthesis on the VIC 20 is typically used in one of two ways. You may write a program that actually plays a song or you can use a program that lets you generate musical notes by pressing keys on the keyboard. Henry Forson published a program called "VIC Harmony" in the November, 1982 issue of *Compute!* magazine that lets you designate which voice you will use (Soprano, Alto, or Tenor), which note to play (A through G or R which stands for rest or silence), and the duration of each note. You can type in your musical instructions and "VIC Harmony" will play for you. You can keyboard this program from the listing in the magazine or send $3 and a cassette tape mailer to Henry Forson at 5 Ward Place, New Monmouth, NJ 0778. He will mail you a copy of the program on cassette.

Table 9.3 Musical notes and their numeric values.

Note	Value	Note	Value
C	135	G	215
C#	143	G#	217
D	147	A	219
D#	151	A#	221
E	159	B	223
F	163	C	225
F#	167	C#	227
G	175	D	228
G#	179	D#	229
A	183	E	231
A#	187	F	232
B	191	F#	233
C	195	G	235
C#	199	G#	236
D	201	A	237
D#	203	A#	238
E	207	B	239
F	209	C	240
F#	212	C#	241

Table 9.4 Voice comparison.

S3 (36876)	S2 (36875)	S1 (36874)
D O	D 0	D O
E C	E C	E C
F T	F T	F T
G A	G A	G A
A V	A V	A V
B E	B E	B E
C 5	C 4	C 3
D O	D O	D O
E C	E C	E C
F T	F T	F T
G A	G A	G A
A V	A V	A V
B E	B E	B E
C 4	C 3	C 2
D O	D O	D O
E C	E C	E C
F T	F T	F T
G A	G A	G A
A V	A V	A V
B E	B E	B E
C 3	C 2	C 1

HIGHEST OCTAVE

LOWEST OCTAVE

If you would like to begin using the VIC 20 as a keyboard music generator, the program listed in Table 9.5 will help you do that. This program is one of more than 20 on software diskettes available from Micro Computer Applications (1517 W. Church, Marshalltown, Iowa 50158, phone 515-752-8845). The program was produced in the Netherlands by members of the PET Benelux Exchange. For information on this and other public domain programs, see Chapter 10.

This program generates a musical sound each time you press a key. The note continues to be played until you press another key. Pressing 0 produces silence. We will not provide a detailed explanation of each line of the program. You should note, however, that it sets variable SS equal to 36874 in line 1. Line 2 pokes numbers into the memory location whose address is SS or 36874, and into SS + 1, SS + 2, SS + 3, and SS + 4. Those are the four music "voices" and the volume. Volume is set to its maximum value of 15 by line 2 and the four voices are set to silence (0).

Line 10 uses the **GET$** instruction to assign the key you press to the string A$. If A$ equals a null string (" ") the computer simply re-executes line 10 again. If you press a key, the computer makes A$ equal to the key you pressed and proceeds down the program. Suppose you press T. The computer will match A$ with T in line 31. Then it will set the variable A equal to 234 and it will go to line 50. At line 50 the alto voice (36874) is silenced by poking a zero. The **FOR. . .NEXT** loop with a terminal value of 50 keeps that voice silent for a fraction of a second. Then in line 51 the alto voice is poked with the value of A. Since you pressed T this time A has a value of 234 and you will hear a relatively high musical tone for this voice. As soon as the alto voice is poked with the value of A, the program moves back to line 10 and waits for you to press another key. All the letter keys except O, P, and L are used in this program as well as the comma and the numeral 0. You could change SS in line 1 so that it stands for the location of one of the other voices and also change the range of octaves generated by this program. See Chapter 8 for information on this and other public domain software.

If you would like to pursue more programming, see the last chapter of this book for additional sources of information and the next chapter on software for programs that will make learning to program easier.

Table 9.5 A program to turn the VIC 20 into a piano.

```
 1 SS = 36874
 2 POKESS,0:POKESS + 1,0:POKESS + 2,0:POKESS + 3,0:
   POKESS + 4,15
 5 PRINT"(CLR)";"PRESS A KEY"
 6 PRINT"FOR A TONE – "
 7 PRINT"0 TO STOP TONE"
10 GETA$:IFA$ = "  "THEN10
20 IFA$ = "A"THENA = 191:GOTO50
21 IFA$ = "S"THENA = 198:GOTO50
22 IFA$ = "D"THENA = 204:GOTO50
23 IFA$ = "F"THENA = 207:GOTO50
24 IFA$ = "G"THENA = 213:GOTO50
25 IFA$ = "H"THENA = 217:GOTO50
26 IFA$ = "J"THENA = 221:GOTO50
27 IFA$  = "Q"THENA = 223:GOTO50
28 IFA$ = "W"THENA = 227:GOTO50
29 IFA$ = "E"THENA = 230:GOTO50
30 IFA$ = "R"THENA = 231:GOTO50
31 IFA$ = "T"THENA = 234:GOTO50
32 IFA$ = "Y"THENA = 236:GOTO50
33 IFA$ = "U"THENA = 238:GOTO50
34 IFA$ = "I"THENA = 239:GOTO50
35 IFA$ = "O"THENA = 0:GOTO50
36 IFA$ = "K"THENA = 223:GOTO50
37 IFA$ = ","THENA = 191:GOTO50
38 IFA$ = "M"THENA = 187:GOTO50
39 IFA$ = "N"THENA = 179:GOTO50
40 IFA$ = "B"THENA = 170:GOTO50
41 IFA$ = "V"THENA = 159:GOTO50
42 IFA$ = "C"THENA = 153:GOTO50
43 IFA$ = "X"THENA = 141:GOTO50
44 IFA$ = "Z"THENA = 128:GOTO50
49 GOTO70
50 POKESS,0:FORI = 1TO50:NEXT
51 POKESS,A
60 GOTO 10
70 IFA$ = "I"THENA = 240:GOTO50
71 IFA$ = "U"THENA = 239:GOTO50
72 IFA$ = "Y"THENA = 237:GOTO50
```

```
 73 IFA$ = "T"THENA = 235:GOTO50
 74 IFA$ = "R"THENA = 232:GOTO50
 75 IFA$ = "E"THENA = 231:GOTO50
 76 IFA$ = "W"THENA = 228:GOTO50
 77 IFA$ = "Q"THENA = 225:GOTO50
 78 IFA$ = "K"THENA = 225:GOTO50
 79 IFA$ = "J"THENA = 223:GOTO50
 80 IFA$ = "H"THEN = 219:GOTO50
 81 IFA$ = "G"THENA = 215:GOTO50
 82 IFA$ = "F"THENA = 210:GOTO50
 83 IFA$ = "D"THEN = 207:GOTO50
 84 IFA$ = "S"THENA = 201:GOTO50
 85 IFA$ = "A"THENA = 195:GOTO50
 90 GOTO10
100 REM ADOPTED BY
110 REM PET BENELUX
120 REM EXCHANGE
130 REM NETHERLANDS

READY.
```

Table 9.6 A program to play "Frere Jacques."

```
100 DIMA(8)
110 POKE 36878,15
120 FOR A = 5 TO 0 STEP -1
130 T = TI + S
140 IF TI < T GOTO 140
150 READ S,A(A + 0),A(A + 1),A(A + 2),A(A + 3)
160 POKE 36874,A(3):POKE 36875,A(4):POKE 36876,A(5)
170 IF S < > 0 GOTO 130
180 RESTORE:NEXT A
190 POKE 36878,0:END
1000 DATA 10,195,207,215,195
1010 DATA 10,195,207,219,195
1020 DATA 10,201,209,215,175
1030 DATA 10,201,209,209,175
1040 DATA 20,207,215,207,195
1050 DATA 20,195,215,195,0
1060 DATA 10,195,207,215,195
1070 DATA 10,195,207,219,195
1080 DATA 10,201,209,215,175
1090 DATA 10,201,209,209,175
1100 DATA 20,207,215,207,195
1110 DATA 20,195,215,195,195
1120 DATA 0,0,0,0,0

READY.
```

Chapter 10

Software

Since the VIC 20 was announced in 1980, many companies have written software for it. Some of that software was mentioned briefly in Chapter 1. In this chapter, we will provide additional information on a few of the more widely distributed programs. We have personally used over 200 pieces of software for the VIC 20 and our comments are included in the descriptions. In the two months we spent writing this section, at least 75 new products for the VIC 20 were announced. Much of the software reviewed in this chapter will still be available if you are reading this several years after it was written, but many new products will have appeared by then. And to complicate things further, many popular programs are revised regularly. We may find problems with software that are corrected in subsequent revisions.

Finally, it would take a very large book to comprehensively review all the VIC 20 software. This is a selective review. Some software was so bad we didn't think it was worth the time to describe it; some was simply not available for review, and some software was so specialized that we did not include it in this introductory book.

We have divided this chapter into several sections. The first deals with recreational software for the VIC 20. The Business and Home Software section features word processing, accounting, spreadsheets, and "other" software. The Language and Programming Aids section reviews software that lets your VIC 20 speak another language (e.g., FORTH, PILOT) and programs that make it easier to write programs on the VIC 20. The third section is on Educational Software, and the final section is on Utility and Telecommunications Software.

Public Domain Software

Before reviewing recreational software, however, there is one type of software that deserves special attention. Most of the programs reviewed

here are commercial products which were developed and marketed to make a profit. There is certainly nothing wrong with that. Another type of software, public domain software, is also available in reasonable quantity for the VIC 20. Several groups have gathered together a large number of programs written by VIC 20 owners and users groups. The creators of public domain programs have donated them to the public. That is, the creators do not expect to be paid for the software. Generally, you need only pay a small handling and media fee to get copies of the software. A typical fee would be $10 for a diskette containing 10 to 20 programs. The quality of public domain programs varies from terrible to great, but that's the same range you find in commercial software. The major difference between public domain and commercial software is the general lack of documentation with public domain programs. You will often find it necessary to spend some time figuring out how the program works while commercial software generally comes with instructions. There is some public domain software available in every category reviewed in this chapter. Many of the larger PET and VIC 20 users groups have large libraries of public domain software which is free to members of the group. The names and addresses of several sources of public domain software for the VIC 20 are listed below:

Public Domain (5025 South Rangeline Road, West Milton, Ohio 45383, phone 513-698-5638). This company collects public domain software for the PET, VIC 20, and Commodore 64 computers. Programs are available on either cassette or diskette. A diskette of 60 to 80 programs costs $10. Two cassettes of software (over 50 programs) are also $10.

Toronto PET Users Group (TORPET, P.O. Box 100, Station "5", Toronto, Ontario Canada M5M 4L6). This organization is one of the most active user groups in the world. Membership in TORPET is only $20 a year. You get monthly issues of their excellent magazine, *THE TORPET*, and access to over 3,000 public domain programs from the club's library. The group regularly publishes descriptions of the programs available in *THE TORPET*; prices are $10 a diskette to members. We recommend this organization highly.

Microcomputer Applications (1517 W. Church, Marshalltown, Iowa 50158, phone 515-752-8845). Harold Montover is the owner of this company, an active Commodore dealership with special emphasis on the school market. Montover has collected thousands of public domain programs for Commodore computers which he distributes for $10 a diskette. His collection includes hundreds of educational programs and some utility software

as well. The company is friendly, cooperative, and has one of the largest collections of public domain software in the U.S.

Recreational Software

Many VIC 20 owners buy and use the computer as if it were a sophisticated video-game machine. With an under $100 price tag, a cartridge slot, color and sound, and a connector for joysticks and game paddles, the VIC 20 is indeed a viable video-game machine.

You can buy games in three different formats—as cartridges that plug into the cartridge slot, as programs on diskettes, and as programs on cassettes. Cassette programs are generally cheaper, but are less convenient to use. Programs on diskette are easier and faster to use, but require you to purchase a disk drive to use them. Diskettes are also somewhat fragile and require careful handling.

Cartridges are the most convenient form for recreational software and are the most expensive. Cartridge games generally sell for between $25 and $55.

Video Vermin (UMI, 3503-C Temple Avenue, Pomona, California 91768, phone 714-594-1351, cartridge—$50). Similar to Centipede, this game has good graphics and fast action. You are a gardener who must use your joystick to aim and shoot at all sorts of pests who invade the garden.

Choplifter (Creative Software, 230 Caribbean Drive, Sunnyvale, California 94086, cartridge—$40). You are the pilot of a helicopter that must fly behind enemy lines to pick up hostages who are under attack. You must land, avoid enemy fire, let the hostages climb aboard, and return to base. Everyone cannot be taken in one trip so you may have to leave some behind to be killed by the enemy. They wave as you fly away. This is technically an excellent game, one of the best we reviewed. Some people may not feel comfortable with the premise because it is too close to the reality of Vietnam and Iran.

Serpentine (Creative Software, cartridge—$40). Like the Apple version, this program puts you in a maze with "snakes" of various colors. You must avoid some, attack some, and handle several types of situations that come up as you play. It has very good graphics and enough complications and variations to hold your interest.

Trashman (Creative Software, cartridge—$40). This program puts you at the wheel of a garbage truck that must travel around a maze collecting

trash. Players must avoid big flies that menace the truck. Trashman is clearly PAC-MAN with a different underlying premise. It plays in virtually the same fashion as PAC-MAN.

Apple Panic (Creative Software, cartridge—$40). Apple Panic is very similar to Donkey Kong, the popular arcade game. You are in a vertical maze which has ladders that let you move from one level to another. Mean Apple monsters roam about the maze and you must dig holes in the brick floors of the maze so the Apple monsters will fall into them. Then you pound them into the hole and score points. The animation is excellent and the action is slower than many video games. The music that plays while you move about in the maze is cute for a few games but it becomes annoying after a while.

Astroblitz (Creative Software, cartridge—$40). This program is fashioned along the same lines as the arcade game, Defender. You are the pilot of a rocket plane. Your job is to destroy objects on the surface of the planet you are flying over (e.g., gun towers) and in the air (e.g., UFO's). It has several levels of play, but even the beginning level requires quick reaction times for successful play.

Defender on Tri (Nufekop, P.O. Box 156, Shady Cove, Oregon 97539, phone 503-878-2113, cassette—$13. Requires at least an extra 3K of memory). You are the pilot of a rescue ship which must travel through a space maze and rescue astronauts. There are many obstacles and problems to deal with and a limited amount of energy on the ship.

Deadly Duck (Sirius Software, Sacramento, California 95827, cartridge—$34). Deadly Duck is similar to Space Invaders in that you control a well-armed duck at the bottom of the screen and must shoot at all sorts of evil meanies who drop relentlessly from the sky. The game has several levels of play, very good color and good animation.

The following games were reviewed in *Electronic Games 1983 Software Encyclopedia*. A short description of the game and a summary of the evaluation are provided here:

Adventureland (Commodore, cartridge—$40). This is an adventure game for the VIC 20 which was created by Scott Adams, one of the best known of the adventure game developers. Most adventure games are short on graphics. You are given information about the "adventure" and you must then give the computer instructions that you hope will help you successfully negotiate the adventure. These are games of strategy and decision making. Time and eye/hand coordination are not as important

here as they are in most video games. Adventureland places you in a deep forest where you will face many obstacles and dangers as you try to make your way through the forest. Commodore distributes several of the Scott Adams adventure series for the VIC 20. Titles include Voodoo Castle, Pirate Cove Adventure, The Count, and Mission Impossible Adventure. Adventureland was rated 7 on a scale of 10.

Aggressor (HES, cartridge—$40). This game received an 8 rating. It is a space action game that lets you pilot your spaceship (via the joystick) into battle.

Bugblaster (Computermat—$20). This game is similar to Centipede and Serpentine. Waves of millipedes come from the top of the screen through a field of cacti. You must shoot them down before they reach the bottom.

Gorf (Commodore, cartridge—$40). This is a version of the arcade game by Midway. It is in the Space Invaders tradition with wave after wave of space meanies that attack you and must be shot down. Excellent graphics and sound.

Omega Race (Commodore, cartridge—$40). This is another Midway arcade game adapted to the VIC 20. You are a warrior who must do combat with several types of enemy ships.

Gridrunner (HES, cartridge—$40). This is the only game that would earn a rating of 10 out of 10. Everything was excellent. You must fly a spaceship that aims to protect a solar power station from attack by enemy Doids.

Krazy Kong (Nufekop, cassette—$20). This program requires you to rescue damsels in distress at the top of a set of stairs while a crazy gorilla throws barrels at you.

Sargon II Chess (Commodore, cartridge—$40). This program converts the VIC 20 into a more than passable chess player with the chess board displayed on the screen. You can select how well you want the computer to play and can even get the computer to suggest a move for you.

Snakman (Microdigital, cassette—$20). This program is a PAC-MAN look-alike that is very well done. Gregory Yob ranked this program eighth in a list of 20 top VIC 20 products (*Creative Computing*, December, 1982).

Tank Trap (HES, cassette—$18). This game pits you against marauding tanks that must be stopped by correct placement of blockades. Several

levels of play also require you to protect innocent civilians in the area while stopping the tank.

The Crickett (Nufekop, cassette—$25). This game is similar to Frogger. You must help a cricket get across a dangerous highway and over a wall and river.

As this section was written, Commodore announced several new game programs for the VIC 20. You may want to test play games like Midnight Drive, Mole Attack, Draw Poker, Jupiter Lander, Menagerie, and Superslot at your local Commodore dealer.

BUSINESS AND HOME SOFTWARE

As we stated before, the VIC 20 is not a business computer. The primary limitation is the 23-character by 22-line display. It is too small. There are some very good VIC 20 business programs which work within the limitation of the 22 by 23 display, but we do not recommend purchasing a VIC 20 to run these programs. Buy a more expensive computer (such as the Commodore Model 64) that has the features needed for business applications.

Perhaps you already have a VIC 20 and are considering the possibility of using it as a business computer? An unadorned VIC 20 will perform some business functions adequately. An expanded VIC 20 (i.e., with disk drive, printer, and a circuit card that lets you expand the display to 22 lines of 40 characters or even 24 lines of 80 characters) can do a respectable job in many small businesses considering its price.

Word Processing Software

A word processing program turns your computer into an electronic text processing system that speeds the creation of all types of written material—from simple letters to book manuscripts. On May 12, 1983, the last American-made manual typewriter rolled off a Smith Corona assembly line in New York. Manual typewriters were once the loyal companions of people in all sorts of professions, from journalists and secretaries to students. Electric typewriters cut into the manual typewriter market first, followed by electronic typewriters and then word processors. Word processors let you create, edit, and revise documents in a fraction of the time that would be required if a typewriter were used. In the future, personal computers, using word processing software and inexpensive printers, are likely to replace typewriters in many small businesses and in homes.

The VIC 20's limited display capacity makes it a marginal word processing computer, but several programs are available. Here is a brief review of some of the better known programs:

Un-Word Processor (Midwest Micro Associates, cassette—$13). This program is not a true word processing program, but it does let you try out the concept without spending a great deal of money. It is a bit awkward to work with, however, and performs only the most basic of word processing tasks.

RapidWriter (H.D. Manufacturing, 91 Long Hill Road, Leverett, Massachusetts 01504, phone 413-549-3744, tape or diskette—$40). This program is a bit more sophisticated than the Un-Word Processor. It is easy to learn to use, has features normally found only in more expensive programs, and can be used to print material on several different printers (some are limited to the VIC 20 printer). RapidWriter is written in BASIC and thus does not operate as fast as word processing programs written in machine language. All in all, however, it does most of the things you would expect in an inexpensive word processing program.

Home Office (Creative Software, cassette—$29, disk—$35). This program is a basic, no frills word processor that takes advantage of the color capability of the VIC 20. You can type in material, edit and correct it, and print it out. The program is easy to learn and displays prompts, hints, and word processing instruction codes in bright colors such as green, yellow, and red. The program has one mode to create material for the first time and another to edit or correct the material. That makes the program a bit clumsy to use, but it is, we feel, well worth the price.

The Home Office package also includes a database management program in addition to the word processor. The database lets you keep track of all sorts of information (e.g., mailing lists, telephone numbers, Christmas card lists, index of tape or record collections). You can quickly search or sort through information and get a printed copy. If, for example, you would like a list of all business associates (or relatives) on your Christmas card list, this program will search through its data and list them for you. Again, this is a basic, no-frills program, but provides good value when the price is considered.

Quick Brown Fox (548 Broadway, Suite 4F, New York, New York 10012, cartridge—$65). Quick Brown Fox (QBF) is a cartridge-based program that is available in versions for the VIC 20 and Commodore 64 (and the IBM PC). It is easily the most professional word processor available for the VIC 20. The manual is comprehensive, well-illustrated, and understandable. Written with a bit of humor, it is oriented toward the novice user who has no word processing experience. When QBF is executed, it asks you what size screen you have. It will work with standard VIC 20 (23 characters per line) and with most of the 40 and 80 column enhancement boards. That is

a very desirable feature because it lets you try out word processing on a standard VIC 20 and add a display expansion later without changing word-processing programs.

First, there are two complaints about QBF which should probably be mentioned. First, it displays white letters on a black background when used with an ordinary color television. Some people may find this aesthetically displeasing and difficult to read. A more serious limitation of QBF is the way it operates. On sophisticated word processing systems, you type in material, view it on the screen, and make any changes you want as you write. The process takes place with the screen full of material and with the program in a standard "create and edit" mode. QBF has a "create" mode that lets you type in material for the first time. In this create mode the screen is not full of material, however. You must press several keys and go to the "view" mode to see a full screen. To edit that material, you press more keys and go to "edit" mode. Other word processors permit you to create, edit, and view material simultaneously. That is, you do not have to move from one mode to another.

Many VIC 20 word processors lack features you would normally expect on a "standard" program for small computers. QBF is sometimes awkward to use, but it has the features of programs that cost over $200. For example, you can tell it to center a heading, underline or bold face selected words, and delete a word. It has provisions for "global replace" which means if you spelled "D. L. Smithy" as "D. L. Smith" throughout a long document you can tell QBF to search for "D. L. Smith" and change it to "D. L. Smithy." QBF would be our choice for a serious VIC 20 word processor. Better programs may replace it in the future, but QBF is currently the best available in its price range. The company that makes QBF plans to offer a spelling program that will check documents for spelling errors. A QBF-compatible, mailing-list program that will allow you to create and maintain lists of names and addresses also is planned.

We were not able to obtain copies of three other VIC 20 word processors for review. HesWriter is a $40 cartridge which was ranked 13 in Gregory Yob's list of top 20 VIC 20 products. It is available from HES (71 Park Lane, Brisbane, California 94005). Another word processor is TOTL.TEXT from TOTL Software (1555 Third Avenue, Walnut Creek, California 94596). This program comes on diskette or cassette and is available in two versions—one for $25 and a more powerful version for $35. We reviewed a version for the Commodore 64 and felt the manual was poorly written and likely to be confusing to novices. The program itself, however, is powerful. Most of the features beginners want in a word processing program are found in this one. It is relatively easy to use once you figure out what the

manual is trying to say. It also works with many different types of printers. It is, however, written primarily in BASIC and is much slower than Quick Brown Fox. TOTL.TEXT has a simpler and more straightforward method of editing material than QBF and will be preferred by some VIC 20 owners. The same company markets a mailing list and label program.

Accounting and Financial Software

Several companies offer accounting software that runs on the VIC 20. Most programs, however, require the installation of at least 8K, often 16K, of additional memory. Many programs are designed for family financial applications, although some will work satisfactorily in a very small business environment. Powerbyte Software (2 Chipley Run, West Berlin, New Jersey 08091), for example, has a complete line of business and home software. Programs from this company include Accounts Receivable/Payable, Business Inventory, Order Tracker, My Profit Margin, Business Calender, Billing Solver, Cash Flow Model, Liner Regression, Stock Ticker Tape, Home Budget, Medical Records, and Mother's Recipes. Some typical accounting software is reviewed below:

Household Finance (Creative Software, cassette—$30, disk—$45, cartridge—$40). There are four different programs in this package. Together they help you keep track of income and spending patterns across months and years. You can keep track of expenses in 16 different categories (e.g., automobile expenses, mortgage/rent, entertainment, utilities, taxes). The program will provide a tabular or graphic display of spending patterns by month and year. Household Finance stores your monthly financial data on tape or diskette and uses the data to create yearly reports. Those yearly reports will be helpful at tax time since deductible expenses are in separate categories (e.g., medical/dental). The manual for Household Finance is well done and comprehensive. If you are like us, the information this program provides may not be welcome data but it is useful information on how you spend your money.

Home Invenory (Creative Software, tape—$15, disk—$20). The Home Inventory program helps you catalog possessions and keep the list current. Having such a list of possessions can mean thousands of dollars should you need to file an insurance claim. It also makes estate planning much easier. The program lets you divide your inventory into up to 100 categories (e.g., appliances, stereo equipment, art works, chinaware, sports equipment). Information such as serial number, date of purchase, purchase price, current value, and description is provided for each item and you can store your inventory data on either a cassette or diskette. You also can obtain a printed copy of the inventory at any time.

Horse Race Handicapping Program (3G Company, Route 3, Box 28A, Gaston, Oregon 97119, phone 503-662-4492, tape—$25). We do not follow the horses and thus cannot vouch for the usefulness of this program. The program accumulates data on 10 different factors which are felt helpful in predicting the outcome of a race. It uses information from the Daily Racing Form on each horse in the race and calculates the odds of each horse winning.

Small Business Accounting (TOTL Software, diskette—$85. Requires at least 24K of memory). This set of programs is a basic, no-frills accounting program that handles accounts receivable, accounts payable, inventory, customer records, invoice printing, and monthly statements. It is suitable for many very small businesses, salespeople, and service professionals. However, we would advise you to get detailed information on this and any other accounting package before buying it. The record keeping requirements of small businesses vary widely and this program may or may not fit your particular needs.

Spreadsheet Software

Companies with large or mainframe computers have routinely used "spreadsheet" software for years. These programs allow you to automate the process of performing tasks that require calculations where the formulas and formats remain the same but the actual numbers used change. Examples of such tasks are job cost estimates, expense reports, loan and mortgage amortization tables, salary grids, break-even analysis, inventory management, routine accounting tasks, and statistical analyses.

Several years ago, some young fellows wrote and marketed a spreadsheet program called VisiCalc that ran on the Apple II and several other personal computers. Computer owners have now spent over $40 million on that one program, and there are at least 50 other programs in the VisiCalc tradition.

When we received our copy of **PractiCalc** (distributed by Micro Software International, 50 Teed Drive, Randolph, Massachusetts 02368, phone 617-961-5700, diskette or tape—$40. Requires a minimum of 16K memory, will use up to 32K.) our expectations were somewhat low. We have used VisiCalc and SuperCalc, two very powerful spreadsheet programs that cost as much as $300 depending on the version. Could a $40 program running on an under-$100 computer compare favorably? PractiCalc, because of the limitations of the VIC 20 screen, does not display as much information on the screen as does VisiCalc, and it does not have some of the fancier features of the more expensive programs. It is, however, a thoroughly professional product that will function quite acceptably in a business environment. The manual is one of the better ones we reviewed. It is

easy for a novice to follow, yet provides useful information for advanced computer users. The program itself has all the basic features of more expensive spreadsheet software and even has some features we preferred. For example, odd-numbered columns are in a different color than even-numbered columns and the "active" cell is blue. The effective use of color makes it easier to set up and use your own forms. PractiCalc is an excellent value for the money. The company that distributes this program plans to market a data base management program, a check-writing program, and a program that lets you compose music on the VIC 20. If PractiCalc is an example of the quality we can expect from Micro Software International, they should be well worth considering.

Other Business and Professional Software

TOTL.TIME (TOTL Software, diskette or tape—$30). If you and/or your colleagues perform work that requires precise scheduling of sequenced activities, this program may be of great help. It can plan the work of a team of individuals who are working on several different projects. Different types of work are recorded separately and reports or charts are generated in many different formats such as by project, by individual, or by activities. The program can produce several types of summary charts (in color on the display). These charts can be printed on virtually any printer that can be attached to the VIC 20.

This is not a simple program, and thus not something you are likely to use for relatively straightforward scheduling of appointments. If, on the other hand, your work involves something like building houses where teams or individuals with different skills (e.g., electricians, carpenters, carpet layers) must be scheduled for work, then TOTL.TIME may be very useful.

Research Assistant (TOTL Software, cassette or disk—$30). If you keep cards or files of information, you must decide how those files are to be arranged. For example, if you keep track of students in a college and the courses they are taking you could create files on the basis of student name, course(s), or major. But what happens when you need to get a list of all the students who are enrolled in a particular course, but the files are arranged by student name? This program lets you type in information in much the same way you would write information on 3 × 5 cards. Then it permits you to search through your electronic "cards" by giving the computer a key word or words that will guide the search. That means the same file can, in effect, be organized by student name, by major, and by the courses the students are taking (or have already completed).

We found this program somewhat complicated to use, but it is one of only a few database management programs available for the VIC 20 and worth considering. A program that appears similar is **Data Manager** (MicroSpec, 2905 Ports O'Call Court, Plano, Texas 75075, disk—$60).

LANGUAGES AND PROGRAMMING AIDS

The VIC 20 computer speaks the computer language BASIC because the BASIC language program is installed in ROM at the factory. BASIC is a very good language, but it is not the only computer language nor is it the best for every application. We will discuss software that lets your VIC 20 speak other computer languages as well as programs that add flexibility and ease of use to the VIC 20 when you do your own programming.

Language Software

VICFORTH (HES, cartridge—$60). FORTH is a language that is growing in popularity among small computer users. It is more structured than BASIC and is, in the opinion of some, a language that provides programmers with more power and flexibility than is available with BASIC. VICFORTH has been well-received and comes with a well-written manual. Another version of this language, FORTH 20, is available from UMI for the VIC 20 in cartridge form.

Programming Aids

BUTI (UMI). BUTI is a utility program that adds 17 new key words to the BASIC used by the VIC 20. Most of the new key words are especially useful to programmers. For example, BUTI allows you to renumber lines of instructions in a BASIC program, something that is often necessary when you are developing and revising a program.

Programmer's Aid Cartridge (Commodore, $60). This cartridge, like BUTI described above, is an aid to the programmer working in BASIC. It adds 20 new key words to BASIC.

VIC-KIT (Skyles Electric Works, 231 E. South Whisman Road, Mountain View, California 94041, phone 415-965-1735). This is another cartridge that adds key words to the VIC 20's BASIC. The VIC-KIT cartridge also has 4K of RAM and, if you wish, a parallel printer interface. Its price is $90.

VICMON Machine Language Monitor (Commodore, $60). The VIC 20 computer can execute programs written in 6502 machine language because it uses the 6502 microprocessor chip. The VIC 20, however, does not have provisions for writing programs in 6502 machine language. If you would

like to learn to use this language, you will need VICMON or a similar program. It helps you write 6502 "assembly language" programs which it converts into 6502 machine language progams the VIC 20 can execute. HES sells a similar program for the VIC 20 called HESMON.

Sketch Pad and Char-Gen (Micro Ware, tape—$25). This program helps you draw high resolution pictures with the VIC 20. There is also a program called Grafix Designer that helps you develop your own special set of characters. See the chapter on graphics and sound for information on other programs that help you use the graphics and sound features of the VIC 20.

Synthesound (HES, cartridge). VIC 20 BASIC is not well-endowed when it comes to key words that let you control the superior music and sound synthesis features of this computer. Synthesound adds a complete set of key words to VIC 20's vocabulary that lets you conveniently create all sorts of music and sound effects.

Turtle Graphics (HES, cartridge—$40). The Logo computer language is currently enjoying a rise in popularity, particularly for use by young children. One aspect of Logo is the use of "turtle graphics," a simple yet powerful system of instructions that permit even preschool children to create and manipulate animated color graphics. This cartridge is not a complete Logo language, but it does allow children to use the most popular aspect of Logo, turtle graphics. It has 30 different key words that allow children (and adults) to control use of color, sound, and animation on the VIC 20 screen. If you would like to read more about Logo and turtle graphics, you might want to read Seymour Papert's book *Mindstorms* or the book *Computers, Teaching, and Learning* by Willis, Johnson, and Dixon.

Game Program Development Kit (French Silk Smoothware, P.O. Box 207, Cannon Falls, Minnesota 55009, phone 507-263-4821, $52). This set of materials was developed for people who are serious about developing machine languge progams, particularly video games, for the VIC 20. It is probably not something you will want to tackle until you have some experience.

Educational Software

There are several hundred educational programs for the VIC 20, and many of them are available free of charge (see the section on public domain software at the beginning of this chapter). In this section, we will review some of the popular educational software:

Typing Tutor plus **Word Invaders** (Academy Software, P.O. Box 9403, San Rafael, California, cassette—$22). You get two programs for the price of one on this tape. Typing Tutor is a drill and practice program that teaches you how to touch type. It has eight levels of difficulty and is a colorful, well-written typing tutor program. We recommend it for any VIC 20 owner who would like to learn to touch type. The program keeps track of errors as well as typing speed, and it highlights in red all the letters typed incorrectly in an exercise.

Word Invaders is an excellent game that lets you practice your typing skills in a video-game environment. Lines of words come marching down the screen toward the bottom. You must type them correctly to erase them from the screen. If any lines of words reach the bottom, you loose. The program keeps track of your speed and assigns a game score based on speed and accuracy (assuming you are not destroyed by words that reach the bottom of the screen). This is an excellent program that makes typing practice an enjoyable pastime instead of boring drudgery.

Flashcard Maker and **Flashcard Quiz** (Academy Software, cassette). These two programs allow you to create quizzes and then use the computer to administer them to students (or take the quizzes yourself). The programs have few fancy options but they do work well.

Visible Solar System (Commodore, cartridge—$40). This program lets you pilot your spaceship on a journey through the solar system. It makes good use of the color graphics features of the VIC 20 to display a model of our solar system. As you pilot your ship, you learn about the location of planets and their relationship to other planets.

Telecommunications Software

An increasing number of personal computer owners are using their machines to gain access to information and services available from other computers over ordinary phone lines. To use the VIC 20 for "telecommunications", you will need a modem and a telecommunications software package. Modems are discussed in the chapter on hardware and accessories. A telecommunications program lets you send instructions to another computer over the phone and receive information from the other computer. Incoming information is displayed on your video display. It may also be sent to your printer, stored on cassette or diskette, or placed in the memory of the computer for later use.

There are several telecommunications programs for the VIC 20 but one we consider a "best buy" is **Terminal-40** from Midwest Micro Associates (P.O. Box 6148, Kansas City, Missouri 64110, cassette—$30. Requires 8K

of memory and can use more if available.) A major limitation of the VIC 20 as a "terminal" is its limited display capacity. Terminal-40 takes care of that with software that creates a 40-character by 20-line display. That one feature makes this program a good buy. Its manual is also well-written and informative. The program has a number of nice features. The function keys on the VIC 20, for example, are used to control special program features. Incoming material is stored in a "buffer" (a special section of the VIC 20's RAM memory). It can be retrieved from the buffer and viewed or printed. Terminal-40 has a color or black-and-white option that lets you select the correct video signal for the clearest, sharpest image. Midwest Micro also sells a more sophisticated program called **SuperTerm-40** that, among other things, lets you store material in the buffer on tape or diskette for later use.

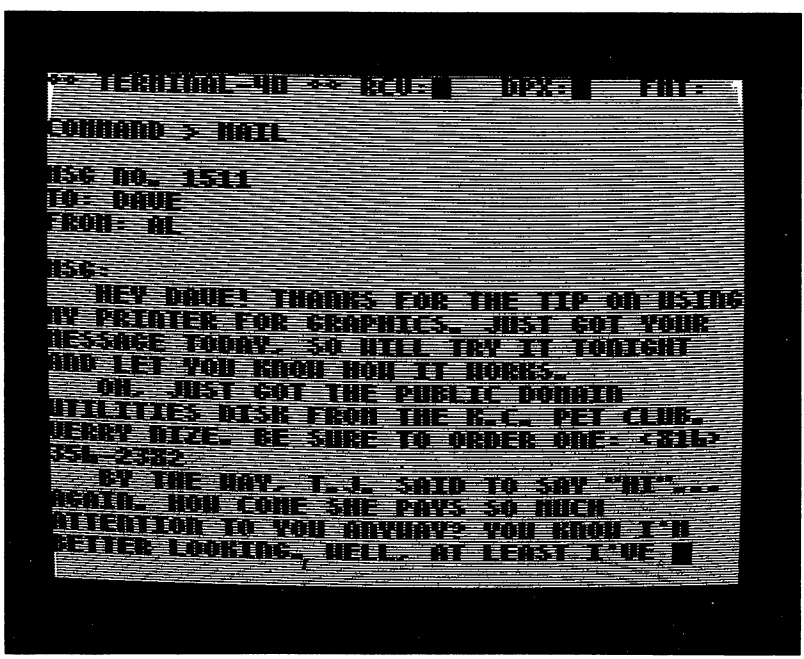


Figure 10.1 A screen display from Terminal-40.

Chapter 11

Selecting Hardware and Accessories

The VIC 20 is a basic computer at a minimal price. The low price prompted hundreds of thousands of people to buy the VIC 20 as their first computer. Its limitations prompted hundreds of companies to design and manufacture accessories to enhance and expand the VIC 20. This chapter explains what many of the accessories do, what they won't do, and points out some of the pitfalls and problems associated with buying accessories.

Please heed two warnings as you read this chapter. First, we have generally included prices when discussing specific products even though computer product prices are notoriously fickle. You may find that some prices have increased and many have decreased by the time you buy accessories. Second, suppliers of computer accessories are an odd assortment. Most companies have good intentions, some provide prompt service and excellent quality. Most eventually deliver products that will work. Exercise reasonable caution when spending your hard-earned money, however, because there are a few crooks and a few totally incompetent entrepreneurs out there. Since the VIC 20 is sold in many department and discount stores, it often happens that the salesperson knows very little about computers and has confused or garbled half of what he or she does know. Be very cautious about what salespeople tell you unless you can actually see the product do what you have been told it will do.

We personally have tried to use as many of these products for the VIC 20 as possible, and to summarize the conclusions of reviews published in magazines. When several products are on the market that do the same thing, we have described the models with which we are most familiar. Products we do not mention may be inferior products, but they also may have been unavailable for review when the book was written. We have tried to be fair in our reviews and to point out positive as well as negative features.

Extra Memory

The VIC 20 computer has a little over 5K of RAM. Although that is enough for many uses, you may find yourself staring at an OUT OF MEMORY message on the screen more frequently than you would like. Some of the larger commercial programs for the VIC 20, in fact, require you to have an extra 3, 8, 16 or 29K of RAM before they will operate. The VIC 20 can handle a total of 32K of RAM memory counting the 5K that is built into the basic machine.

Additional memory is added to the VIC 20 via the cartridge slot on the back left side of the computer. Many different companies offer memory expansion cartridges that plug into this cartridge slot. Commodore itself manufactures 3K ($40), 8K ($60), and 16K ($110) cartridges. In addition, Commodore's SuperExpander Cartridge ($70) includes 3K of RAM.

Other companies that manufacture memory for the VIC 20 include:

Data-20 Corporation (23011 Moulton Parkway, Suite B10, Laguna Hills, California 92653). Their well-made cartridge has 20K of RAM and costs $80.

OEM (2729 South U.S. 1, Suite 12, Fort Pierce, Florida 33450, phone 305-465-9363). A 4K RAM is $40, 8K is $50. The 8K board is designed in such a way that you can plug another cartridge into the back of it. Some memory expansion cartridges will not permit you to plug in any other cartridge when they are in use.

Apropos Technology (350 N. Lantana Avenue, Suite 821, Camarillo, California 93010, phone 805-482-3604). The expansion RAM cartridge manufactured by this company costs $169. For that you get 27K of RAM. Add that to the VIC 20's 5K, and you have all you can use—32K. In addition, the cartridge has slots for plugging two other cartridges into it and a switch that lets you determine which expansion slot is used. That means you can plug in two of your favorite games and flip a switch to select the one you want to play. A fuse, a reset switch, and switches that control how much of the memory is active also are on the board.

MicroWave (1342 R Route 23, Butler, New Jersey 07405, phone 201-838-9027). This company's 16K expansion RAM is $119. This board also includes two slots for additional cartridges that are controlled by a switch. It also has a memory allocation switch and a reset switch. (A reset switch lets you regain control of your computer when it "hangs up" because of a problem in the program it is running. The reset switch lets you restart the computer without erasing the progam in memory.) This board also in-

cludes a switch that lets you "protect" part of the memory. Protecting memory lets the computer read data in that area but it cannot be changed until it is "unprotected."

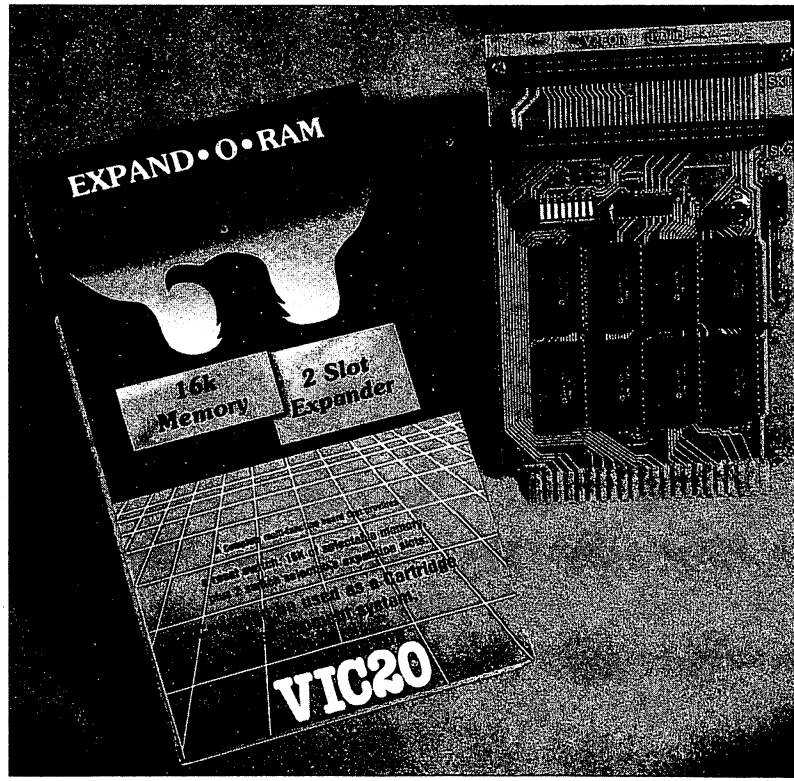


Figure 11.1 The 16K RAM expansion board from MicroWare.

Expansion Boards

When Commodore designed the VIC 20 it made provisions for plugging in only one cartridge at a time. Some VIC 20 fanatics, however, find they need space for four or five cartridges at once. To deal with this problem some memory boards have extra slots on them to plug in one or two cartridges.

Another way to get room for additional cartridges is to buy an expansion board. These boards plug into the cartridge slot on the back of the VIC 20

and allow you to plug in from two to six different cartridges at once. Some have provisions for selecting which cartridges are "active" and some even include fuse protection and additional switches to control the operation of the computer. Here are a few companies that make expansion boards:

Digital Interface Systems Company (P.O. Box 8715, Portland, Oregon 97207, phone 503-295-5890). One of the products this company produces is a 5-slot expansion chassis. It uses a rotary switch to determine which cartridges are active; it has additional circuitry that makes the use of memory on an expansion chassis more reliable (data is "buffered"), and there is a fuse and reset switch. Finally, the chassis has provisions for plugging in an external power supply to avoid overloading the power supply in the VIC 20. It is a nice product.

Precision Technology (2970 Richards Street, Salt Lake City, Utah 84115, phone 801-487-6266). Their $60 card has three slots, a fuse, and switches that activate any or all the slots.

Data20 Corporation (This company's address was previously mentioned.) Their 4-slot expansion chassis with fuse is $50.

Cardco (313 Mathewson, Wichita, Kansas, phone 316-267-6525). This company produces a 3-slot ($40) and a 6-slot ($100) expansion chassis. Both are fuse protected and incude a reset switch as well as a cartridge selector switch. Some owners have reported problems with interference when using the Cardco 6-slot model because it connects to the VIC 20 with a cable.

OEM produces a 3-slot expander for $40. If switches are needed that determine which slot is active, the price is $45.

Compuscope (6400 Signal Street, Tillamook, Oregon 97141) has a very well-designed 8-slot expansion board dubbed the "Supermother." This board has switches that determine which slot is active, a reset button, fuse, write protect switch, and software that lets you make backup copies of cartridge programs on diskette or cassette. This board is buffered to improve performance and costs $170.

ROM Expansion Boards and PROM Burners

If you create a marvelous program for your VIC 20 and would like to permanently install it in a Read Only Memory chip, there are devices that allow you to do just that. **Gloucester Computer Bus Company** (6 Brooks Road, Gloucester, Massachusetts 01930) produces the PromQueen, a device that plugs into your cartridge slot. This $200 product lets you create the programmed ROM chips that are the heart of video-game cartridges. The PromQueen has 4K of RAM which you can use as you develop your

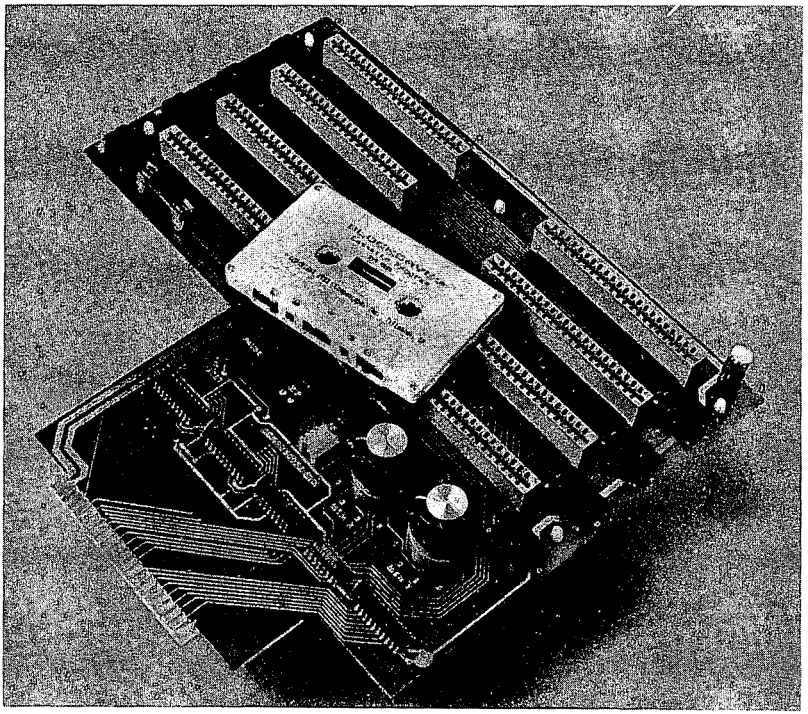


Figure 11.2 The Supermother expansion board.

program. There is also a special socket for inserting blank programmable ROM chips (2716 or 2732) and a power supply. The product comes with a program called Hexkit 1.0 which makes it easier for you to write machine language programs and place your program in a programmable ROM. This is a professional product that will require some experience in machine language programming to use.

Another company, OEM, produces a $20 ROM board that lets you plug in ROM chips and run them on the VIC 20. OEM will also create programmed ROM chips from programs you send them.

Printers and Printer Interfaces

Commodore markets a printer, the **VIC 1525 Graphic Printer**. This $400 printer is an inexpensive Japanese import that has been modified to operate correctly with the VIC 20. It has three positive points and a few negative points. On the positive side, the printer is relatively inexpensive. It is also capable of printing all the VIC 20's graphic characters as well as letters, numbers, and symbols. Finally, the 1525 printer can be plugged

into the VIC 20 without the need for special cables or interfaces. Plug it in, turn it on, and it works. On the negative side, the 1525 is a relatively slow printer and print quality is mediocre at best. And, although we have no concrete data, the robustness of this printer might be questioned.

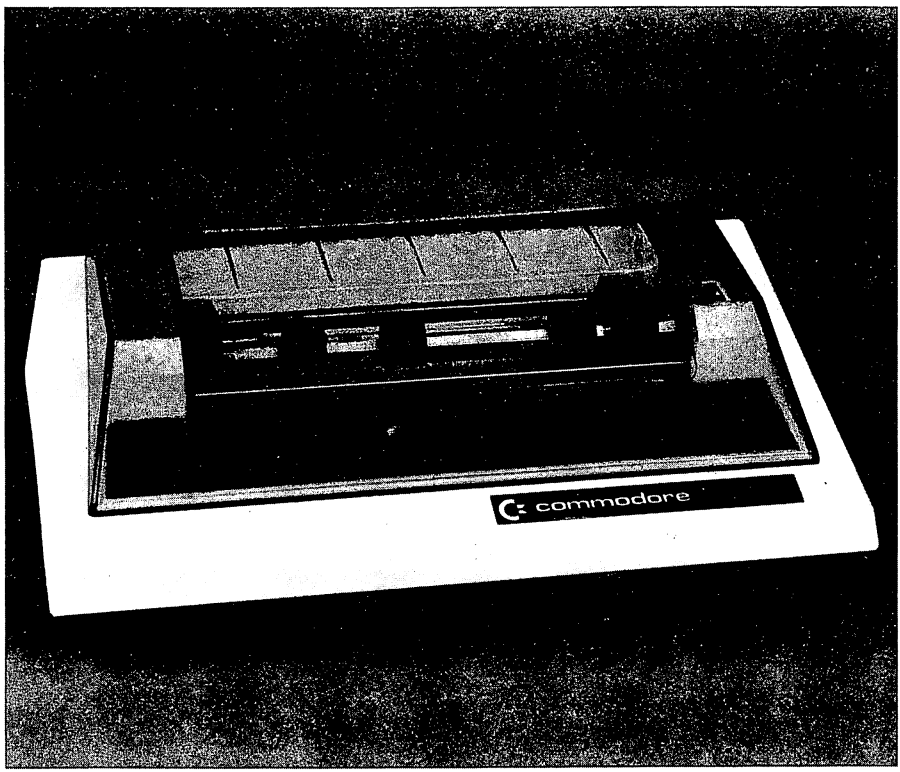


Figure 11.3 The Commodore 1525 printer.

There are at least 200 different models of printers on the market today and they range in price from under $100 to several thousand dollars. Quality, as you might expect, varies as well. Some printers produce barely legible print while the output of others looks as if it came from a high quality office typewriter. Several of the magazines mentioned in Chapter 11

regularly publish articles on selecting printers for your computer. Jerry Willis' book—*Computers For Everybody 1984 Buyers Guide*—also has a section on printer selection. You may want to consult other sources if you are printer shopping. We will concentrate here on the issue of compatibility with the VIC 20.

If you successfully connect a printer to your VIC 20 three major obstacles must be overcome:

1. The printer must accept data in the form the VIC 20 transmits it *or* an interface must be placed between the VIC 20 and the printer that translates the signals form VICese to Printerese.

2. The cable between the computer and printer must have the proper connectors on each end.

3. The code used by the VIC 20 must be the same one used by the printer *or* a program must translate the codes for the printer.

Let's deal with the interface issue first. Printers generally come in two different flavors—serial and parallel. Serial printers accept data a bit at a time with each character code made up of 7 or 8 bits. Parallel printers accept data 8 bits at a time over 8 separate data lines. This means that parallel printers are faster because they can accept more data (8 bits on 8 lines as compared to 8 bits on 1 line with a serial printer). The VIC 20 contains a serial interface port (the round DIN plug with six holes on the back of the VIC 20). In the best of all possible worlds, this serial interface would be where you could connect a serial printer. As you might expect, the world could stand a little improving. Every single printer we are aware of uses an industry standard voltage level when sending data to the printer. It is called the RS-232C standard and it uses −12 and +12 volts. The VIC 20 and the 1525 printer use 0 and +5 volts instead. That means this interface is not compatible with 99% of the serial printers out there. If you intend to use a serial printer, you will have to buy an RS-232C interface cartridge. The version from Commodore sells for $50 and plugs into the user port on the back of the VIC 20. This particular serial interface is wired to operate with a modem rather than a printer. That generally means the cable to the printer will have to be rewired.

Parallel printers also are a problem because the VIC 20 does not have a standard parallel interface connector. (Most parallel printers use the voltage levels adopted by Centronics Corporation.) That leads us to the second issue—connectors. Serial printers generally use a special connector called a 25 pin D connector. Most computer stores carry cables with this type of connector on each end. The Commodore RS-232C interface has a plug that fits this type of connector.

Parallel printers use one of two types of connectors. One is called a "Centronics compatible" connector because it was pioneered by Centronics, a U.S. printer manufacturer. A few parallel printers expect you to provide a cable with an "edge card" connector on it. These are often hard to find. In any case, you will have to build or buy a cable that correctly plugs into your VIC 20 on one end and fits your printer on the other. Getting a cable that works is sometimes the most difficult aspect of getting your printer up and running.

The third issue to be considered is that of code compatibility. The code used by the VIC 20 is not exactly the same as the industry standard code, ASCII (short for American Standard Code for Information Interchange). VIC 20 code for letters and numbers is the same, but there is no ASCII equivalent for VIC 20 graphics codes (most print as letters or symbols). A few of the VIC 20 codes tell the printer to do things you would rather it not do (such as stop printing and go into a resting state). The VIC 1525 printer is set up to accept the same codes as the VIC 20; you might want almost all the other printers to use the ASCII code.

Some Solutions

While we were working on this book, we had the opportunity to use a program called **Smart ASCII** from Midwest Micro Associates (P.O. Box 6148, Kansas City, Misouri 64110). Smart ASCII is $60. That includes a program on cassette tape (it can be transferred to disk) and a cable. We used this program with an **Epson MX-80 printer**, but there are at least 70 models it will work with. The quality of print from the Epson was far superior to that available on the Commodore 1525. Smart ASCII solves all three problems noted above. A parallel port is provided, the cable works with many standard printers, and the program provides three different types of "translation" which you can select. The VIC 20 code that tells the computer to clear the screen is translated to CLR. The code for switching to reverse characters prints as RVS (Reverse Off prints as rvs), and the code to set character color to red predictably prints as RED. The instruction manual that comes with Smart ASCII is clearly written and informative. The program is compatible with several of the more popular word processing programs for the VIC 20, a point that should not be overlooked because some printer driver programs will operate only with BASIC programs.

Another company, MicroWare Distributing, distributes the **Tymac Parallel Printer Interface**. It is $119 and comes with a printer cable that has a Centronics connector. The cable plugs into the serial socket on the VIC 20. This interface has several more features than the Smart ASCII. It has 2K of buffer memory, which means the computer can send 2,000 characters to the interface and then begin another task while the buffer takes over the

job of sending character codes to the printer. The literature on this interface indicates it can duplicate VIC 20 graphics on many printers, but you must specify which printer you will be using when the interface is ordered.

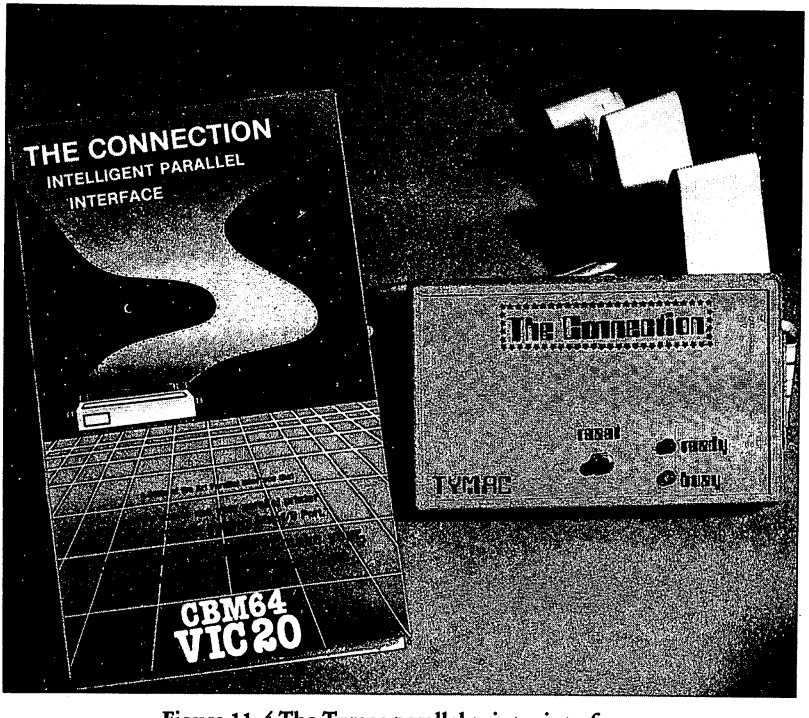


Figure 11.4 The Tymac parallel printer interface.

Other companies that manufacture parallel printer interfaces include **Cardco** ($80), **OEM** ($100), and **Micro Wold Electronix** ($120—6340 W. Mississippi Avenue, Lakewood, Colorado 80226, phone 303-934-1973). Serial printer interfaces are manufactured by **Data20** ($70, includes switches so you don't have to rewire the cable), and **OEM** ($40). Do not shop for interfaces simply on the basis of price, however, since minimal systems may not work correctly or do everything you want. Decide what type of printer you will use, what software you plan to use it with (e.g., only BASIC, accounting software, a word processor?), determine if the interface is compatible with that software, and decide what you want the interface to do (e.g., translate VIC 20 code to ASCII, print VIC 20 graphics). Then select the interface that does the job you need done. Figure 9.5 shows the

printout from a standard parallel printer using the software and cable provided by OEM. Note that the OEM products allow you to print VIC 20 graphics and inverse video characters.

Figure 11.5 OEM printer output.

Video Enhancement Hardware

Inexpensive computers aimed at the home and school market generally achieve their low prices in one of several ways: they use an inexpensive keyboard, they make many items "options" which really aren't (you must buy them to have a usable computer), or they skimp on the video display. The VIC 20's RAM memory is on the short side (the "option" approach), but memory is not that expensive. The video display, however, leaves something to be desired. Yes it does have color and graphics, but that does not compensate for the measly 23-character by 22-line display. The 22 × 23 display format is the feature most often criticized on the VIC 20. When writing a BASIC program, you can write a program line that is as long as 88 characters, but the VIC 20 will need three or four screen lines to display that program line. It makes for a messy and confusing display. A better format is 24 lines of 40 characters. Even better is a 24- or 25-line by 80-character format.

Several companies recognized the need for a higher capacity video display on the VIC 20 and designed products that let you use either a 24 × 40 or a 24 × 80 format. The VideoPak from Data20 costs $300 with 16K of extra RAM ($400 with 64K RAM). It gives you a choice of either a 24 × 40 format or a 24 × 80. The VideoPak plugs into the cartridge slot on the VIC 20. It lets you use all the graphics characters available on the VIC 20 keyboard and it includes software for using the VIC 20 as a terminal. The quality of the display is quite good when used with a video monitor.

Normal color televisions, however, are not designed to display 24 × 80 computer displays. The characters are generally so fuzzy that you cannot read some of them. Eye fatigue also is very likely. So if you plan to use the VideoPak in the 80-character mode, plan on buying a video monitor as well.

Light Pens, Trackballs, and Game Paddles

Educational and recreational applications of the VIC 20 sometimes call for a means of input to the computer other than the keyboard. Many arcade games, for example, do not work well if you must press keys to control movement. Commodore originally sold a video-game joystick for the VIC 20 that looked suspiciously like the one Atari makes. Apparently Atari thought it was a little too much like theirs and they brought suit to stop Commodore from selling it. Commodore now sells a different one that, in our opinion, is simply not as good as the Atari version. If you are shopping for joysticks, you may want to test drive several before buying. Joysticks vary considerably in their feel and fit, and a device that does not fit your playing style can cause fatigue and low scores. Some people feel the joystick made by Atari is difficult to use because you can't feel when you have moved the stick far enough to register. Others find its stick too small and too short for their tastes. At $10, however, it is an inexpensive product that is available almost everywhere. It will plug into the VIC 20 with no modifications. Companies such as Baylis, Datawasher, Discwasher, Spectravision, Suncom, and Wico manufacture higher-priced but more adaptable models in a variety of formats and styles. Wico's Red Ball joystick, for example, costs $35, is sturdily built, and has two strategically placed fire buttons. Wico, which also manufactures joysticks and trackballs for arcade games, has a large line of products which include the Power Grip Joystick ($37). It has a large handle that fits your hand. A deluxe model ($45) comes with three different interchangeable handles.

Both Wico and TG Products market a "trackball" game controller that can be substituted for a joystick on the VIC 20. Trackballs let you roll a large round ball that is set in the top of the controller to direct the action of

a game. This procedure is more precise than many joysticks and can be performed more quickly. Trackballs are generally much more expensive than joysticks, however. For evaluations of new game controller products you may want to look in current issues of magazines such as *Compute!* or *Video Games*.

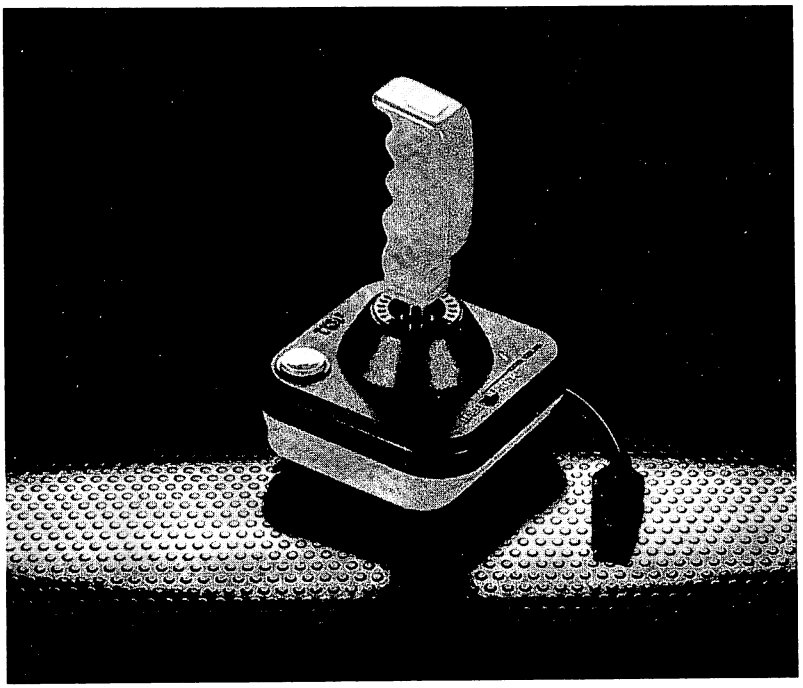


Figure 11.6 The Powergrip joystick from WICO.

Another device that may be of interest is a light pen. Light pens look a lot like regular ball-point pens. When properly interfaced to the computer, however, a light pen can be used to determine where on the screen the user is pointing the pen. We tested the light pen from 3 G Company (Route 3, Box 28A, Gaston, Oregon 97119, phone 503-662-4492). This one costs $40 and comes with a demonstration program and instructions on how to use the pen in your own programs. The pen itself has a sensitivity adjustment on it. The brightness on your monitor or television may also require adjustment before the pen works correctly. The demonstration program that came with the pen displayed a series of multiple-choice questions on the screen. To select an answer, all we had to do was place the

pen on the screen where the answer was located. The pen worked well. Light pens are likely to be increasingly used in education where they can provide computer access both to students who are too young to read and to some handicapped students who may not be able to deal with a keyboard.

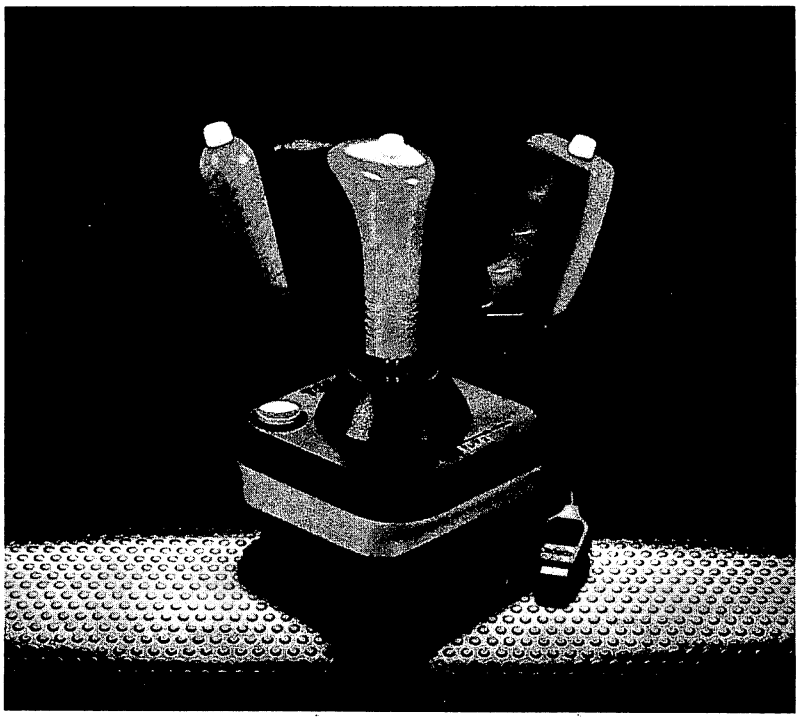


Figure 11.7 The WICO joystick comes with three different handles.

Modems

A modem (short for modulator-demodulator) is a required accessory if you plan to use your VIC 20 for communicating with other computers and information utilities over the phone. A modem takes the signals from your computer and converts them into tones that can be reliably transmitted over phone lines. It also converts tones transmitted to your computer into signals the VIC 20 can process.

Modems can be connected to the computer in two ways. Some plug directly into the computer; others plug into a serial port. Because the VIC 20 does not have a "standard RS-232" port, the most inexpensive way to

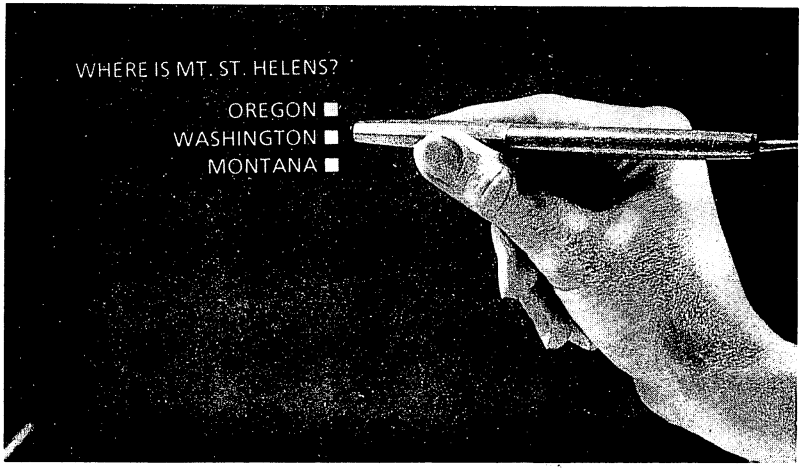


Figure 11.8 The 3G light pen.

Figure 11.9 The VIC 20 modem.

attach a modem to the VIC 20 is to buy one that connects directly. Commodore's VICmodem does just that and costs around $100. That is an excellent price. The VICmodem is a "direct connect" model on the phone end as well. "Acoustic" modems convert computer signals into audible tones that are transmitted and received by placing the handset of a standard phone in rubber cups on the modem. Acoustic models don't work as reliably as direct connect models that plug directly into phone outlets as if they were another phone. Noise in the room, for example, may be picked up and transmitted as data by an acoustic modem.

HES (71 Park Lane, Brisbane, California 94005) also sells a direct connect modem for the VIC 20 ($80) but we have not had a chance to test it. Both the HES and Commodore models transmit data at 300 bits per second. That is the standard speed on most national systems such as The Source and CompuServe, but there is likely to be a shift to higher speeds soon (i.e., 1200 bits per second). Modems that transmit and receive data at 1200 BPS, however, generally cost between $400 and $600. If we were buying a modem for the VIC 20 today, the choice would probably be one of the under $100 300 BPS models.

The inexpensive models, however, are not the only type of 300 BPS modems available. You can spend more and get very fancy features such as auto dial. That means the computer can tell the model to "call Joe" and the modem will dial the number for you and signal when the phone is answered. If you would like more detail on selection of a modem, you may want to read an article by one of the authors (JW) in the June 1983, issue of *Easy Home Computer* magazine. It explains the features of modems and suggests what to look for in a modem for your computer.

A modem, however, is not all that is needed to use your computer as a telecommunications device. You must have software that converts your computer into a "terminal" that can communicate with other computers. See Chapter 10 for information on telecommunications software for the VIC 20.

Disk Drives, Cassette Recorders and Substitutes

It has been the practice of many computer manufacturers to sell their basic computer at a very low price and then charge very high prices for accessories such as disk drives. Texas Instruments used that strategy with its 99/4A computer. It is to Commodore's credit that the 1541 disk drive for the VIC 20 is priced very competitively. The 1541 is capable of storing over 160,000 characters of information on one side of a diskette and, while it is somewhat slow as disk drives go, ours has been a very reliable unit. Because it works well and is priced aggressively, the likelihood of another company offering its own model is low. The only area besides speed of operation

where the 1541 could be faulted is capacity. The need for high-capacity disk storage, however, is most often associated with business applications and the VIC 20 is not a business computer. There is a way, however, of adding high capacity disk drives to the VIC 20. See the section on IEEE Interfaces in this chapter.

There is one commercially available alternative to the 1541 disk drive. It is the Stringy Floppy from Exatron (181 Commercial Street, Sunnyvale, California 94086, phone 800-538-8559). This product uses specially designed tape cassettes (called wafers) to store programs and data. The price of the Stringy Floppy is under $200 and it appears to work reliably and quickly. Exatron has established a reputation as a company that makes an excellent product and bends over backward to provide assistance to users of their equipment. The difference in price between the disk drive and the Stringy Floppy is between $100 and $200. For that savings, you get a product that works well but is not likely to be supported by many software producers. Most of the software for the VIC 20 is on cassette, disk, or cartridge. Buying the Exatron unit will give you fast, reliable storage but it will mean you won't be able to buy many programs in the medium you use (wafers). Consider the drawback of incompatibility while you consider the possible savings.

A similar caution should be voiced in connection with the Commodore tape recorder system. Many home computers tout their ability to use standard cassette recorders. We have never (absolutely never) used a computer with a standard garden variety tape recorder that consistently loaded and saved tapes reliably. It is a very unusual event when our VIC 20 does not load a program properly. There are, however, several companies that sell hardware that lets you use a standard tape recorder with your VIC 20. MicroWare, for example, sells a $50 interface that lets you use standard recorders. In addition, this device allows you to connect two tape recorders to the interface and duplicate a tape. Cassettes, after long use, will wear out so there is something to be said for the ability to "back up" a tape that contains a $20 or $30 program. In general, however, we look on tape interfaces for the VIC 20 with some suspicion. Why tamper with a good thing unless the savings is substantial?

Cardco and Integrated Controls also sell recorder interfaces. Eastern House Software (3239 Linda Drive, Winston-Salem, North Carolina 27106, phone 919-924-2889) markets the VIC Rabbit Cartridge ($40) which speeds up the standard Commodore recorder. With this product, the VIC 20 loads and saves data and programs at approximately six times its normal rate. This may be a valuable feature to individuals who use the cassette frequently. Note, however, that the Rabbit Cartridge will not speed up loading of commercially-produced cassette software unless you

load it into the computer at its regular speed and then use the Rabbit Cartridge to save it on another cassette at the faster speed.

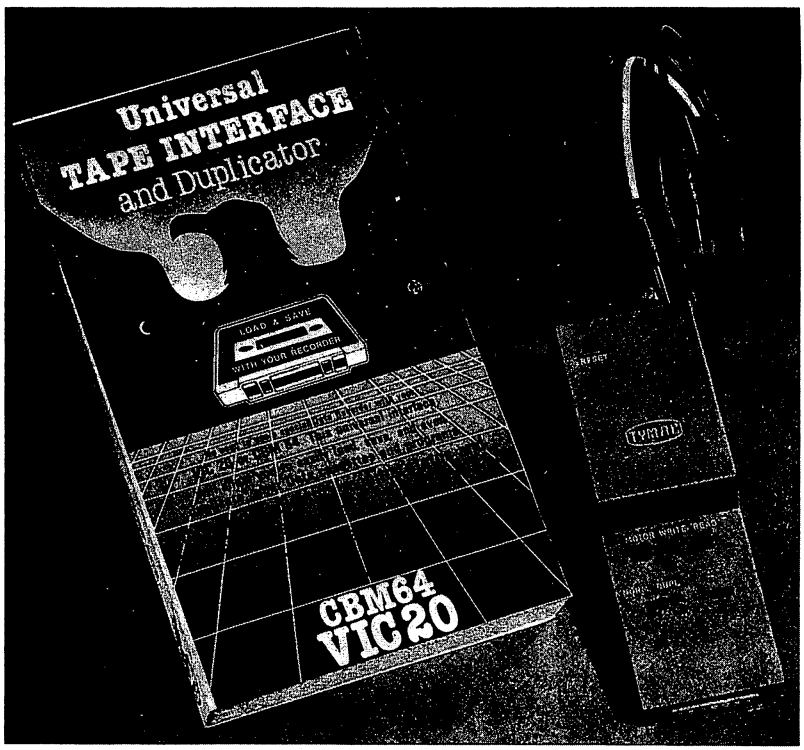


Figure 11.10 The tape recorder interface from MicroWare.

Networks and IEEE Interfaces

Commodore's 1541 disk drive and 1523 printer both connect to the VIC 20 through a serial port. That method of connecting printers and disk drives differs from the approach used on other Commodore equipment. On the PET and CBM models peripherals are connected via an interfacing approach called IEEE-488. This approach is a very reliable and very fast method which we find preferable to the serial method used on the VIC 20. As you might expect, the IEEE-488 standard is more expensive than serial port input/output which is why it is not used on the VIC 20.

At least three companies, however, offer an interface cartridge that lets the VIC 20 operate disk drives and printers which are designed for IEEE-488 interfaces. That means you can use high-capacity disk drives such

as the Commodore 4040, 8050, and 8052 on your VIC 20 as well as high quality printers such as the Commodore 8023. The IEEE-488 interfaces for the VIC 20 generally consist of a piece of hardware that plugs into the VIC 20 expansion port. Then a cable can be run from the interface to printers and disk drives. Micro Systems Development (11105 Shady Trail, Suite 104, Dallas, Texas 75229, phone 214-241-3743) has an interface board that sells for $100. Although the instructions that come with the board are inadequate (two pages), the board itself seems to be well constructed. This company generally sells to dealers rather than end users, however. Southwest Micro Systems (2554 Southwell, Dallas, Texas 75229, phone 214-484-7836) does sell to end users. The mail order price for the VIC IEEE-4888 interface is $75. A similar interface from OEM is $80.

A different type of interface is sold by Small Systems Engineering (1056 Elwell Court, Palo Alto, California 94303, phone 415-964-8201). Called Interpod, this $180 device connects to the VIC 20's serial port by a cable. Interpod has provisions for connecting Commodore disk drives and printers that use the IEEE-488 protocol to the VIC 20 as well as standard serial printers, plotters, and speech synthesizes which use the RS-232C standard. There are even provisions for selecting the correct serial speed (called BAUD rate or BPS—bits per second) from the interface.

Richvale Engineering (10610 Bayview, Richmond Hill, Ontario L4C 3N8, phone 416-884-4165) is another established manufacturer of hardware and software for the Commodore computer line. They produce a product called the RTC V-LINK, which functions in much the same way as the Interpod described above. Both provide an IEEE-488 interface for the VIC and both have standard RS-232 serial interfaces. V-LINK also has a parallel interface and additional software that lets you use the same disk drive commands that are used on the PET computers with 4.0 BASIC. The disk drive commands in 4.0 BASIC are much easier and more convenient to use than the ones available in Commodore 2.0 BASIC, the version used on the VIC 20 and Commodore 64 computers. V-LINK also has a program that makes it much easier to write and revise machine language programs. At $185 (Canadian), V-LINK is a lot for the price.

The concept of "networking" may not be familiar to many readers, but it is an important one if you plan to use several computers in one location. Under normal circumstances, if you planned to use eight VIC 20 computers in a classroom where computer litaracy courses were taught, you would need to provide at least one disk drive or one cassette unit for each VIC 20. Peripherals such as disk drives and printers are expensive; they generally cost much more than the computer. One way of saving on the cost of peripherals is to use a "resource sharing" network system. That means you can connect several VIC 20's to one or more disk drives and a printer. Since

the disk drive is actually used less than 2% of the time and printer usage generally averages less than 5% of a typical computing session, it makes sense to let the computer share these resources rather than provide a disk drive and printer for each keyboard. The VIC 20 is a rather forgiving computer and it is possible to unplug the disk drive and/or printer after use and plug it into another computer which needs access. We do not recommend this, however, because of the wear and tear it causes on connectors. There are, however, networking accessories that let you interface many VIC 20 computers to one or more disk drives and printers. One of the least expensive systems is the VIC/64 Switch which is manufactured by Computer Marketing Services (300 West Marlton Pike, Cherry Hill, New Jersey 08002). It sells for $150 and allows you to connect up to eight VIC 20 and/or Commodore 64 computers to the VIC/64 Switch. You then connect 1540 or 1541 disk drives and a compatible Commodore printer to the VIC/64 Switch and start computing. When one of the eight computers needs access to the printer or disk drive, the interface connects the computer to the peripheral and operation proceeds as if the disk or printer was connected directly to the computer. If another computer asks for access at the same time, the interface puts that computer in a "wait" state until the first computer finishes. This is a very inexpensive means of allowing shared-resource networking. The only expense involved is the VIC/64 Switch and cables from the computer to the interface ($10 to $20 each).

New Commodore Products for the VIC 20

As this book was written, Commodore announced several new products which can be used on the VIC 20. The products described here, however, were "announced" but not yet in production. Commodore has a reputation for announcing things well in advance of the time they are available and has been known to announce and advertise products and then never actually produce and distribute them. We would therefore caution you to consider the material here as gossip that may or may not come to pass.

Color Monitor. Commodore has recently begun distributing a 12" color monitor that is compatible with the VIC 20 and Commodore 64. The monitor has been specially designed to take advantage of all the video-control signals which are generated by the Commodore 64 and VIC 20.

Color Plotter. This is another aggressively-priced peripheral that works with the VIC 20 and Commodore 64. For $200, you get a printer/plotter that will print standard characters and draw high-resolution charts, graphs, and illustrations in up to four colors.

Music Keyboard. Until now, music synthesizers that had their own piano-style keyboard cost at least $600, usually more. Commodore plans to sell a music system with just such a keyboard for under $100. Early prototypes seem to be well-designed and come with very good software for creating and playing your own compositions.

Organizers and Useful Accessories

There are companies that manufacture cases, desk-top organizers, dust covers, and many more products for the VIC 20. Although items in this category are not absolutely necessary they can often make using your computer easier, more productive, and less time-consuming. Here are just a few of the items available:

Carrying Cases. Computer Case Company (5650 Indian Mount Court, Columbus, Ohio 43213, phone 800-848-7548) manufactures and distributes cases designed specifically for a particular computer. The case for a Commodore 64 or VIC 20 computer and 1540/1541 disk drive is $129. These are well-made cases that will protect the computer while it is being transported. These cases are not, however, designed to be shipping cases that can be checked as luggage on an airplane.

Another company, Southern Case Company (2315 Laurelbrook Street, P.O. Box 28147, Raleigh, North Carolina 27611, phone 800-334-0551), also manufactures cases for the VIC 20. Their case has room for the keyboard, a cassette recorder, the power supply, four cassettes, and two joysticks. It is a rugged unit made of twin-walled plastic with foam rubber between the two walls. The case costs $80 and seems to be strong enough to be used as checked baggage if it is absolutely necessary that the computer be transported in that fashion. Another model from Southern Case has provisions for the keyboard and a disk drive. It is also $80.

Another handy accessory for your computer is a desk organizer. We used the Micro Powerbench from Cab-Tek (Riverside Street, Nashua, New Hampshire 03062, phone 603-889-1961) and found it to be very useful. It slips over your computer and provides a place for the television or monitor to sit. There is also room for the disk drives inside the unit. The Micro Powerbench has an on/off switch and a power strip on the back where you can conveniently plug in the power supplies for your computer, several disk drives, and a printer. For an additional cost you can get a model that has a powerline filter and a fan. Prices for the Micro Powerbench range from $80 to $160, depending on the options. The same company also has printer enclosures that keep dust and debris out of your printer and also provides some sound isolation. The printer enclosures are $99 to $199 depending on the size of the printer.

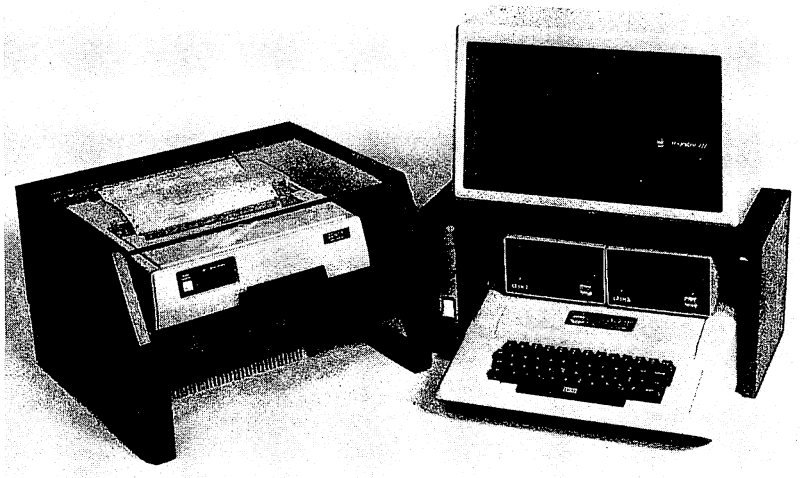


Figure 11.11 A micro power bench and printer enclosure from Cab-Tek.

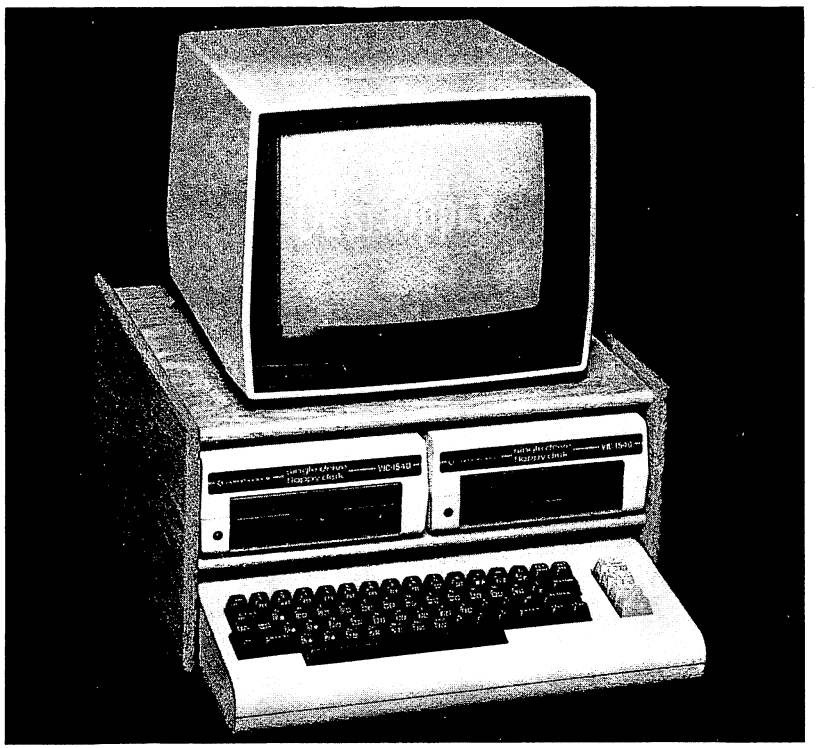


Figure 11.12 The DeskTopper from Madison Computers.

A less expensive but extremely useful desk organizer is distributed by Madison Computer (1925 Monroe, Madison, Wisconsin 53711, phone 608-255-5552). It retails for $50 and has a shelf for your disk drive, room for a manual or two and a flat top surface where the monitor can be placed.

There are hundreds of other products that will make using your VIC 20 more pleasurable. One of the easiest and most effective ways of keeping up with new accessories is to subscribe to one or more of the magazines that cater to owners of Commodore computers.

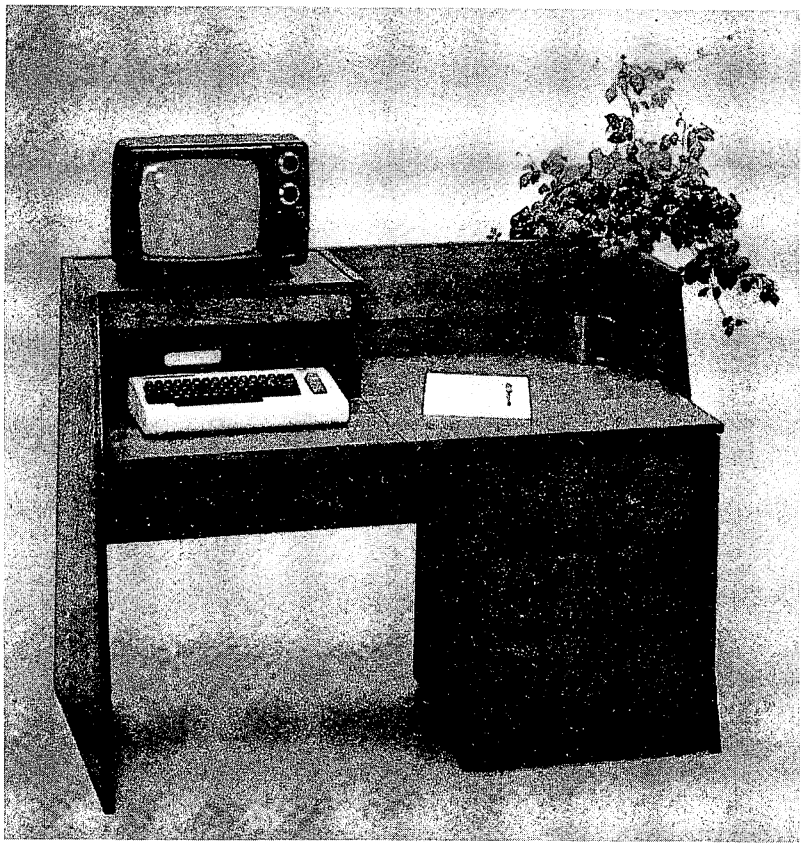


Figure 11.13 A Computer desk.

Chapter 12

Sources of More Information

We hope this book has served as a good beginning point to the use of your VIC 20 and to the field of personal computing. In spite of 11 chapters of information, however, we have only scratched the surface. There is more, much more, to be learned about your computer and about personal computing. This final chapter will provide you with some clues for finding the additional information you need.

Magazines

Compute! We feel this is an excellent magazine for owners of the VIC 20 computer. It is published monthly and carries articles on the Commodore computers as well as the ATARI, Apple, TRS-80 Color Computer, and the Timex Sinclair. It is well-edited, very nicely illustrated, and filled with programs you can type and run on your computer. Most issues contain at least two or three programs that will run on the VIC 20. In addition, there are all sorts of articles on how to use the VIC 20. Most of the articles are for intermediate or beginning users, but a few are quite advanced. *Compute!* also has several regular columns on topics such as educational computing and programming in Logo. This magazine has an excellent balance of editorial material. A typical issue will have articles on programming, reviews of commercial software and hardware, general articles on topics such as word processing, games, and the social implications of computers, and articles that include programs you can type and use. In addition, the magazine carries hundreds of ads for products you may want to buy. At $20 a year, this is a magazine we strongly recommend. There is a toll-free number for subscriptions (800-334-0868) or you can write them at *Compute!* Magazine, P.O. Box 5406, Greensboro, North Carolina 27403. The people who produce the magazine also publish a line of books. *Compute!'s First Book of VIC* is $12.95 and contains many of the articles published in the magazine plus additional chapters of new material.

Compute!'s Commodore Gazette. This magazine published its first issue in the summer of 1983. It is aimed at novice owners of the Commodore 64 and VIC 20 computers and has an article mix similar to *Compute!*. Subscriptions are $15 a year (12 issues). See above for phone and address.

PowerPlay. This magazine is published four times a year by Commodore and costs $10 a year. It is not as good as the two magazines above, but it is one of the best company magazines published. It is well-illustrated, publishes articles for beginning and intermediate users, and accepts ads for companies other than Commodore (something some company magazines do not permit). The focus of this magazine is on home and recreational computing. Each issue generally has one or two programs as well as articles on a variety of topics. Most issues also carry a list of active user groups for Commodore computers. You can subscribe by sending $10 to Commodore Business Machines, The Meadows, 487 Devon Park Drive, Wayne, Pennsylvania 19087.

Commodore. This magazine is also published by Commodore. It concentrates on professional and business applications with the more expensive Commodore computers, but the magazine also publishes quite a few articles on the VIC 20 and Commodore 64. It is $15 a year for six issues. Commodore currently offers a discount if you subscribe to both *Commodore* and *PowerPlay*. We think these are worth the subscription price.

Micro. This magazine publishes articles relevant to owners of computers that use microprocessor chips in the 6502 and 6809 families. All the Commodore computers are thus covered. It tends to focus on articles about programming your computer and on building accessories yourself. There are also many tutorial and general interest articles, however. The March, 1983 issue, for example, contained an article by Fred Wallace on high resolution plotting with the VIC 20 and the 1515/1525 printers. Well-written and edited, it is published 12 times a yar. Subscriptions are $24 from *Micro*, 34 Chelmsford Street, P.O. Box 6502, Chelmsford, Massachusetts 01824.

The Transactor. Like *Micro*, this publication is aimed at the more advanced user. Although not as large as *Micro*, all the articles are relevant to Commodore computers. While it publishes reviews of commercial software and hardware products, the focus is on technical articles that deal with programming issues and/or the hardware design of particular computer models. Many articles are about the VIC 20 and Commodore 64. It is published six times a year by Canadian Micro Distributors, 500 Steeles Avenue, Milton, Ontario Canada L9T 3P7. Annual subscriptions are $15.

Strictly Commodore. This bi-monthly magazine is not as flashy as some of those described above, but it carries lots of information for owners of Commodore computers. It is $18 a year for six issues. The address is 47 Coachwood Place N.W., Calgary, Alberta Canada T3H 2E1.

Commander. We have not been able to get a copy of this magazine but it is aimed at owners of Commodore computers. Subscriptions are $22 a year from Micro Systems Specialities, P.O. Box 98827, Tacoma, Washington 98498. They have a toll-free number for orders—800-426-1830.

Commodore Computing. This excellent British publication covers the entire line of Commodore products and has articles for beginning through advanced users. It is 12.5 British pounds a year if you live in England, 15 pounds if you live overeas. Write to *Commodore Computing*, MAGSUB, Oakfield Huse, Perrymount Road, Haywards Heath, Sussex RH16 3DH.

VIC Computing. This is another British magazine. All the articles in this magazine are on the VIC 20. Subscriptions are six pounds in England, nine in Europe, and 16 overseas. The address is *VIC Computing*, 39-41 North Road, London, England N7 9DP.

Fox 20. This "magazine" is a little different. It is published monthly on cassette. You load the cassette into your computer and read informative articles on the VIC 20 as they appear on your screen. In addition, each issue contains at least five programs for the VIC 20 that you can load into your computer from cassette and run. Many are games, but there is quite a variety. It is well done, interesting, and useful. Subscriptions are $53 a year from *Fox 20*, P.O. Box 507, Deer Park, Texas 77536. The phone number is 713-473-6723.

20 Load. Although we have not seen this publication it is also published monthly on cassette and appears to be similar in intent to *Fox 20* above. Subscriptions are $50 a year from *20 Load*, 550 Grant Avenue, Junction City, Kansas 66441. The phone number is 913-762-4730.

In addition to the specialized magazines mentioned above there are several other computer magazines that regularly publish material on the VIC 20 computer. Most of these are available on newsstands where you can browse through them to determine if they fit your needs. *Creative Computing* is a fat, fact-filled monthly that covers the general field of personal computing. It regularly publishes articles on the Commodore computers. Joel Swank writes a regular column in this magazine titled Commodore's Port. The information contained in this column alone is often worth the price of the magazine for VIC 20 and Commodore 64 owners. *Easy Home Computer* magazine also publishes material on the VIC 20 and is oriented

toward novice users. *Computers and Electronics* is another magazine that may be of interest to VIC 20 owners. Jerry Willis' book *Computers for Everybody* which is published by dilithium Press includes a description of over 20 other magazines that cover different aspects of personal computing.

Books and Book Publishers

A stroll through the computer section of your local bookstore is likely to show you just how popular the VIC 20 really is. There are at least 50 different books about the VIC 20 in current distribution. Few bookstores are likely to have more than five to ten of them, but there is an increasing tendency for bookstores to carry a large number of computer books. In this section we will not try to list all the VIC 20 books currently on the market. There are too many, and the list would be out of date in a matter of months since new books are appearing regularly. Instead, we will note some of the more active publishers of books on the VIC 20. The publishers noted are likely to be bringing out new books as well. Many of the new books each year are reviewed in the magazines listed above. In addition, several of the suppliers listed in the next section add new books to their catalogs each year:

dilithium Press (P.O. Box 606, Beaverton, Oregon 97075, phone 800-547-1842). dilithium is the publisher of this book and several more on the VIC 20. The company has over 100 books in print, most of them at the beginning and intermediate level, on personal computing. You can get a free catalog by phoning them at the toll-free number listed above. dilithium publishes several books on PET BASIC, the version used in the VIC 20 and also publishes *32 BASIC Programs for the VIC 20*, a book of programs.

Creative Computing Press (39 East Hanover Avenue, Morris Plains, New Jersey 07950, phone 800-632-8112). This company, now a division of the publishing giant Ziff-Davis, has a good line of books. Several of their publications will be of interest to VIC 20 owners. Sally Larsen's book "Computers for Kids" (the VIC 20 version is $6) is a Creative Computing book.

Reston Publishing Company (11480 Sunset Hills Road, Reston, Virginia 22090, phone 800-336-0338). Reston is a division of Prentice-Hall and is rapidly increasing the number of books it publishes for novice and intermediate computer users. A very popular Reston book is *Kids and the VIC 20* by Edward Carlson. Another is *Zap! Pow! Boom! Arcade Games for VIC 20* by Tim Hartnell and Mark Ramshaw. This company also publishes a popular arcade game for the VIC 20 called "Minor 3049er."

Howard W. Sams (4300 West 62nd Street, P.O. Box 558, Indianapolis, Indiana 46206, phone 317-298-5400). Sams is best known as a publisher of rather technical books in the area of electronics, including computers. Recently, however, Sams has begun publishing introductory and intermediate level books on personal computing. Sams publishes most of the manuals that are packed with the Commodore computers. It also publishes the *VIC 20 Programmers Reference Guide*, a $17 tome that includes a great deal of information on programming the VIC 20 in both BASIC and machine language. The company also publishes the *Commodore Software Encyclopedia* ($10) which lists hundreds of programs for the VIC 20 as well as other Commodore computers.

One book you may want to consider is *The Complete VIC* which is available from **Macro Dynamics** (8950 Villa La Jolla Drive, Suite 1200, La Jolla, California 92037) for $15. It contains descriptions of over 800 products for the VIC 20 as well as reviews of many products and references to articles on the VIC 20. The same company publishes *The Complete 64*.

In addition to the books and magazines we have already mentioned, there are two other excellent sources of information on your VIC 20 computer. Several of the "information utilities" mentioned in the first chapter of this book let you search databases for articles, software reviews and technical notes on the VIC 20. In addition, both the magazines published by Commodore print regularly updated lists of local user groups. Most of these groups have regular meetings and are happy to help the novice who is having trouble. One of the best known "local" user groups is Toronto Pet Users Group (P.O. Box 100, Station S, Toronto, Ontario Canada M5M 4L6). A yearly membership is $20. For that you will receive a very good club newsletter and access to the club's library of over 3,000 programs which are free to members (a small handling and copying charge is the only cost).

Suppliers

There are hundreds of companies that produce and/or sell products for the VIC 20 computer. If you are lucky, you will live in an area where one or more retail stores carry a wide range of products for the VIC 20. That, alas, is not the fate of most of us. Even when you read a positive review about a great new product or program for the VIC 20, it is difficult for many to buy it locally. A large number of stores simply do not carry many accessories.

One solution to this problem is to order products by mail. Magazines like *Compute!* carry hundreds of ads in every issue. Many products must be ordered from the manufacturer, but there are a few distributors who try to stock a wide range of products from many different suppliers. Some of these distributors even publish catalogs that describe the products they

carry. Often the mail-order supplier will sell products at a "discount" while the local stores do not. You can save money buying by mail, but remember that a local store often provides convenience, local support, assistance and advice for computer owners. In addition, if a product is faulty, it is usually much easier to correct the problem if you buy locally. We find ourselves buying some products from local suppliers and ordering some by mail. Below is a list of some mail-order distributors of VIC 20 hardware and software. Remember, however, that companies come and go at an amazing rate in this business. Current issues of magazines that cater to VIC 20 owners will also have ads for distributors.

Data Equipment Supply Corporation, 8315 Firestone Boulevard, Downey, California 90241, phone 213-923-9361.

Olympic Sales Company, P.O. Box 74545, 216 S. Oxford Avenue, Los Angeles, California 90004, phone 800-421-8045.

Mooseware Incorporated, P.O. Box 17868, Irvine, California 92713.

SJB Distributors, 10520 Plano Road, Suite 206, Dallas, Texas 75238, phone 800-527-4893.

Computer Outlet, Park Place, Upper Level, 1095 E. Twain, Las Vegas, Nevada 89109, phone 800-634-6766.

CompuSense, P.O. Box 18765, Wichita, Kansas 67218, phone 316-684-4660.

Programs International, Moravia Center Industrial Park, Baltimore, Maryland 21206, phone 301-488-7719.

Southwest Micro Systems, 2554 Southwell, Dallas, Texas 75229, phone 214-484-7836.

JMC, 1025 Industrial Drive, Bensenville, Illinois 60106. JMC's catalog is $2 and lists many books.

Universal Software, 185 Mulberry Street, Claremont, New Hampshire 03743, phone 800-343-8019.

Harmony Video and Electronics, 2357 Coney Island Avenue, Brooklyn, New York 11223, phone 800-227-8927.

Some Final Words

We hope you have found this book useful and that you continue to explore the possibilities of personal computing. If you have suggestions you feel would improve the next edition of the book, please write us at dilithium Press. Happy computing!

Appendix A

Abbreviations for BASIC Keywords

As a time saver when typing in programs and commands, VIC 20 BASIC allows the user to abbreviate most key words. The abbreviation for the word **PRINT** is a question mark. The abbreviations for the other words are made by typing the first one or two letters of the key word, followed by the SHIFTed next letter of the word. If the abbreviations are used in a program line, the key word will **LIST** in the longer form. Note that some of the keywords when abbreviated include the first parenthesis, and others do not.

Command	You press	This appears on the screen
AND	A SHIFT N	A
NOT	N SHIFT O	N
CLOSE	CL SHIFT O	CL
CLR	C SHIFT L	C
CMD	C SHIFT M	C
CONT	C SHIFT O	C
DATA	D SHIFT A	D
DEF	D SHIFT E	D
DIM	D SHIFT I	D
END	E SHIFT N	E
FOR	F SHIFT O	F

Command	You press	This appears on the screen
GET	G SHIFT E	G
GOSUB	GO SHIFT S	GO
GOTO	G SHIFT O	G
INPUT#	I SHIFT N	I
LET	L SHIFT E	L
LIST	L SHIFT I	L
LOAD	L SHIFT O	L
NEXT	N SHIFT E	N
OPEN	O SHIFT P	O
POKE	P SHIFT O	P
PRINT	?	?
PRINT#	P SHIFT R	P
READ	R SHIFT E	R
RESTORE	RE SHIFT S	RE
RETURN	RE SHIFT T	RE
RUN	R SHIFT U	R
SAVE	S SHIFT A	S
STEP	ST SHIFT E	ST
STOP	S SHIFT T	S
SYS	S SHIFT Y	S
THEN	T SHIFT H	T
VERIFY	V SHIFT E	V
WAIT	W SHIFT A	W
ABS	A SHIFT B	A
ASC	A SHIFT S	A
ATN	A SHIFT T	A
CHR$	C SHIFT H	C

Command	You press	This appears on the screen
EXP	E SHIFT X	E 
FRE	F SHIFT R	F 
LEFT$	LE SHIFT F	LE 
MID$	M SHIFT I	M 
PEEK	P SHIFT E	P 
RIGHT$	R SHIFT I	R 
RND	R SHIFT N	R 
SGN	S SHIFT G	S 
SIN	S SHIFT I	S 
SPC (	S SHIFT P	S 
SQR	S SHIFT Q	S 
STR$	ST SHIFT R	ST 
TAB (	T SHIFT A	T 
USR	U SHIFT S	U 
VAL	V SHIFT A	V 

Appendix B

Error Messages

Here is a list of the error messages generated by the VIC 20 with a description of the causes.

BAD DATA String data was received from an open file, but the program was expecting numeric data.

BAD SUBSCRIPT The program was trying to reference an element of an array whose number is outside of the range specified in the DIM statement.

CAN'T CONTINUE The **CONT** command will not work, either because the program was never **RUN**, there has been an error, or a line has been edited.

DEVICE NOT PRESENT The required I/O device was not available for an **OPEN**, **CLOSE**, **CMD**, **PRINT#**, **INPUT#**, or **GET#**.

DIVISION BY ZERO Division by zero is a mathematical oddity and not allowed.

EXTRA IGNORED Too many items of data were typed in response to an INPUT statement. Only the first few items were accepted.

FILE NOT FOUND If you were looking for a file on tape, and END-OF-TAPE marker was found. If you were looking on disk, no file with that name exists.

FILE NOT OPEN The file specified in a **CLOSE**, **CMD**, **PRINT#**, **INPUT#**, or **GET#** must first be **OPEN**ed.

FILE OPEN An attempt was made to open a file using the number of an already open file.

FORMULA TOO COMPLEX The string expression being evaluated should be split into at least two parts for the system to work with.

ILLEGAL DIRECT The **INPUT** statement can only be used within a program, and not in direct mode.

ILLEGAL QUANTITY A number used as the argument of a function or statement is out of the allowable range.

LOAD There is a problem with the program on tape.

NEXT WITHOUT FOR This is caused by either incorrectly nesting loops or having a variable name in a **NEXT** statement that doesn't correspond with one in a **FOR** statement.

NOT INPUT FILE An attempt was made to **INPUT** or **GET** data from a file which was specified to be for output only.

NOT OUTPUT FILE An attempt was made to **PRINT** data to a file which was specified as input only.

OUT OF DATA A **READ** statement was executed but there is no data left un**READ** in a **DATA** statement.

OUT OF MEMORY There is no more RAM available for programs or variables. This may also occur when too many **FOR** loops have been nested, or when there are too many **GOSUB**s in effect.

OVERFLOW The result of a computation is larger than the largest number allowed, which is 1.70141884E + 38.

REDIM'D ARRAY An array may only be **DIM**ensioned once. If an array variable is used before that array is **DIM**'d, an automatic **DIM** operation is performed on that array setting the number of elements to ten, and any subsequent **DIM**s will cause this error.

REDO FROM START Character data was typed in during an **INPUT** statement when numeric data was expected. Just re-type the entry so that it is correct, and the program will continue by itself.

RETURN WITHOUT GOSUB A **RETURN** statement was encountered, and no GOSUB command has been issued.

STRING TOO LONG A string can contain up to 255 characters.

SYNTAX A statement is unrecognizable by the VIC 20. A missing or extra parenthesis, misspelled key words, etc.

TYPE MISMATCH This error occurs when a number is used in place of a string, or vice-versa.

UNDEF'D FUNCTION A user defined function was referenced, but it has never been defined using the **DEF FN** statement.

UNDEF'D STATEMENT An attempt was made to **GOTO** or **GOSUB** or RUN a line number that doesn't exist.

VERIFY The program on tape or disk does not match the program currently in memory.

Index

How to Use The VIC 20™ Computer

Here are all the secrets you need to know to successfully and happily operate the VIC 20 Computer. *How to Use the VIC 20 Computer*:

- ☑ Introduces you to the computer and its basic components.
- ☑ Tells you what the components do and how they work together.
- ☑ Gives you step-by-step instructions on how to set up or install the VIC 20 and how to get it up and running.
- ☑ Shows you how to load and save your programs on diskette or standard audio cassettes.
- ☑ Tells you how to type in, use, and modify programs published in books and magazines.
- ☑ Presents you with other sources of information about the computer such as magazines, books, and users groups.

With the emphasis on practical information, this guide will be your constant companion as you learn to effectively use the VIC 20 computer.

ISBN 0-88056-134-3

>>$3.95

dilithium Press
GENERAL INTEREST