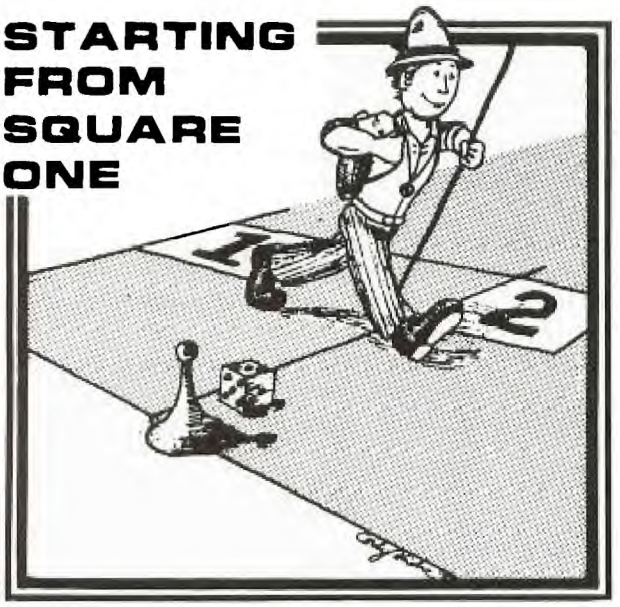
1
Starting Out

1

Starting Out

Taking the first steps on an exciting adventure!

STARTING FROM SQUARE ONE



How to Buy A Computer

". . . be assured that you are embarking on an exciting adventure . . . and realize that ownership is not only exciting but helpful and productive too."

In this article I will offer some suggestions to help those of you shopping for your first personal computer. I will not directly compare brand names, nor will I attempt a technical critique of the TI-99/4A home computer—but I will point out some of the TI machine's exceptional features.

What follows is a general discussion of computer shopping techniques written by and for the computer novice who is experiencing the bewilderment of trying to make a wise purchase in a market exploding with products. I offer these suggestions from the perspective of a writer who is not a computer professional. I have owned a TI-99/4 for a year and a half and also recently bought a competitive brand computer. In addition, I plan to purchase a third brand during 1982. Therefore, I am not dedicated to a single brand of computer although I am impressed with the 99/4A capabilities. All of my comments apply equally to the 99/4 and the new 99/4A unless otherwise noted.

My computers are used to develop computer-assisted instruction (CAI) for applications in the field of rehabilitation. The following suggestions result from the actual experiences of a beginner faced with the task of learning about computers—one who has spend literally hundreds of hours poring over manuals and magazines, and peering into a monitor screen.

Because my background is in psychology and counseling I can't resist beginning with some general, facilitative remarks. First of all, no matter which computer you eventually buy, you will later regret your choice at times. No one computer will have all the features you want; you'll have to compromise on some features—just remember that the grass always looks greener. . . so be aware that your buyer's anxiety may not totally disappear the instant you take possession of your new computer.

Secondly, regardless of how impressed you are with your new computer's gee-whiz features, you will quickly adjust your expectations upward. Whatever you buy now you will probably soon want to expand, with either more hardware (machinery/gadgets) or more software (programs).

Thirdly, start now! Don't wait for computers to come down to $9.98—they probably never will. The manufacturers will just keep on making them more sophisticated for about the same money.

And last of all, don't expect your friends, spouse, etc., to be as thrilled as you are about your computer. It is up to *you* to educate *them*.

Who Buys A Personal Computer

Rumor has it that someone once tried to profile the "typical" personal computer buyer for more effective marketing strategies. The survey data showed one shared factor: The majority of buyers wanted to become rich by writing and marketing a very successful program. In other respects, they are all different, and are using their computers for myriads of different purposes. So you're not alone when you go out to buy a computer—you may even find yourself in one of the following categories:

Type 1—The electronics amateur who is intrigued by all the technology. Fiddling with the equipment is enough reason for him to buy. We should be grateful to him: When he began buying kits and tinkering around with them a few years ago, he started the home computer craze.

Type 2—The aware parents who want the family to be up on "the latest." The family can play games and learn about computing as well as do the budget, and so on. The average family will want a computer that is flexible, versatile, inexpensive and "friendly" (easy to use). It should be expandable so that it can grow as the family's needs grow. This market has yet to peak.

Type 3—The small business owner or professional person who wants to automate the office. He will agonize over how much computer to buy. If it's not enough, it could well become merely a toy for his kids, but why buy a $10,000 system if a $3,500 package will do the job? This system will probably need both large amounts of data storage capacity and word processing capabilities.

Type 4—The educator interested in computer-assisted instruction (CAI). He or she will need a computer capable of displaying eye-catching color graphics and animations along with text, speech and sound.

Type 5—The scientist or engineer who will use the machine at work or home because it is easier than standing in line to get on the company's big main frame computer. Even companies that own big "braniac" computers are buying micros to spread around to key people.

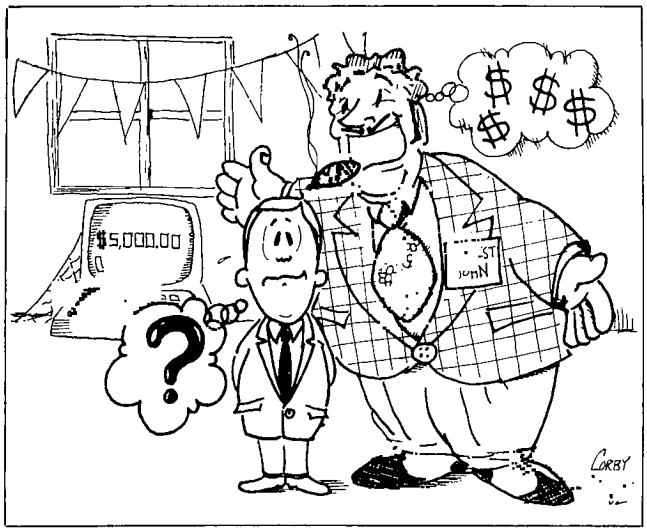
The list could go on and on, but I hope I have made my point that the "typical" microcomputer buyer is anything but typical. We all have one thing in common, however.

We have all been bitten by the computer bug and the only known cure is to take the plunge and get our very own microcomputer!

Types of Sellers

If you as a buyer are feeling overwhelmed by all the computer choices, pity the typical salesperson. He may be more at home with stereos and televisions, and entirely new to computers. Or he may be a programmer or technician entirely new to selling. Odds are that you'll meet the former more often at your local computer outlet; just as buyers exist at every level of sophistication, so do sellers. More important than knowledge of computer technology, though, is the willingness of the computer salesperson to help you learn. After all, we're all new to personal computing.

If you haven't already, you will shortly encounter at least one of the following salespeople:



Type 1—The sincere young man or woman who produces a nervous smile and confesses, "I only started in this department yesterday; let me see, where is the power switch on this little beauty. . ." Don't leave too soon, though. If you've the time and patience, you and the trainee can learn a lot about the computer in an hour.

Type 2—The equally sincere salesperson who introduces himself and says, "What can I show you. . .we have a 48K whiz-banger with a double DOS and CP/M on special. . ." This individual will joyously prattle on until your glazed eyeballs communicate either lack of interest or comprehension. (They are equivalent in the clerk's opinion.) You can then leave the store with a handful of pamphlets and a heart full of doubt—and possibly a car full of computer.

Type 3—The merchandising expert who moves computers the same way he used to move TV sets, stereos, etc. This type cannot refrain from knocking the competition by saying things like, "Brand X is almost out of business, that's why we don't handle 'em. . .what'd ya' say you do for work. . .I sell a number of Crunchy 100's to people in your field." This individual may be able to tell you a lot about his computer since he will be shrewd enough to read up on all the features of his machine; you may actually learn something if you have the confidence and patience to endure a barrage of irrelevancies.

Type 4—The skilled and sensitive sales professional who has developed a good knowledge of computing, or vice versa, the computer professional who has developed basic competence as a salesperson. This person will ask you right off what you want to do with your computer and help you with the answer if you aren't sure. You will appreciate this individual's patience and willingness to find out information for you. He or she will consult with a superior or even call the manufacturer without fear of appearing ignorant. When you meet people like this, respect their time and effort and show your appreciation. We don't want them to get discouraged and switch jobs. There will be little danger of this, however, since they will probably be making a lot of sales with many happy customers!

How To Shop

Be careful not to equate the amount of advertising you see for a computer with its technical sophistication or suitability for your needs. Take the time to go beyond mere advertising when you shop. Talk to computer owners, or visit a local computer club. But remember to expect some very prejudiced views, because people always try to convince themselves that *their* choices are best. Be cautious, too, of magazine reviews of various computers. Articles with extensive charts and diagrams may look impressive, but they are sometimes simply wrong. I have read articles which declared that the TI machine had no high-resolution color graphics or memory expansion capabilities. Well, TI has one of the best high-resolution color graphics capabilities on the market and can be expanded to a 48K system. I have noticed similar errors on other brands as well.

So visit several stores, read a few computer magazines, like *Home Computer Magazine*, and get your confidence up so the salesperson won't intimidate you. I am impressed with the TI-99/4A as I grow more familiar with it, but very little of this knowledge came from advertising or from salespeople: It came from *use* of the machine.

You may also need to know a little computer jargon, although the better salespeople will avoid trying to impress you with their vocabulary. If you don't already have one, pick up a glossary of terms while you are out for your first visit to the computer store. For starters, you should study the accompanying glossary for an understanding of its terms: With just a few of these terms tucked away in your memory banks, you can walk into the computer store with more confidence and less quiver in your voice when you ask to see the "Brainiac 3000" computer.

Ask to see a demonstration of each computer you think you can afford. But be aware that many demonstration programs you see are written in a program language other than BASIC—i.e., the language available to the user on most small computers. Consequently, the demo may be super impressive with lots of color graphics, animation, and sound, but find out if you can duplicate these effects readily with the BASIC programming language available to you. If you are interested in having good color graphics in your programs, ask the salesperson to enter some simple statements in BASIC to illustrate the computer's ability to perform the following:

A. Clear the screen.
B. Change the screen color.

C. Plot the color shapes on the screen. Try to place a "duck" or a "car" on the screen. Find out if the user can create his own shapes or is he limited to pre-defined shapes stored in the computer's memory.
D. Place a graphic shape and text on the same screen. Some computers can do one or the other without elaborate and difficult programming.

Happily, the 99/4A does all of the above with ease. You can program in 16 colors with simple, easy-to-use BASIC statements. If graphics are important to you, check out the TI Extended BASIC graphics capabilities. They are sensational and compete with computers costing as much as a thousand dollars more. If you want sound capabilities in your program, ask for a demo of the following:

A. Play a three note chord.
B. Play a simple scale.
C. Demonstrate the highest and lowest frequency programmable.
D. Demonstrate the loudest and softest volume of sound possible.
E. Create sound effects like a "choo-choo" or an "explosion."

Speech synthesis adds an exciting dimension to computing, especially in educational programs. Texas Instruments makes it easy to integrate speech into BASIC progams with its speech synthesizer and *Speech Editor* Command Cartridge or the *Terminal Emulator II* Command Cartridge. The *TE-II* will synthesize any English word typed into the computer; the *Speech Editor* will allow you to choose from a vocabulary of over 300 words. By all means get a demonstration of speech synthesis if you are interested in computer-assisted instruction—it is well worth the added cost.

The Editor

Regardless of the type of use you plan for your computer, you will definitely need a good editor. However, if you can think and type without errors, you can skip this section and not worry about editing.

Good, you are honest! I found out the importance of an editor the hard way. Not one salesperson mentioned this feature in any of my shopping except to say that I could correct errors. From this treatment of the subject, you might conclude that all editors are alike. The galaxy of differences between computer brands and their editing capabilities can make them either a joy or a pain to use.

So, what is an editor? Somewhere buried in all that fabulous circuitry is a component which interprets all of the instructions you type in. It turns your instructions—words—into the ones and zeros that the computer understands. It interprets the program for the computer. It will also edit or change, program statements after they have been entered into the computer. When you are writing and debugging (removing errors from) programs, you are bound to make typing errors. Typing the whole line over would correct these, but it is very time consuming and irritating, especially when there may only be one or two mistakes in 25-50 characters! If you could only correct the mistakes without disturbing the rest of the line.

You can: A good editor will permit you to modify a line of a program by inserting or deleting characters or words with a single keystroke, while displaying the changes on the monitor screen exactly as made in the program. A poor editor will require multiple keystrokes, and won't display the corrections as they are made. It will make you pound many keys and ultimately resort to retyping. The TI editor is far superior to my Number 2 computer's editor, and is equivalent to a good word processor in its correction capabilities. (I am writing this article on my 99/4 using a simple word processing program I wrote myself. It uses all the editing features resident in the computer and works very well for editing text.)

I cannot overemphasize the importance of the editor, and strongly recommend that you evaluate it carefully before you buy. Sit down at the keyboard and have the salesperson walk you through some editing. Don't let the clerk do it because he may pick simple tasks to make it look easy.

For instance, you might enter this program line;

```
100 PRINT "NOW IS THE TIME FOR ALL GOOD
MEN TO COME TO THE AID OF THERE
COUNTRY."
```

(If you are new to programming, let me point out that this BASIC statement will cause the words inside the quotes to be displayed on the monitor if you RUN the program.)

Notice that the word *THERE* is mispelled; so correct the spelling without retyping the entire line, then insert the word *BEST* before the *TIME*. If you can't accomplish this by the store's closing time, ask the salesperson to do it; if he can't do it with ease, give serious thought to buying another brand of computer.

While you are at this, ask the salesperson to demonstrate *resequencing* for you. Resequencing is a simple but valuable (and frequently unavailable) feature which permits you to renumber your program line numbers in order to insert additional lines into an existing program if necessary. For example, you might type in this simple BASIC program:

```
10 PRINT "HELLO"
11 PRINT "WHAT IS YOUR NAME?"
12 INPUT N$
13 PRINT "THANK YOU ";N$
14 END
```

Notice that you don't have any room between lines for additional lines. If you later decide to change the program, you either have to type the program over or resequence the line numbers to provide space. Normally, you don't intentionally get yourself into corners where it is necessary to resequence your programs, but it does frequently happen (courtesy of Murphy's Law). On the TI machine, resequencing is easily accomplished by typing RES and pressing the ENTER key. Presto! The program looks like this.

```
100 PRINT "HELLO"
110 PRINT "WHAT IS YOUR NAME?"
120 INPUT N$
130 PRINT "THANK YOU ";N$
140 END
```

Now you can add additional lines between the original ones. Many computers do not have the resequencing function built in so you have to load in a separate program from a disk or tape. This function is important enough that it should be built into the machine as it is in the TI-99/4A.

General Considerations

Regardless of the sophistication of the system, you should expect certain fundamental "creature comforts." First, it

is mandatory that the screen be clear and easy on the eyes. You may not fully appreciate this during a brief demonstration in the store, but spend an hour or two peering at the screen in your basement, and you'll know what I mean. Your 19-inch TV at home may not display characters as sharply as the store's 9-inch monitor. On a big screen, the characters may appear more ragged because the dots composing the characters are larger and more spread out. Instead of white characters on a black background or vice versa, the TI has exceptionally sharp black characters formed by an 8 × 8 dot matrix with a pale blue background. It's also possible to change the characters and background to any of 16 colors.

My only criticism of the TI display capabilities is that with TI BASIC it is limited to a line of 28 characters for text or 32 for graphics. [With TI's *Editor/Assembler* or TI-Writer Command Cartridge, you have a 40 character "window" which automatically scrolls horizontally across an 80-column "page." The Video Display Processor chip inside the computer actually has a 40-column "text mode," and the software produces the doubling effect.—Ed.] Some computers display fewer, but many display lines up to 80 characters or more. My Number 2 computer displays 40, but I see little practical difference between it and the TI machines. However, the 80-character display and lower-case characters are desirable if you plan to do extensive word processing (letters, reports, etc.). The TI-99/4A has a type of lower-case which is actually compressed upper-case; it works very well. You can do word processing with a 28 character format, but you won't be able to see the text on the screen exactly as it will appear on the printed page; with an 80 column format, however, you will. Your printer should have the capability of printing both upper- and lower-case characters with the proper program, so that you need not worry about having lower-case resident in your computer.

Another "creature comfort" to consider is the computer keyboard. The original TI-99/4 was criticised for having a keyboard smaller than a conventional typewriter. Actually it is very easy to use and one can touch-type on it very efficiently. But TI modified the keyboard on the TI-99/4A so it is more like a standard typewriter and added some function keys and a repeat key function to improve the computer's flexibility.

If you select a disk system for program and data storage rather than a cassette tape system, you will have the advantages of speed and convenience but you will sacrifice something, too. In addition to the higher cost of the disk system (maybe 10 times the cost of a tape recorder), you will also lose some of your program space (random access memory, known as RAM) inside the computer. Some systems will have a 2K overhead (2000 bytes) while others may require 10K or more. It is desirable to have a low overhead so that your valuable program memory space will be available for programs. The 99/4A disk system digests about 2K of your RAM leaving a nominal 14k for programs (on the standard 16K system). To put this into perspective, one page of typed, double-spaced material with liberal margins is equivalent to about 2K of information. If you buy a 16K computer which has a 10K overhead for the disk system, you would only have about 6K of program space after you turn on the disk system. And one of the very popular computer brands actually has a 10K overhead! So when you look at one in a store the salesperson will probably insist that you get (and pay for) at least 32K of RAM. Moral: 16K memory in computer "A" does not necessarily equal 16K memory in computer "B." Texas Instruments gives you a lot of memory for the money.

How much memory will you need in your computer? For most home use, a 16K computer is generally considered a satisfactory start. For business and educational applications you will probably need more memory—48K is satisfactory in most cases. That covers your program requirements inside the computer. For permanent storage of large amounts of data such as student grade records and inventory reports, you will use disk or tape. Such storage is relatively cheap. A diskette (called a "floppy disk" because it is flexible plastic) can store 90K or more of information on a 5¼inch surface costing a mere four or five dollars. You can store the equivalent of about 50 typed pages on one such disk. Cassette tape is okay for home use and for back-up copies of your disk data, but is generally too slow for serious business or educational applications.

Service

Check out the service policy on your computer before you buy. Some manufacturers will exchange defective components, and others want to repair and return the original unit. If *downtime* is critical to you, choose the system which can be replaced in the shortest time. My 99/4 developed intermittent problems after more than a year of very heavy use, and TI exchanged it for a factory rebuilt unit for only $45.00 with same day service and no questions asked. If trouble develops during warranty, the exchange charge is minimal. When I thought I had a defective disk system during the third month of ownership, the service center would have exchanged the entire disk system for about $3.50, but as it turned out, I had a bad diskette instead.

Where to Buy

Deciding where to buy your computer can be difficult. Should you buy from a local computer store, a department store, or perhaps from a mail-order outlet? You can get some terrific bargains from a mail-order firm. You'll see dozens of ads in any computer magazine and nearly all will accept credit cards, making it very easy to buy. I saved nearly 40% on my TI machine buying it from a firm in another city across the state; my Number 2 computer cost almost $500 less from out-of-state company than from the local store. The argument for buying from a local dealer and paying more is that you can count on better personal service if your machine goes on the blink. This may or may not be true depending on your dealer's integrity and quality of service. You could buy locally and still have problems with service. In my opinion, the overhead of the local computer store justifies the higher prices. If you can afford it and desire peace of mind, buy locally.

In the case of TI computers, you can exchange the defective unit for a factory rebuilt unit at one of the exchange centers. It won't matter where you originally purchased the unit. You can check with your local dealer to see if a service center is near you.

Another point to consider is that we really should not abuse the local computer store owner's time by letting him educate us if we have no intention of buying locally. It is fair to expect him to compete with other dealers for our dollar by demonstrating his wares and services, but unfair

to sit through an hour or two of free demonstrations if we've already decided to buy through the mail. After all, we want the computer store to succeed, since it will advance personal computing in general.

Miscellaneous Points

Ask the salesperson if the computer you select can perform the graphics, sound, and text functions you desire just as it comes out of the box, or must you buy additional attachments or plug-in devices. You may find the demonstration you witnessed on a "loaded" floor model cannot be performed on a basic unit without adding several hundred dollars of additional equipment. On the other hand, you may find that most of the desirable features are built right into the basic computer.

It is also essential to have clear, concise, easily understood manuals which explain how to use your computer. You should not have to have any knowledge about computers to understand the basic introductory and tutorial manuals for your computer.

If you have not yet bought that first computer, be assured that you are embarking on an exciting adventure. The excitement and pride you'll experience when opening the box on the first day is like a dozen Christmas celebrations combined. Enjoy the experience, and realize that ownership is not only *exciting* but *helpful* and *productive* too.

In the meantime read all you can and shop carefully until you just can't stand it any longer. . .then take the plunge. Go out and get that computer!

Glossary of Terms

BASIC—Beginners All Purpose Symbolic Instruction Code is a program language developed at Dartmouth in the early 60's; it is the most common of all programming languages for small computers. BASIC is relatively easy to learn and is an effective and powerful language for most small computer applications.

bit—The smallest piece of information your computer deals with. It is equivalent to a circuit being turned either on or off. Like a light bulb, a computer logic circuit is either on or off; this equals one bit of information. Most home computers use an 8-bit microprocessor, but Texas Instruments and IBM have a 16-bit microprocessor. The advantages of the 16-bit configuration are too technical for this discussion, but we can generally say that more powerful and accurate computing can be accomplished. It has been predicted that the 16-bit microprocessor will be the future industry standard.

byte—The amount of memory necessary to code a character (a number/letter/punctuation, etc.) A byte has 8 bits in it. A computer which has 16K bytes of memory has 16 thousand bytes and can work with about 16 thousand characters of information in a single program.

chip—The circuits of the computer are fabricated on silicon chips. A chip is typically about 1/4 inch on a side. Today's chips are so sophisticated that the basic components of an entire computer can be fabricated on a single chip.

CRT (monitor)—The TV-like screen (cathode ray tube) to which the computer outputs information like numbers/letters/graphs, etc.)

disk drive—The accessory which stores and retrieves information on plastic (mylar) diskettes. The DOS (see below) controls the operation of the disk drive.

disk operating system—Sometimes called DOS and sometimes pronounced like "DOSS." It is the set of instuctions (software) which controls the storing and retrieving of information with the disk drive.

diskette—A plastic disk coated with an oxide upon which data and programs are stored using the disk drive under control of the DOS. Diskettes come in either of two sizes, 5 1/4 inch or 8 inch. The TI-99/4A uses the 5 1/4 inch.

firmware—Generally speaking, firmware is a chip in which a program has been stored permanently. It is "soft" in that it is a program (see software) but "hard" to the extent that it is an electronic chip rather than a diskette or tape. Hence it is "firmware." Firmware is used to store programs which are used repeatedly, and need not be changed or modified. (see ROM)

hardware—The actual physical machine, i.e., keyboard, CRT, printer, etc.

integrated circuit (IC)—If you look into the back of an old radio, you will see a lot of resistors, capacitors, and the like. Each component will be discrete—i.e., separate from other resistors, etc. which surround it. Integrated circuits, on the other hand, have many such individual components packed together or integrated in a small area. (See chip.) If you peer into a computer, you will see rows of little black boxes plugged into circuit boards. Each little black rectangle may have thousands of components integrated into it.

input/output (I/O)—Input is the data that goes into the computer via the keyboard as well as disk drives, tape recorders, etc. Output is what comes back out of the computer to the monitor screen, disk drive, tape recorder, and printer. (Throughput is what happens in between).

microcomputer—All computers used to be very large and esoteric and were called "mainframes." But miniaturization with integrated circuits has resulted in very powerful computers of small size coming into being. That is, you could pack a lot of computer into a very small box. These computers were initially called "minicomputers." But as the reduction in size continued, small desktop-size computers were produced with sufficient computing capacity to still be very useful. These are called "microcomputers." The difference in power between the mini and the micro is diminishing rapidly, so that it will soon be difficult to tell a mini from a micro. For now, all home computers are considered microcomputers.

modem—A device that connects your computer to the telephone so you can communicate with other computers. It works by MOdulating and DEModulating a sound tone.

peripherals—All those hardware devices which plug into your computer such as disk drives, tape recorders, printers, and modems.

printer—A peripheral device which will print a copy (called hardcopy) of your computer's output. Very handy to have for correspondence and for program debugging.

program—The set of coded instructions which directs the activities of your computer. Without a program, your computer is just so much metal and silicon junk. (See software and BASIC.)

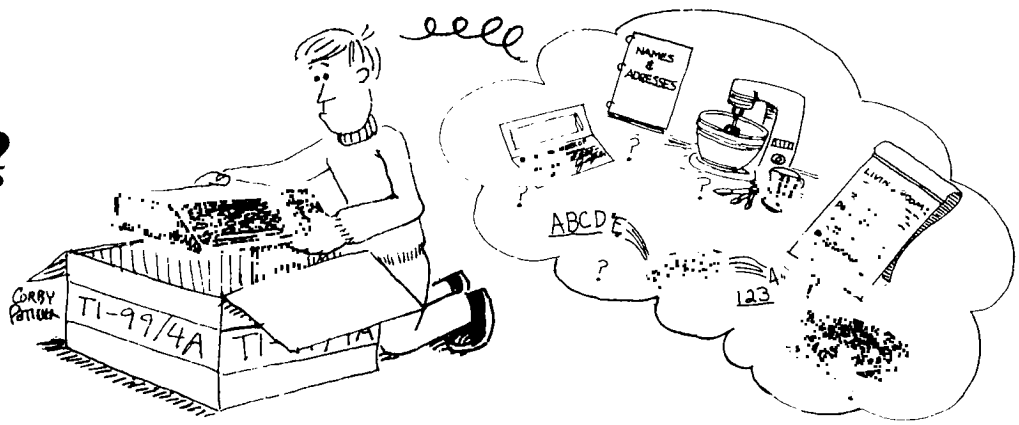
RAM—Traditionally, the abbreviation for random access memory. But the name is a little misleading. Both RAM and ROM memory are random access. More accurately, RAM should be described as read and write memory (contrast with ROM). RAM is the memory you are using when you program a computer. It is also the memory to which your computer salesperson is referring when he says, "This one has 16K memory." The more RAM you have, the bigger programs you can run. When you turn your computer off, all the contents of RAM is erased. So if you wish to avoid having to type in hundreds of program lines everytime you use your computer, you must save programs on tape or disk for future use.

ROM—This is read only memory. That's right, you cannot "write" anything to a ROM; you can only "read" it. This means that you cannot change the contents of a ROM memory like you can a RAM memory. ROM contents are usually not changed; therefore they are used for *firmware*.

RS-232C—A common interface specification used to define the link between the computer and some other device like a modem or a printer.

software—It is not the physical machine (hardware) and usually not the permanent programs stored on chips (firmware) that instructs the computer on how to perform a task. It is the program stored on disk or tape. You can see that a tape or plastic disk is not as much a part of the computer as a chip (not as "firm"); therefore the programs stored on tape or disk are called "software."

NOW WHAT?



Congratulations, you're the new owner of a TI-99/4A Home Computer!! Now what? You have it all unpacked and need to know what to do with it, right? Fortunately, you have *The Best of 99'er*, and we'll give you a few ideas to start you on your way.

Of course you can plug in a variety of Command Cartridges that can teach you exercise, challenge you to a chess game, help with your finances, or do a multitude of other things. But the real fun and challenge is making that machine do what *you* want it to do.

When I got my computer, many of my friends asked, "Well, what can it do?" And the next questions were: "Can you balance your checkbook with it?" "Can you file names and addresses?" "Can you keep track of other things such as household inventories?" "Can you do your income taxes?"

The TI-99/4A is so versatile that you *can* do all of these home applications plus a myriad of business and professional applications. You'll soon be "hooked" on your computer and be one of those computer nuts who stay up all night saying, "I'll just make *one* more change in this program, and then . . . ."

Let me just give you a few ideas for programming and then you'll be on your own.

Most households own a calculator. Now, with a calculator you just punch in numbers and symbols and get an answer. Your computer can manipulate numbers too, but it can also interact with you, using words. And it can do the same process over and over again. You can also save your program and the data and use it again a month or a year later. You'll soon find your computer is a valuable household addition.

To make an interactive program you'll need to use PRINT or DISPLAY and INPUT. PRINT and DISPLAY do the same thing on the screen in TI BASIC. You probably have discovered how to PRINT messages, so let me just give you one hint here. A colon in a PRINT (or DISPLAY) statement means, "Go to the next line." The screen will be much easier to read if you have a few spaces here and there rather than all the printing jammed up. You may use more than one colon in the statement to get more blank lines. Here's an example:

```
100 CALL CLEAR
110 PRINT "THIS IS A SAMPLE."
120 PRINT : "HELLO" : "HOW ARE YOU?" ::::
130 PRINT "START SPACING HERE."
140 PRINT : "I SKIPPED ONE LINE."
150 PRINT :: "I SKIPPED TWO LINES."
```

I usually start a program by clearing the screen. Line 110 prints a message. You'll notice the line actually prints then moves up one line. The first colon in Line 120 says, "Go to the next line," then print HELLO. Another colon—so HOW ARE YOU? starts on the next line, then you "go to the next line" four times. The number of blank lines is the number of colons at the end of the line, minus one. If the colons are at the beginning of a statement, the number of lines is equal to the number of colons. Don't get confused—just RUN this program and experiment a little to learn how to use the spacing effectively.

INPUT is how you enter something from the keyboard while the program is running. You may PRINT a message and then INPUT like this:

```
100 PRINT "WHAT IS YOUR NAME?"
110 INPUT NAME$
```

Remember, string variables need $ at the end of the variable name; numbers do not. This program will print the message, then print a question mark on the next line, blink the cursor and wait for the user to enter something.

INPUT also allows a prompting message:

```
100 INPUT "WHAT IS YOUR NAME?":NAME$
```

This time the cursor will blink in the space immediately following the prompt message and print your response there as you key it in.

When programming responses, you generally use INPUT. However, on a one-stroke answer I like to use CALL KEY. The user will just have to press one key (won't have to press ENTER), *and* you can block out unacceptable answers. For example, suppose you need a yes or no answer.

```
400 PRINT "ANSWER Y OR N"
410 CALL KEY(0,KEY,STATUS)
420 IF KEY=78 THEN 500
430 If KEY<>89 THEN 410
440 Continue here for "Yes" answer
.
.
.
500 Continue here for "No" answer
```

Only Y and N are accepted; any other key pressed is ignored. Another example is:

```
400 PRINT "CHOOSE 1, 2, 3, OR 4"
410 CALL KEY (0,K,S)
420 IF K<49 THEN 410
430 IF K>52 THEN 410
440 ON K-48 GOTO 1000,2000,3000,4000
```

Only 1, 2, 3, or 4 will be accepted, then the program will branch to the appropriate section. Remember that the K value in the CALL KEY statement is the ASCII code number of the character pressed.

Now you are armed with some basics of interactive programming. Let's try some specifics and answer those questions above.

Checkbook Balancing

Ah-ha! That's already in your TI-99/4A *User's Reference Guide,* page III-22. Just key that program in and add your own embellishments to make it *your* program. I like to take advantage of TI's color and sound to enhance a program, so let's add a little color at the beginning. Add:

```
115 GOSUB 500
```

This means go down to Line 500 and do some stuff then come back. Now add Lines 500 to 660. A complete modified listing follows. Try it. Then adapt it to what you want.

```
100 REM  CHECKBOOK BALANCE
110 CALL CLEAR
115 GOSUB 500
120 INPUT "BANK BALANCE: ":BALANCE
130 DISPLAY :"ENTER EACH OUTSTANDING"
140 DISPLAY "CHECK NUMBER AND AMOUNT."
150 DISPLAY
160 DISPLAY "ENTER A ZERO FOR THE"
170 DISPLAY "CHECK NUMBER WHEN FINISHE
    D."
180 DISPLAY
200 INPUT "CHECK NUMBER: ":CNUM
210 IF CNUM=0 THEN 250
220 INPUT "CHECK AMOUNT: ":CAMT
230 CTOTAL=CTOTAL+CAMT
240 GOTO 200
250 DISPLAY "ENTER EACH OUTSTANDING"
260 DISPLAY "DEPOSIT AMOUNT."
270 DISPLAY
280 DISPLAY "ENTER A ZERO AMOUNT"
290 DISPLAY "WHEN FINISHED"
300 DISPLAY
320 INPUT "DEPOSIT AMOUNT? ":DAMT
330 IF DAMT=0 THEN 360
340 DTOTAL=DTOTAL+DAMT
350 GOTO 320
360 NBAL=BALANCE-CTOTAL+DTOTAL
370 DISPLAY :"NEW BALANCE= ";NBAL
380 INPUT "CHECKBOOK BALANCE: ":CBAL
390 DISPLAY "CORRECTION= ";NBAL-CBAL
400 END
500 CALL CHAR(96,"0")
510 CALL CHAR(97,"0000FF")
520 CALL CHAR(98,"86184333402CC61")
530 CALL COLOR(9,13,11)
540 CALL HCHAR(10,6,96,18)
550 CALL VCHAR(11,6,96,6)
560 CALL HCHAR(11,7,98,5)
570 CALL HCHAR(11,12,96,4)
580 CALL HCHAR(11,16,97,8)
590 CALL HCHAR(12,7,96,17)
600 CALL HCHAR(13,7,97,17)
610 CALL HCHAR(13,18,96,2)
620 CALL HCHAR(14,7,97,17)
630 CALL HCHAR(15,7,96,17)
640 CALL HCHAR(16,7,96,9)
650 CALL HCHAR(16,16,97,8)
660 RETURN
```

As the program is written in the manual, there may be a few problems. There is no DIMension statement, so if you have more than ten outstanding checks or deposits you will get an error. Because there is really no need to even worry about subscripts, delete **(N)** in Lines 200, 210, 220, and 230 and **(M)** in Lines 320, 330, and 340. You may then also delete Line 190 and change Line 240 to GOTO 200; and delete Line 310 and change Line 350 to GOTO 320.

Remember what I said above about spaces, and insert a colon before the first quote mark on Lines 130, 250, and 370 to make the screen easier to read. You may wish to add SOUND and red lines if the balance or correction is negative. Try your own ideas.

Name and Address File

Another easy solution—find Issue 2 of *99'er Magazine* and use the *Electronic Home Secretary* program. [Reprinted in this volume.—Ed.] What? You haven't keyed it in yet?? I thought *everyone* grabbed his issue of *99'er* and immediately keyed in *all* the programs!!

You can probably use this program as is, or adapt it to your needs to make your address file, phone list, Christmas list, or even a wedding invitation list. You can add a printer to print address labels if you want.

Recipe Conversions

How about recipes? Some people cook with a dab of this, a glug of that, enough flour until it looks right, and cook it until it's done. But a computer is more precise and will give you exact amounts. Try this program to convert a recipe.

```
100 REM  RECIPE CONVERSION
110 CALL CLEAR
120 DIM AMT(20),M$(20),ING$(20)
130 PRINT "ENTER NUMERICAL AMOUNT"
140 PRINT "(USE DECIMAL IF FRACTION)."
150 PRINT "THEN ENTER MEASURE (C,TSPP.
    )"
160 PRINT "THEN ENTER INGREDIENT."
170 PRINT :"ENTER '0' TO END RECIPE."
180 INPUT "AMOUNT: ":AMT(I)
190 IF AMT(I)<=0 THEN 250
200 INPUT "MEASURE: ":M$(I)
210 INPUT "INGREDIENT: ":ING$(I)
220 I=I+1
230 PRINT
240 GOTO 180
250 CALL CLEAR
260 FOR J=0 TO I-1
270 PRINT AMT(J);M$(J);" ";ING$(J)
280 NEXT J
290 PRINT :"MULTIPLY BY WHAT NUMBER"
300 INPUT "OR DECIMAL FRACTION? ":F
310 CALL CLEAR
320 PRINT F;"TIMES ORIGINAL RECIPE"::
330 FOR J=0 TO I-1
340 PRINT F*AMT(J);M$(J);" ";ING$(J)
350 NEXT J
360 PRINT :"CONVERT AGAIN? (Y/N)"
370 CALL KEY(0,K,S)
380 IF K=89 THEN 290
390 IF K<>78 THEN 370
400 END
```

Let's show an example of this program. Key it in then RUN. Remember you need to use decimal fractions.

AMOUNT: 2
MEASURE: CUPS
INGREDIENT: SHORTENING

AMOUNT: 2
MEASURE: CUPS
INGREDIENT: SHORTENING

AMOUNT: 2
MEASURE: CUPS
INGREDIENT: SUGAR

AMOUNT: 2
MEASURE: (just press ENTER)
INGREDIENT: EGGS

AMOUNT: 1.5
MEASURE: TSP
INGREDIENT: ALMOND EXTRACT

AMOUNT: 2
MEASURE: TSP
INGREDIENT: BAKING POWDER

AMOUNT: 4
MEASURE: CUPS
INGREDIENT: FLOUR

AMOUNT: 4
MEASURE: DOZ.
INGREDIENT: ALMONDS

AMOUNT: 0

If you want to triple the recipe, you would next enter 3. Answer CONVERT AGAIN? (Y/N) with Y, and this time try .5 and the recipe will be halved. While someone is keying in this program, another member of the family can try this recipe. It's Grandpa's Almond Cookies. Mix the ingredients together in order (except the almonds), roll in balls, and flatten slightly on cookiesheets. Press one almond on top of each cookie and brush with egg. Bake at 375 for about 10 minutes.

You may use this program as part of a larger program that retrieves the recipe from a file, then asks if you want to convert the recipe. You may want to READ the recipe from DATA statements rather than using INPUT. You can get fancy and print the title and instructions and draw pictures. [Also check out *Micro Bartender* in this book—a program that can be adapted for any recipe file.—Ed.]

Inventory

There are many ways to approach an inventory program. Ten programmers will come up with ten different programs. One possibility is to use the *Electronic Home Secretary* program. Here is one method for a household inventory. Use DATA statements and enter each item in the following order: room number, item, cost.

```
100 REM  HOUSEHOLD INVENTORY
110 FOR I=1 TO 4
120 READ R$(I),C(I)
130 NEXT I
140 CALL CLEAR
150 CALL SCREEN(4)
160 PRINT TAB(9);"INVENTORY"
170 PRINT :::"CHOOSE:"
180 PRINT :"1. LIVING ROOM"
190 PRINT :"2. KITCHEN"
200 PRINT :"3. STUDY"
210 PRINT :"4. WHOLE HOUSE"
220 CALL KEY(0,K,S)
230 IF K<49 THEN 220
240 IF K>52 THEN 220
250 CALL CLEAR
260 CH=K-48
270 CALL SCREEN(C(CH))
280 PRINT "INVENTORY OF ";R$(CH)::::
290 TOTAL=0
300 RESTORE 460
310 READ ROOM,ITEM$,COST
320 IF ROOM=9 THEN 380
330 IF CH=4 THEN 350
340 IF ROOM<>CH THEN 310
350 PRINT ITEM$,"$";COST
360 TOTAL=TOTAL+COST
370 GOTO 310
380 PRINT "TOTAL","$";TOTAL
390 PRINT "DIFFERENT ROOM?(Y/N)"
400 CALL KEY(0,K,S)
410 IF K=78 THEN 430
420 IF K=89 THEN 140 ELSE 400
430 END
440 DATA "LIVING ROOM",15,KITCHEN,8
450 DATA STUDY,10,"WHOLE HOUSE",12
460 DATA 3,COMPUTER,1000
470 DATA 3,DESK,129.50
480 DATA 2,"MICRO. OVEN",450
490 DATA 1,PIANO,5900
500 DATA 3,PRINTER,225.50
510 DATA 2,REFRIGERATOR,425
520 DATA 1,SOFA,425
530 DATA 2,STOVE,525
540 DATA 1,TELEVISION,475.50
550 DATA 3,TELEVISION,150
560 DATA 3,TYPWRITER,300
570 DATA 9,ZZZ,9
```

Only a few items in a few rooms are shown here to illustrate the logic of the program. You will probably want to include more rooms, the year purchased, and perhaps depreciation, replacement value, and a few other remarks. And don't forget to add titles to make the information more meaningful. You can use this program idea for any kind of inventory from food storage to retail products. Extended BASIC allows nice formating of output (with PRINT USING or IMAGE) so the numbers line up. It is also possible in regular BASIC by testing the length or the size of the numbers and printing accordingly.

I entered the DATA items in alphabetical order so they will be listed alphabetically, but you could use a sort routine to alphabetize the items or list the items according to cost. Following is a basic sorting routine:

```
100 REM  SORTING
110 REM  N=NUMBER OF ITEMS
120 DIM A(50)
130 CALL CLEAR
140 INPUT "N= ":N
150 IF N>1 THEN 180
160 PRINT :::"N MUST BE LARGER THAN ONE
    , TRY AGAIN PLEASE.":::
170 GOTO 140
180 IF N<51 THEN 210
190 PRINT :::"N MUST BE LESS THAN OR EQ
    UAL TO FIFTY (50). TRY AGAIN.":::
200 GOTO 140
210 PRINT
220 FOR I=1 TO N
230 INPUT "A"&STR$(I)&" = ":A(I)
240 NEXT I
250 PRINT :::
260 LIM=N-1
270 SW=0
280 FOR I=1 TO LIM
290 IF A(I)<=A(I+1)THEN 350
300 AA=A(I)
310 A(I)=A(I+1)
320 A(I+1)=AA
```

```
330 SW=1
340 LIM=I
350 NEXT I
360 IF SW=1 THEN 270
370 FOR I=1 TO N
380 PRINT A(I);
390 NEXT I
400 PRINT
410 END
```

You can use this interchange sort algorithm to arrange a list of numbers in *ascending* order. In this example, the user inputs the numbers of items in the list, N, and then enters each number (in any order). For this example, N is limited to 50. The maximum execution time for 50 numbers is about 50 seconds.

Within a FOR-NEXT loop, each number is compared to the next number. If the first number is larger than the second number, those two numbers in the array are switched and SW is set equal to 1 to indicate a switch is made. If the first number is smaller than or equal to the next number, the loop returns to the next pair of numbers to compare.

If SW = 1, at least one switch has been made and the process is repeated with SW reset to zero and the limit LIM of the loop set to the place a switch was made (the numbers after the last switch will be in ascending order with the largest number of the orginal list situated last in the series.)

To change this algorithm to rearrange a list of numbers in *descending* order, simply change the "less than" sign in statement 230 to "greater than." More efficient (and complex) sorts are available for large sets of numbers, but this algorithm is sufficient for smaller sets of numbers.

The alphabetizing algorithm is the same as this interchange sort algorithm with the list of variables changed to string variables. Just change all occurences of A to A$ and AA to AA$. In a regular program the INPUT and PRINT formats would be different from this example.

In the inventory application, we have three variables for each item: room, item name, and cost. These could be read in as arrays and the sort routine would need to interchange all three items. For example, let A(I) be the cost that you are sorting. You would need to add:

```
242 RR$=R$(I)
244 II$=ITEM$(I)
252 R$(I)=R$(I+1)
254 ITEM$(I)=ITEM$(I+1)
262 R$(I+1)=RR$
264 ITEM$(I+1)=II$
```

This coding ensures that the variables associated with each cost are interchanged in the same order as the costs are interchanged. You could also combine the room number and item name into one string variable to be interchanged with the cost variable.

Income Tax

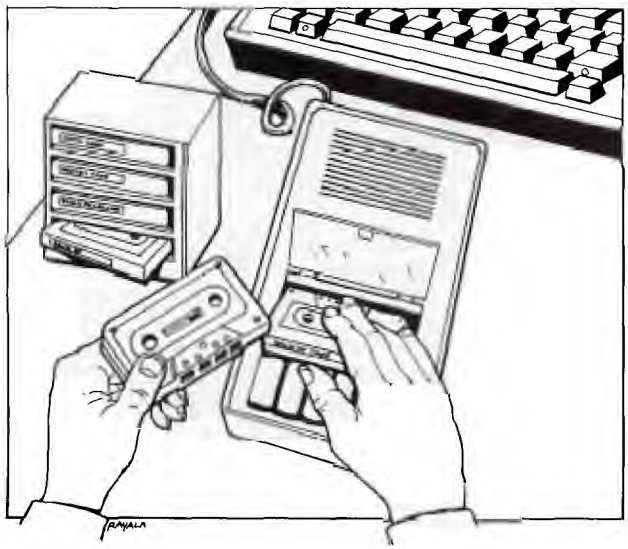
Probably the most common use for the computer when helping out at income tax time is keeping track of expenditures in different deduction categories. You can use the same program idea discussed above in the inventory section, but with a slightly different data structure. Instead of room number, you would use category (medical, interest, contributions, etc.). You would probably still use item and cost, and possibly add the date of expenditure. Your DATA statement would look like this:

```
500 DATA 1, "DR. PAYNE",25.50,"MAY 9"
```

for a medical expense of $25.50 to Dr. Payne on May 9.

I have suggested several ideas to help you get started writing your own programs for your own home, business, or professional applications. Now you just need to DO IT!

Recently someone asked me for a special income tax program— one that would indicate zero taxes to be paid. Hmmmm. I'm still thinking about that program . . . .



A Beginner's Guide

To Cassette Operation With A Home Computer

You bought your TI-99/4A Home Computer because the plug-in Command Cartridges looked like a quick and easy way to get started. You played the games and typed in the programs that you found in the *User's Reference Guide*. Now comes the moment of truth—*What to do next?* The answer, fellow 99'ers, is easy: Learn how to use a cassette tape recorder with your computer so that you can begin to build up a program library by recording and saving the many excellent software programs that are printed in *99'er Home Computer Magazine.*

TI manufactures a special cassette recorder (PHP-2700) for use with its computers that comes supplied with a dual cassette cable. If you cannot locate this recorder, things may get a little complicated. Finding a recorder that provides satisfactory results is not as easy as you'd think. To explain why, I will have to give you a quick background on how a computer talks to a tape recorder and vice versa.

What the Recorder Records

In order to do the wonderful things your computer is capable of doing, *bits* (the "offs" and "ons" that computers use) have to be arranged into patterns. This is true not only for numbers, but also for letters. For example, if you type the letter "A" on the keyboard, your TI-99/4A really sees a pattern that looks like this: off-on-off-off-off-off-off-on. If we think of an "off" as a zero and an "on" as a one, the pattern looks like this: 01000001. Remember that everything your TI-99/4A does is based on groups of *binary* numbers like that. A group of 8 bits is called a byte.

Learning to count in binary is beyond the scope of this article, but there are a number of books or articles around that can teach it to you. What you should know for now is that each letter and character has its own pattern of zeros and ones (its own binary value). For example, the 65th pattern (a *byte* value = 65) represents the letter "A" in the ASCII character coding system used by the TI-99/4A and most computers. This means that 65 is the ASCII value of letter "A." That is why the computer will give you back an answer of 65 if you ask for the value of ASC("A").

In order to read data from a tape recorder, your computer will have to be able to read in bytes of data. That means that it will have to understand "offs" and "ons" when listening to the tape. But, the TI Home Computer doesn't listen to the cassette tape for "off" and "on" sound. Rather it listens to two different frequency tones that represent the two states.

Not all Recorders Are Equal

It appears generally true that it takes more power for a cassette tape recorder to produce or reproduce a high frequency than it does to produce or reproduce a lower frequency tone. If the volume is not high enough during either recording or playback, your computer won't hear anything, or it might not be able to hear the higher frequency tone. In order to help the TI-99/4A hear the high frequency tones properly, the tone control knob on the recorder should be set at or near the maximum level. Even if this is done, some tape recorders cannot handle the high frequency. If your recorder doesn't have a tone control, there's a good chance it was meant to handle only the frequencies of human speech and won't be mechanically able to handle the high frequency tone at all.

Since it is possible that your recorder cannot reproduce the high frequency tones properly, your computer has to be sure that it has *read* all the data. How can it be sure that nothing was lost? Your computer counts the number of "ons" that it heard. After every so many bytes, it expects to read a number on the tape. This number tells the computer how many "ons" it should have read. If the two numbers don't match, a *parity error* has occurred and the computer will tell you that you have a problem.

Now suppose that the volume is set high enough to reproduce the high level tones, but is up *too* high? Well, too much volume causes distortion in a tape recorder. This distortion will mean that some of the tones will not be heard accurately by the computer at all. It's just as if someone screamed in your ear. You know something was said, but you don't know what it was.

A Remote Possibility

There is one additional problem that may crop up even with tape recorders that satisfy the above criteria: Almost all cassette recorders have a remote control jack which allows you to stop the recorder by pressing a button or switch located on the microphone. Unfortunately, since this jack is meant to work with the manufacturer's own microphone, there is no guarantee that the jack is hooked up the same way in each tape recorder. In fact, there is a 50-50 chance that a non-TI tape recorder model you may buy or already own will not be compatible with the Home Computer system. This means that the drive motor of your recorder might not be capable of being turned on and off *automatically* by the computer when the plug on the TI cable is inserted into the recorder's remote jack.

Luckily, if this is true for your recorder, Emerald Valley Publishing Co. sells an inexpensive adapter (called "TEXSETTE") which is used between your recorder and the TI cable. If you don't want to spend the money for this adapter, you can get by without it by *manually* starting and stopping the tape [except if you intend to use cassette data files, in which case the automatic operation is necessary.—Ed.].

The conclusion you can draw from all this is that your TI-99/4A requires a tape recorder with specific attributes in order to consistently guarantee good results. If you do not already own a recorder, I strongly suggest that you buy the model PHP2700 tape recorder from Texas Instruments. If you do have a recorder, you should try it out before incurring the expense of purchasing a new one.

Plugging In!

Now that we have discussed why some recorders won't work at all or won't work with the remote control jack plugged in, let's get down to business. Shut off your machines and plug the wide connector (with 9 holes in it) into the back of your computer. The other end of the cable has two cords. One cord has three plugs attached (labeled plug #1), and the other (plug #2) has only two. The tape recorder that you connect to plug #1 will be called "CS1" by the computer. If you are lucky enough to have a second usable tape recorder, you can hook up that one to plug #2. It will be called "CS2" by the computer. Just follow the installation instructions printed on the card that came with the TI cassette cable. If your tape recorder does not have a remote control jack, just ignore the instructions to insert the black plug. Note that CS2 does not have a playback plug. You can only record on CS2.

Plug the tape recorder into an electrical outlet and you are now ready to check out your system. [A battery-operated tape recorder is usually too unreliable for recording and playing back data for your computer because of the possible fluctuations in speed and amplifier gain over the life of the battery.—Ed.] Load a high-quality (remember we have to record those high tones accurately!) C-10, C-15, or C-30 blank tape into the tape recorder. The number part of the tape code gives the number of minutes of recording time available on both sides of the tape. A C-10 tape has 5 minutes of recording time on each side. You can use a tape as long as a C-60, but never anything longer. This is because longer tapes are thinner, stretch more, and may not maintain proper speed in the recorder. For this first test, make sure the tape is completely blank. Turn on your computer and select TI BASIC. Key in the following 4-line program:

```
100 PRINT "HELLO"
110 I=30
120 PRINT "MY VALUE IS";I
130 END
```

Turn up the volume on your TV (or monitor) by a few notches so that you can hear a slight hum. Set the volume control on your tape recorder midway between the lowest and highest settings. Set the tone control (if there is one) up to maximum. [Or, if you are using the TI PHP2700, follow its manual's setup instructions.—Ed.] Now type in SAVE CS1 and press the ENTER button. Follow the instructions that the computer gives you to rewind the tape and begin recording. When you press "record" on your tape unit and then press the ENTER button on the computer, the tape should start moving.

If the tape doesn't start moving, you have a non-compatible remote control jack. If this is the case, wait for the computer to leave recording mode and print the "VERIFY (Y/N)" message. When it does, type in an "N". Now remove the plug from the remote control jack and begin the recording process all over again (by typing SAVE CS1 and pressing the ENTER button). When you are told to record, you should now see the tape moving.

Getting Adjusted

After a short pause, you will actually hear your program being recorded onto the tape. The recording consists of an initial long phrase of a single tone, followed by bursts of sound with a very short pause between bursts. The initial tone is used to tell the computer on playback that data is coming. This tone is recorded before each program and each block of data (which we will talk about later). When the recording is over, you will get the verify message (see above). Type in a "Y" (you don't have to press the ENTER button). Follow the instructions about rewinding the tape. When you play back the tape, listen to the sounds that it is making. Note that the volume is much louder than when you recorded. If that initial tone does not sound pure (if it seems to warble, with the tone going higher and lower), you are probably using a recorder that won't work well consistently. If the tone does seem pure, you're halfway home!

When the tape goes silent, the program has finished loading. You should get a message that says either "DATA OK" or "ERROR IN DATA". If no message prints, then the volume setting was too low and your computer is still waiting for the first recognizable byte of data. It will eventually get tired of waiting and give you a "NO DATA FOUND" error. Just wait for this message to appear, or shut off your computer and start all over again.

If you got the "DATA OK" message, you are home free! Relax and go on to the next paragraph. If you were unlucky enough to get a "NO DATA FOUND" error, turn up the volume one notch. Write down the latest notch on a piece of paper. In either case, respond to the computer question by entering an "R" to re-record. The computer will guide you in another recording session. Keep repeating the process until you can't change the volume any further, or the "DATA OK" message appears or the error message has changed (i.e., from "NO DATA FOUND" to "ERROR IN DATA"). If you can't change the volume any further, your recorder just isn't good enough. Don't aggravate yourself any longer—go out and find somewhere to buy the TI recorder. If the "DATA OK" message has appeared, you are in good shape. If the message has changed, back off your last change by half a notch. For example, if moving the control from 6 to 7 made the "ERROR IN DATA" message appear, try the recording process again at 6½. If that doesn't work, try it at ¼ notch intervals. If *that* doesn't work, forget it. Buy a different recorder.

After you get the "DATA OK" message, mark the volume setting in some way. I usually dip a toothpick in white paint (a light nail polish will also work) and dab

a line on both the recorder and the control so that I can easily see that the volume setting is correct. You now have a functioning cassette tape system and are ready for bigger and better things.

Better Safe than Sorry

When you entered the SAVE CS1 command, you told the computer to copy the bytes that represented your program inside the computer onto a tape. The entire program is saved each time. Your program is still in the computer, however. If you agree to verify your tape, TI BASIC will read the data from the tape and compare it in a byte-for-byte manner with the program still residing in memory. Unless the two match perfectly, your 99/4A will issue a warning that you have a bad tape. ALWAYS VERIFY ANY SAVEs BEFORE ENDING A PROGRAMMING SESSION!

The tape version of the program is saved in a "machine image" format that is meaningful only to TI BASIC. You cannot, however, write a TI BASIC program that will read this tape. The only way to get your program back into the 99/4A is via the OLD CS1 command. This will load the program back into the machine. Anything that may have been in the computer before the OLD CS1 will be lost. By the way, you can SAVE CS2 (if you have a recorder hooked up to cable #2) and then read in the tape by entering OLD CS1. Of course, you have to move the tape over to the recorder attached to cable #1 first!

The instructions built into the TI-99/4A whenever you enter the SAVE CS1 or OLD CS1 command assume that you have only one program per side of tape. A long program will require about 3-4 minutes of recording time. This means that it is possible to save about 4-5 programs on each side of a C-30 tape. If your recorder has a tape counter, just keep track of where the next free space on the tape is located. Then, when the computer tells you to rewind the tape, just fast-forward to that next free spot on the tape instead. Make sure to keep a log of what programs are recorded on tape and where they are located. [If you don't want to be bothered by this, and want maximum reliability, it is better to use C-10 cassettes and record only *one* program per side.—Ed.]

A cassette tape recorder will usually have the ability to record a new program directly over an old one. It is good to get into the habit of completely erasing a tape, however, when you no longer need it. This ensures the best possible recording the next time you use the tape.

Filing Data

The cassette recorder also makes a handy data storage device for use in your computer programs. Suppose that you have written a program to keep track of the bowling scores and figure out the handicap of each member of your bowling league. You don't want to re-enter this information each time you run your program. What you need is a way of saving the data when you are through with it so that it can be read in the next time around. Some people do this by entering the information in DATA statements each time before SAVEing the program. A better way of doing this is to write out a small *file* of data onto tape. Your program can then read in this data file the next time it runs. TI BASIC has an easy way of doing this by using the INPUT # and PRINT # statements.

Before you can read or create a file, you must tell the computer a little about your file. This is done by the OPEN statement. Your reference manual does a pretty good job of explaining this statement, so I'll just go over the parts specifically dealing with cassette tape files.

Unlike the SAVE command which writes out your entire program as a large "chunk" of data, BASIC data files can only handle small chunks of data, called *records*, at a time. Each file can contain 1 or more records. All cassette records in a file must be of the same size. They can all be 64 bytes (characters) long, 128 bytes long, or they can all be 192 bytes long. You can specify other lengths as part of the OPEN statement, but TI BASIC will boost the number up to either 64, 128, or 192. If a record you want to write is shorter than the length that you specify, TI BASIC will add enough blanks at the end of the record to make it the right length.

Each record can contain as much data as you can fit in a record of that size. When you have a statement that uses PRINT # and ends with a semi-colon, BASIC will add that data to the record, but will *not* write anything out to the tape. When BASIC sees a statement with PRINT # that *doesn't* end with a semi-colon, it will write out everything in a record (including this last piece of data) to the tape. When the record is written to tape, it is preceded by the steady high-pitch tone that starts off a SAVE. That means that BASIC uses a lot of tape to write a single record. In fact, if you use records that are only 64 bytes long, it is possible that more room is spent on the tape for the start tone then is used to record the data! Remember that more room on the tape means slower reading by the computer. That's why I usually use 192 byte records and try to fit as much data as possible into each record. Doing this will cut down on the number of records written to tape, and make the program run faster.

Because TI BASIC only writes to tape when you tell it to, the computer must have total control of the cassette recorder so that it can start and stop the recorder as needed. This means that the black remote-control plug must be inserted (and functional!). If your remote jack is not compatible with the TI-99/4A, you will *not* be able to use the recorder for saving and reading data under program control.

You can store in two different formats. DISPLAY format means the data is saved just the way it would look in a DATA statement. INTERNAL format saves the data in the same way that the computer stores the information internally. Numbers require 8 characters (bytes). Strings (i.e., names) require 1 byte (for the length) plus the data itself. I usually save my data in INTERNAL format so that I know the length needed for numbers no matter how big or small they are.

THE BASICs of Record Keeping.

Let's write a part of a program that will save each bowler's name, his pin average and his handicap. Pretend that we have 60 bowlers in our league. If we restrict each bowler's name to a maximum of 45 characters, we will need a total of 62 bytes per bowler (45 bytes + 1 = 46 for the name + 8 for the average + 8 for the handicap = 62). We can therefore fit the data for 3 bowlers into one 192 byte record. (See Listing 1.) If you have not

filled up a record by the time the program hits the CLOSE statement, TI BASIC will fill out the record with blanks and write it out. You do not have to worry about writing out a last record that is partially full. Just remember always to program in a CLOSE statement. To read the data file into your program, you need program instructions that almost duplicate the write program (see Listing 2, below).

When your program executes the OPEN statements, the computer will issue commands about rewinding the tape and pressing ENTER. When INPUTing from tape, the screen will scroll up one line to indicate that it has begun processing the tape just before it reads the first record.

Once you have these basic components working and understood, you will probably wish to embellish them with things like update capability, printing of bowlers' statistics, etc.

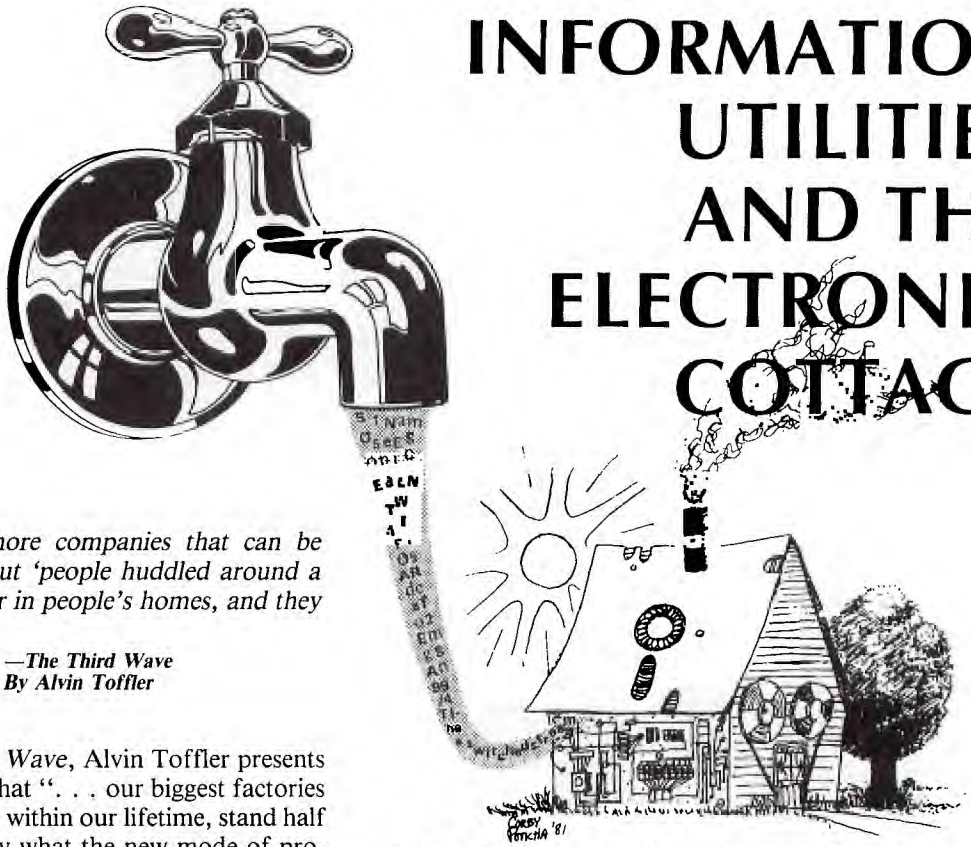
I have often been asked why TI provides the CS2 plug. I have to admit that most manufacturers do not provide dual cassette support. It is useful if you must process more data in your program than the computer can handle inside its memory. You would need *two* recorders hooked up, and would read in as much data as possible (for example, as file #1) on CS1, then do whatever you have to, and finally write the updated data out on CS2 (as a different file number). You would then go back and read in the next batch of data from CS1, update it, and write it out. You repeat this until there is no more data on CS1. This allows a *small* computer to handle very *large* files.

At this point you should have the basic knowledge for choosing a cassette recorder, and getting it to work with your computer. Keep in mind that tape storage transforms your Home Computer into a very powerful and versatile machine. And once you get familiar with the few simple procedures and precautions, each occasion of saving and loading programs and data files will become second nature. . .one might even say, "filled with memories . . . ."

```
100 REM  ********************
110 REM  *  LIST 1 CASSETTE OPER.  *
120 REM  ********************
130 REM   ROOM FOR 60 BOWLERS NAMES, A
    VERAGES, HANDICAPS
140 DIM B_NAME$(60),B_AVG(60),B_HANDI(
    60)
150 REM  ****** INITIALIZE NAME ARRAY
160 FOR I=1 TO 60
170 B_NAME$(I)="EMPTY
    "
180 NEXT I
190 REM  ******* INPUT BOWLER DATA LOO
    P
200 REM
210 FOR I=1 TO 60
220 CALL CLEAR
230 PRINT "INPUT DATA  FOR BOWLER #";
    I::::::
240 INPUT "NAME?  ":NAME$
250 REM  IF BLANK NAME, EXIT LOOP
260 IF NAME$="" THEN 390
270 REM  SET NAME LENGTH TO 45 CHARAC
    TERS
280 NAME$=NAME$&"
                                    "
290 B_NAME$(I)=SEG$(NAME$,1,45)
300 PRINT
310 INPUT "AVERAGE?  ":B_AVG(I)
320 PRINT
330 INPUT "HANDICAP?  ":B_HANDI(I)
340 PRINT
350 NEXT I
360 REM  ****** END OF INPUT FROM KEYBO
    ARD
370 REM  ********************
380 REM  ****** BUILD RECORDS AND WRITE
     TO TAPE
390 CALL CLEAR
400 PRINT "  <<< BUILDING RECORDS >>>":
    ::::::
410 REM  OPEN FILE FOR OUTPUT
420 OPEN #1:"CS1",OUTPUT,INTERNAL,SEQU
    ENTIAL,FIXED 192
430 FOR I=1 TO 60 STEP 3
440 REM  PRINT THREE SETS OF DATA PER
     RECORD
450 PRINT #1:B_NAME$(I);B_AVG(I);B_HAN
    DI(I);
460 PRINT #1:B_NAME$(I+1);B_AVG(I+1);B
    _HANDI(I+1);
470 PRINT #1:B_NAME$(I+2);B_AVG(I+2);B
    _HANDI(I+2)
480 NEXT I
490 CLOSE #1
500 REM  ****** END OF WRITE TO TAPE SE
    CTION
510 END
```

```
100 REM  ********************
110 REM  *  LIST 2 CASSETTE OPER.  *
120 REM  ********************
130 REM   ROOM FOR 60 BOWLERS NAMES, A
    VERAGES, HANDICAPS
140 DIM B_NAME$(60),B_AVG(60),B_HANDI(
    60)
150 REM  ****** INPUT BOWLER DATA F
    ROM TAPE
160 REM
170 CALL CLEAR
180 PRINT "  <<< READING RECORDS >>>"::
    ::::
190 REM  OPEN FILE FOR INPUT
200 OPEN #1:"CS1",INPUT ,INTERNAL,SEQU
    ENTIAL,FIXED 192
210 FOR I=1 TO 60 STEP 3
220 REM  READ THREE RECORDS FROM THE
     TAPE
230 INPUT #1:B_NAME$(I),B_AVG(I),B_HAN
    DI(I),
240 INPUT #1:B_NAME$(I+1),B_AVG(I+1),B
    _HANDI(I+1),
250 INPUT #1:B_NAME$(I+2),B_AVG(I+2),B
    _HANDI(I+2)
260 NEXT I
270 CLOSE #1
280 REM  ****** END OF READ FROM TAPE
    SECTION
290 REM  ********************
300 REM  ***** DISPLAY ON SCREEN SELEC
    TED BOWLER
310 REM
320 CALL CLEAR
330 PRINT "INPUT BOWLER NUMBER":::
340 INPUT A
350 IF (A<1)+(A>60)=-1 THEN 320
360 CALL CLEAR
370 PRINT "BOWLER'S NAME --"
380 PRINT B_NAME$(A)
390 PRINT :::"AVERAGE --";B_AVG(A)
400 PRINT :::"HANDICAP --";B_HANDI(A):
    :::
410 INPUT "ANOTHER? (Y/N) ":B$
420 IF B$="Y" THEN 320
430 END
```

INFORMATION UTILITIES AND THE ELECTRONIC COTTAGE



". . . we find more and more companies that can be described . . . as nothing but 'people huddled around a computer.' Put the computer in people's homes, and they no longer need to huddle."

—The Third Wave
By Alvin Toffler

In his book, *The Third Wave*, Alvin Toffler presents a powerful argument that ". . . our biggest factories and office towers may, within our lifetime, stand half empty . . . . This is precisely what the new mode of production makes possible: a return to *cottage industry* on a newer, higher, electronic basis, and with a new emphasis on the home as the center of society." Toffler goes on to single out many powerful socio-economic forces that are presently fueling this transition and points to the software production industry which has already set an early example as the fastest growing cottage industry of the 1980s.

Within the last three years, the microcomputer community has been witnessing the unfolding of an extraordinary event. I say "extraordinary" not because of what has already happened, but rather, for what it portends for the future. What is this event, and what great significance does it hold? Quite simply, the event has been the birth and maturation of "information utilities"—a significant event because of their awesome potential to speed up Toffler's timetable and change the way most of us live and work *within this current decade!*

There's certainly nothing mysterious about utilities. All of us are already familiar with telephone, electric, water and gas utilities. These are necessary and valuable resources delivered to and consumed in the home. If we now add *information* to this list, we create an "information utility"—a service that brings information to a place where the general public can access it and put it to use. . .and where the cost of packaging and delivery is *shared* by the consuming public. And what better, more convenient place is there for the general public to consume this information than in the home—the forthcoming "electronic cottage."

The New Timesharing

Timesharing, the foundation of all information utilities, is certainly not new. It was originally developed to serve the needs of business by providing companies with access to computer power without them having to buy expensive data processing equipment. Custom programming and technical assistance were available at extra cost to those who couldn't use the "canned programs."

What these information utilities have done is add a new wrinkle to the traditional timesharing concept. Using the famous "baking soda technique"—whereby a producer of this unglamorous age-old product continually dreams up and advertises new uses for it—they have repackaged timesharing to make it palatable to a much greater potential market. But lest you jump to the wrong conclusion, I should point out that these utilities are *not* simply pushing an old service to a new market. Rather, what we really have here is the creation of an *entirely new dimension* to timesharing—an attempt to satisfy a mass audience with extremely diverse needs and wants. . .and do it at an *affordable* price.

Information Services for the Masses.

To provide you with some appreciation for the great diversity of presently available information services, let's take a brief look at one of the largest, fastest growing utilities, *The Source* (a service mark of Source Telecomputing Corporation, a subsidiary of The Reader's Digest Association, Inc.) At present, The Source offers over 1,200 services in areas such as:

(1) computer-based message services
(2) proprietary databases
(3) business and professional applications packages
(4) personal and corporate services
(5) consumer purchasing
(6) entertaiment
(7) education
(8) "classical" timesharing

All these services enter a subscriber's home or business through existing telephone lines (using the *packet-switching*

networks of Telenet and Tymnet.) A local number is available in over 360 U.S. cities for accessing The Source. A subscriber types in (on a computer terminal connected to the telephone line, or a self-contained microcomputer with appropriate software to emulate a terminal) his or her private ID account number, and then chooses from a menu of services. Since subscribers can command the "host" computer in plain English (in a somewhat abbreviated form), very little instruction is necessary to do meaningful things—an extremely important attribute of any information utility.

Although an information utility such as The Source hopes, in the not-too-distant future, to be able to feed millions of inexpensive computer terminals in U.S. households, its present subscriber base is drawn from the business community and a small segment of the vast consumer community—the segment which presently owns microcomputers.

It's not surprising that businesses of all types are attracted to very inexpensive services such as electronic mail, travel arrangement, applications software packages, programming access to mainframes, and business/industry news. It does, however, take some stronger incentives to lure the consumer segment of the microcomputer community—the present-day pioneers who purchased their micros for home use. It's to this group that information utilities like The Source must ultimately cater if they hope to eventually reach the economy of distribution and substantial return-on-investment that are possible in a mass market.

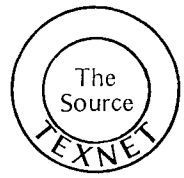
To this end, consumers with microcomputers are presently being wooed with a rapidly expanding array of personal services (such as bookkeeping, correspondence, travel arrangements and keeping track of investments), educational programs, home economics assistance, plus activities and information that the *whole* family can use—especially games, movie and product reviews, news and sports reports.

The TEXNET Turn-On

If having the services and activities of The Source in your home isn't exciting for you, how about having it together with the following package of special enhancements: color graphics and animation, music and sound effects, a software exchange with hundreds of free programs plus state-of-the-art synthetic speech—*actually "spoken" to you!* No, all this isn't just a "wouldn't-it-be-great-if" speculation of things to come, but rather embellishments to the basic Source menu.

The special services and enhancements I've been describing are available to users of the Texas Instruments TI-99/4A microcomputer, and come under the TEXNET (a service mark of Texas Instruments, Inc.) umbrella. Besides the microcomputer, the only additional items that are needed to take advantage of *all* of the special TEXNET features are an RS232 Interface and modem (for establishing a compatible telephone connection), a plug-in *Terminal Emulator II* Command Cartridge (the software for the micorcomputer), and the plug-in Solid State Speech Synthesizer—the Texas Instruments peripheral that "voices" the synthetic speech. The synthesizer won't be necessary if speech capability isn't desired.

Just how, exactly, are TEXNET and The Source related? According to Craig W. Vaughan (President, Software Sorcery, Inc.), a systems support consultant to Source Telecomputing Corporation and Texas Instruments, TEXNET *appears* to encompass The Source totally. That is to say, TEXNET subscribers have access to everything Source subscribers do *plus* additional special services that require the Texas Instruments Home Computer for access and use. Graphically, it would appear like this, with the outer ring of TEXNET including everything within The Source's inner ring, and expanding its own outer ring of special services over time. This is only an *appearance,* however, as Vaughan pointed out; "In reality, TEXNET users will be running a *shell program* . . .on The Source system."



Services on TEXNET fall into two major groups: (1) directory or lookup textual information, and (2) interactive or transfer services. In this first group there will be a product and technical newsletter (TI News), TI Software Directory, TI User Groups, TI Service Centers, and TI Phonetic Dictionary (helpful when programming with text-to-speech). The second group of services is really what TEXNET is all about. First, there are the *transfer* services. Sophisticated error-checking software in the *Terminal Emulator II* Command Cartridge will permit any of hundreds of user programs from the TI Software Exchange to be *downloaded* correctly into another user's system. Eventually, we can expect to see on TEXNET the capability for *direct* uploading and downloading between users. The TI Graphics Library and TI Music & Sound Library will work the same way: A TEXNET subscriber will be able to download the color graphics, musical scores, and sound effects into his own system for later use in his own programs.

The *interactive* services on TEXNET are really speech enhancements of services already available on The Source. For example, the electronic mail service—probably the most highly used service, and reason enough for many to be Source subscribers—is made even more intriguing when you mail is "read" to you by your machine's electronic voice. And if "electronic voice mail" intrigues you, wait till you experience TI Voice Chat: TEXNET users will be able to participate in "spoken"interactive communication, CB-style. Well almost . . . . What actually happens is that one user types in something, and the words get converted back into synthetic speech on the other end; the typed-in reply gets sent back, and then also gets converted to speech. So what we actually wind up with is a real-time verbal conversation *between two speech synthesizers!*

There's one short paragraph in the latest Source brochure that perfectly sums up what's presently happening in the world of information utilities:

> *"This brochure is obsolete. By the time you read this brochure, new information and communication services will have been added to The Source. Old data bases will have been updated, and streamlined "userfriendly" access procedures introduced."*

Without a doubt, it's an exciting time to be living and learning along the new information frontier.

For more information on TEXNET and the Source, see your TI dealer or contact The Source Telecomputing Corp., 1616 Anderson Rd. McLean, Virginia 22102.

DATA COMMUNICATIONS & the TI-99/4A

If you have invested in an RS232 interface and a modem in addition to your TI-99/4A system, you have the possibility of tapping a vast information network through existing and planned computer time-sharing services. A variety of information services such as news, financial information, computer games, various data bases, and program exchange, to name just a few, are provided through information utilities such as *The Source* (by Source Telecomputing Corporation). TEXNET, a collaboration between Source Telecomputing Corporation and Texas Instruments, will enhance data base services with the addition of text-to-speech, color graphics, and music. This service is available exclusively to users of the TI-99/4A. TEXNET and The Source are covered in another article. [See "Information Utilities & the Electronic Cottage."—Ed.] This article is an examination of basic data communications between the TI-99/4A and other computers.

Data Communications Concepts

A number of *coding schemes* have been devised to represent characters in order to input information into a computer. The most widely used code is the American Standard Code for Information Interchange—more commonly known as ASCII code. It is a 7-bit code which can represent 128 character configurations. Figure 1 illustrates the bit patterns associated with each of the characters. An eighth bit, called a *parity bit*, is commonly included in the ASCII code. The parity bit used to detect errors in the bit stream which might be due to the reading or transmission of the data. Parity of a ASCII coded signal can be odd or even. An ASCII code with even parity must contain an even number of ones; for an odd parity the number of ones must be odd (i.e., 1, 3, 5, 7). The Texas Instruments *Terminal Emulator II (TE-II)* Command Cartridge enables you to tailor your TI-99/4A to fit the characteristics of the remote computer system. With the communications device menu, you can specify the parity of the received or transmitted signal—odd, even or none (no parity bit)—and set the number of data bits at 7 or 8.

Figure 1 ASCII

MOST SIGNIFICANT OCTAL DIGIT	LEAST SIGNIFICANT OCTAL DIGIT 000	001	010	011	100	101	110	111
0 000	NUL	SOH	STX	ETX	EOT	WRU	RU	BEL
0 001	BS	HT	LF	VT	FF	CR	SO	SI
0 010	DLE	DC1	DC2	DC3	DC4	MAK	SYN	ETC
0 011	CAN	EM	SUB	ESC	FS	GS	RS	US
0 100	SP	!	"	#	$	%	&	'
0 101	(	)	*	+	,	−	.	/
0 100	0	1	2	3	4	5	6	7
0 111	8	9	.	;	<	=	>	?
1 000	@	A	B	C	D	E	F	G
1 001	H	I	J	K	L	M	N	O
1 010	P	Q	R	S	T	U	V	W
1 011	X	Y	Z	[	\	]	↑	←
1 100	_	a	b	c	d	e	f	g
1 101	h	i	j	k	l	m	n	o
1 110	p	q	r	s	t	u	v	w
1 111	x	y	z	{	|	}	~	DEL

NUL	Null or tape feed (control-shift P)
SOH	Start of heading (control A)
STX	Start of text (control B)
ETX	End of text (control C)
EOT	ansmission (control D)
WRU	(control E)
RU	ledge (control F)
BEL	...ll (control G)
BS	Backspace (control H)
HT	Horizontal tab (control I)
VT	Vertical tab (control J)
FF	Form Feed (control L)
CR	Carriage Return (control M)
SO	Shift out (control N)
SI	Shift in (control O)
DLE	Device link escape (control P)
DC1	Device control 1 (control Q)
DC2	Device control 2 (control R)
DC3	Device control 3 (control S)
DC4	Device control 4 (control T)
NAK	Negative acknowledge (control U)
SYN	Synchronous idle block (control V)
ETC	End of transmission block (control W)
CAN	Cancel (control X)
EM	End of medium (control Y)
SUB	Substitute (control Z)
ESC	Escape (control-shift K)
FS	File separator (control-shift L)
GS	Group separator (control-shift M)
RS	Record separator (control-shift N)
US	Unit separator (control-shift O)
SP	Space
DEL	Delete, rub out.

In data communications terminology, a *full duplex* channel implies that information can flow in two directions simultaneously. On a *half duplex* channel, the information can flow in both directions, but not simultaneously. If you select the half duplex mode from the *TE-II* communications device menu (and set the modem accordingly), the characters you send will be "echoed" back to your monitor or TV set, and appear on the screen. The echoed or extra character does not occur if full duplex is selected.

The public telephone network can provide means of communication from your TI-99/4A to another computer or information service. The information or bit stream that your computer sends and receives, travels serially through the network. That is to say that the bits making up a character are sent and received one after another.

There are a variety of modes of serial data transmission. Your modem transmits data *asynchronously*. This means that any set character is sent independently of any other character, and that the character bits are preceded by a start bit and followed by at least one stop bit. *Synchronous*

transmission requires that both the sending and receiving modems are synchronized by a clock signal. The rate at which data is transmitted (or received) is termed the *baud rate*. The formal definition of a baud is the reciprocal of the length of the shortest pulse used to create a character. Since all the bits of the ASCII code are equal in length, the terms "bits per second" and "baud" can be used interchangeably. A baud rate of 110 requires a minimum of 2 stop bits; at 300 baud and higher a minimum of 1 is required. The *TE-II* software allows you to choose between two baud rates (110 or 300), and your modem usually limits your use to either 110 or 300. The RS232 interface (without the *TE-II*) also allows you to use baud rates of 1200, 2400, 4800 or 9600. The higher rates can be used to output data to a printer or to send data to another TI-99/4A connected directly to your system.

The function of your modem is to convert the binary pulse train (1s and 0s) from your computer to some form of analog signal (tones) that can be transmitted over a telephone line. You will note that in the transmit mode your modem emits a continuous tone. This tone is called the carrier signal. When sending data from your TI-99/4A, the modem's function is to *modulate* (vary the amplitude or frequency) this carrier signal. It also works in the opposite direction by *demodulating* the carrier, so that the ASCII code sent to your TI-99/4A can be properly interpreted. Thus, the term "modem" is derived from the two words which describe its function: MOdulation and DEModulation. A common modulation technique is called frequency shift keying (FSK). This technique converts the binary pulses from the computer to two tones of different frequency. For example, if the carrier signal has a frequency of 1500 Hz, a 1 might be transmitted at 2000 Hz and a 0 at 1000 Hz.

Terminal Emulator II Command Cartridge

The *TE-II* Command Cartridge implements all 128 characters of the standard ASCII code, which is illustrated in Figure 1. It's also possible to send any standard ASCII *control characters* (used for signaling a remote computer or device to perform a predefined function), and display lines containing more than 40 characters by "wrapping" the extra characters onto a second line. The most powerful feature of the *TE-II* is the ability it gives users to store received data on tape or disk. You can review this data after logging off the remote computer, and can also send it to a printer or another computer.

Listing 1

```
100 REM  **DISPLAY FILE**
110 REM  READS IN A DISPLAY FILE AND O
    UTPUTS IT TO THE TV
120 INPUT "ENTER FILE NAME: ":FN$
130 OPEN #10:"DSK1."&FN$,FIXED 80,DISP
    LAY
140 REM  USE <LINPUT> BELOW IF EXTENDE
    D BASIC
150 INPUT #10:X$
160 PRINT X$
170 IF EOF(10)THEN 190
180 GOTO 150
190 CLOSE #10
200 GOTO 200
```

The format of the data stored by the *TE-II* is ASCII (display format) and is of fixed record length of 80 bytes (characters). In order to make further use of the information, it is necessary to write programs using BASIC. A simple example of such a program is shown in Listing 1. Line 130 opens a saved disk file using the OPEN statement. The following line inputs an ASCII character string; if the record denotes the end of file (EOF), the program ends. Otherwise, the program returns to the INPUT statement (line 140) and continues to read the data file until an EOF is detected.

Data Communications Using BASIC Programs

Display format files can be sent from your TI-99/4A to another computer under control of the BASIC listing shown in Listing 2. The program assigns file number 1 to the indicated disk filename, and file number 2 to port 1 of the RS232 interface. Each record or character string is input from the disk and then transmitted to the remote computer. Of course, this assumes that a means of recording this data is resident on the remote computer. This program can be used, for example, to efficiently transmit a pre-recorded message, or text file to another Home Computer.

The program listings in Listing 1 and 2 have a common flaw: If a display format file contains commas, the character string will be terminated by the first comma encountered. This is due to the fact that the BASIC INPUT statement interprets a comma as a separator between character strings or data items in display format data. (See page II-126 of the *User's Reference Guide.)* This flaw can be overcome by using the LINPUT statement from Extended BASIC.

Listing 2

```
100 REM  **TRANSMIT FILE**
110 REM  TRANSMIT FILE TO REMOTE COMPU
    TER
120 REM
130 CALL CLEAR
140 INPUT "ENTER FILE NAME: ":FN$
150 REM  THE FOLLOWING <OPEN> MUST BE
    MODIFIED TO MATCH THE STRUCTURE OF
     YOUR FILE
160 OPEN #1:"DSK1."&FN$,VARIABLE 192,D
    ISPLAY
170 REM  THE FOLLOWING <OPEN> MUST BE
    MODIFIED TO MATCH THE COMPUTER TAL
    KED TO.
180 OPEN #2:"RS232.DA=8.NU",VARIABLE,O
    UTPUT,DISPLAY
190 REM  USE THE <LINPUT> BELOW IF EXT
    ENDED BASIC
200 INPUT #1:X$
210 PRINT #2:X$
220 IF EOF(1)THEN 240
230 GOTO 200
240 CLOSE #1
250 CLOSE #2
260 END
```

TEXT-TO-SPEECH ON THE HOME COMPUTER

Go ahead—shout something at your Home Computer. . .but don't be surprised if it answers you back! Welcome to the exciting world of talking computers—a world in which synthetic speech will very soon cease being a novelty, and will instead become instrumental in the everyday interactions between humans and machines.

If you have a Texas Instruments Home Computer, you're one jump ahead of everyone else in taking advantage of this revolutionary communications tool. All you need additionally is the Speech Synthesizer peripheral, and the plug-in *Terminal Emulator II* Command Cartridge. Text that you type on the console keyboard will be converted to synthetic speech, and "spoken" through your TV set or monitor. There's no fixed vocabulary to constrain you, and personal phrases can be called up under program control through the TI BASIC computer language.

But this is only the beginning. . . . If you connect TI's RS-232 interface and a modem to this configuration, you can have access (through your telephone) to the electronic mail, database, entertainment, and computing facilities of the SOURCE and its offspring, TEXNET. The TEXNET service allows TI-99/4A users to access all the menu selections from its parent information utility plus some additional features *with the enhancements of text-to-speech, sound effects, music, and color graphics!* Imagine a weather report with a color graphic representation of a bright sun being blotted out by ominous looking rain clouds, while "Stormy Weather" is being played in the background, and the temperature, wind, humidity and other vital statistics are flashed on the screen and recited to you by your speech synthesizer—an exciting prospect at the least.

Linear Predictive Coding (LPC)

When Texas Instruments made the first single-chip speech synthesizer in 1978, its original application was in their *Speak & Spell* learning aid. The chip, A TMS5100, is essentially an electronic model of the human vocal tract (a mathematical model implemented as a filter network) that produces speech through a technique known as *Linear Predictive Coding* (LPC). There have been other approaches to speech storage-methods employing digitized speech and pulse-coded modulation—but these result in very high data rates (64,000-100,000 bits per second). And the higher the data rate, the fewer words of speech the available memory can hold.

The value of TI's LPC technique is its modest memory requirement: It provides speech quality nearly comparable to these other methods, at a much lower data rate (1,200 bits per second). For example, a speech reproduction of the words "Texas Instruments" requires approximately 90 times as many bits using digitized speech techniques as it requires with LPC.

What is the secret to LPC's economy of storage? The "P" in the middle that stands for "Predictive." Here's how it works: A speech waveform is originally sampled and encoded. This data is used to calculate the coefficients of the linear equations of the digital filter network that will control the "shape" of this synthetic vocal tract. When excitation noise (a chirp function and white noise generator) is applied to this filter network, the circuitry produces a simulation of the resonant effects of the mouth and nasal cavities.

Allophones

This is fine if a user doesn't mind being confined to a pre-stored vocabulary. A pre-stored vocabulary is, however, under-utilizing the synthesizer's capability to produce *any* spoken work *on demand* as long as it has the appropriate input data. This is where TI's recently unveiled *allophone stringing technique* comes into play. TI linguists have chosen 128 separate sounds called "allophones" that can be linked together to sound out any word in the English language. Allophones are variations of a particular "phoneme" (the smallest unit of speech that can distinguish one utterance from another) that are modified by the environment in which they occur. For example, the aspirated (followed by a puff of air) "p" in "pin" and the non-aspirated "p" in "spin" are allophones of the phoneme "p." These allophones represent the sound more accurately than the phoneme.

A total of 128 allophones are grouped in a library occupying only 3 kilobytes of memory storage. Each allophone is identified with a numerical code (indicating the parameters for setting the filter characteristics in the LPC synthesizer). When a word is entered into the computer, its ASCII representation is identified; the computer then searches through a set of rules (contained in 7 kilobytes of memory storage) to pick out the appropriate allophones and string them together in the proper sequence (*concatenation*) to represent the keyed-in words.

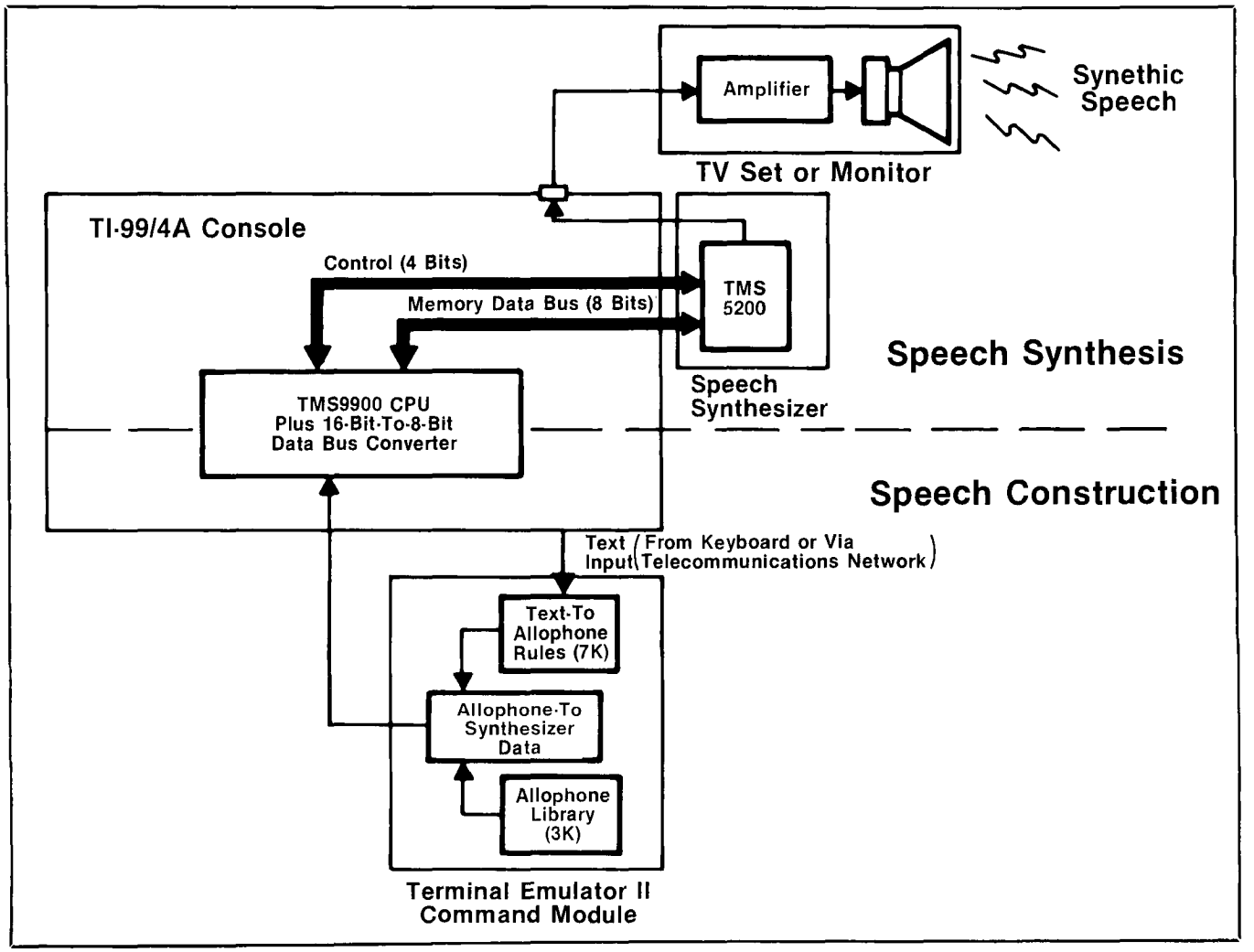
The rules (about 650 presently) overcome most of the many pronunciation exceptions and irregularities in the English language; they're able to select both phonemes and allophones correctly over 90% of the time. However, speech scientists have found it impossible to achieve 100% accuracy in a text-to-speech system of this type since there are too many silent letters and incongruities that humans perceive, but that the computer cannot discern. To get around this problem, some words must by typed into the computer phonetically or entered allophone-by-allophone.

If this were all the text-to-speech software did, the quality of the speech would sound monotonous and unnatural. TI therefore provides its software with the ability to produce more lifelike inflection: Users can add stresses to certain syllables, and required pitch patterns to particular points in a sentence. Questions then sound like questions, and commands like commands!

Text-to-Speech with the TI-99/4A

The chip used in the speech synthesizer peripheral that attaches to the TI-99/4A is a TMS5200—a second generation of the TMS5100 (used in the Speak & Spell). It has the following added features : (1) "Speak External" input which allows the chip to accept speech data from a source other than a Speech ROM (read-only memory), (2) an internal buffer to store chunks of data (freeing the computer for other tasks), and (3) a memory data bus allowing it to work with any standard 8-bit microprocessor. (The 16-bit data bus of the TMS9900 micropressor is converted to 8 bits for use with all TI-99/4A peripherals.)

The text-to-speech production by this configuration is a two part process : (1) the *speech construction phase* in which letters are translated into a digital representation of component sounds and are concatenated (strung together), and (2) the *speech synthesis* phase in which the LPC circuitry "voices" the spoken words through a simulation of the mouth and nasal cavities. As seen in the accompanying diagram, speech construction is handled by the software resident in the *Terminal Emulator II* Command Cartridge, and speech synthesis by the TMS5200 chip within the separate speech peripheral.



Just the sound of the name Walt Disney conjures up images of all those fantastic animated movie classics spanning over a quarter century of entertainment for young and old alike. But recently, the celluloid magic of Disney Studios has taken on a new dimension with the release of their eagerly awaited science-fantasy, *TRON*—an incredible computer graphics extravaganza in which fantastic vistas of texture and light are generated artificially by computer. As movie-goers worldwide continue to be awed by *TRON's* video warriors and computer programs fighting for survival in an electric universe, a new awareness of computers—and in particular, the mind-boggling possibilities of computer-generated imagery—permeates the consumer cosmos. With one wave of Disney's digital wand, the glass of Cinderella's slipper has been magically transformed into the cathode ray tube of a video monitor.

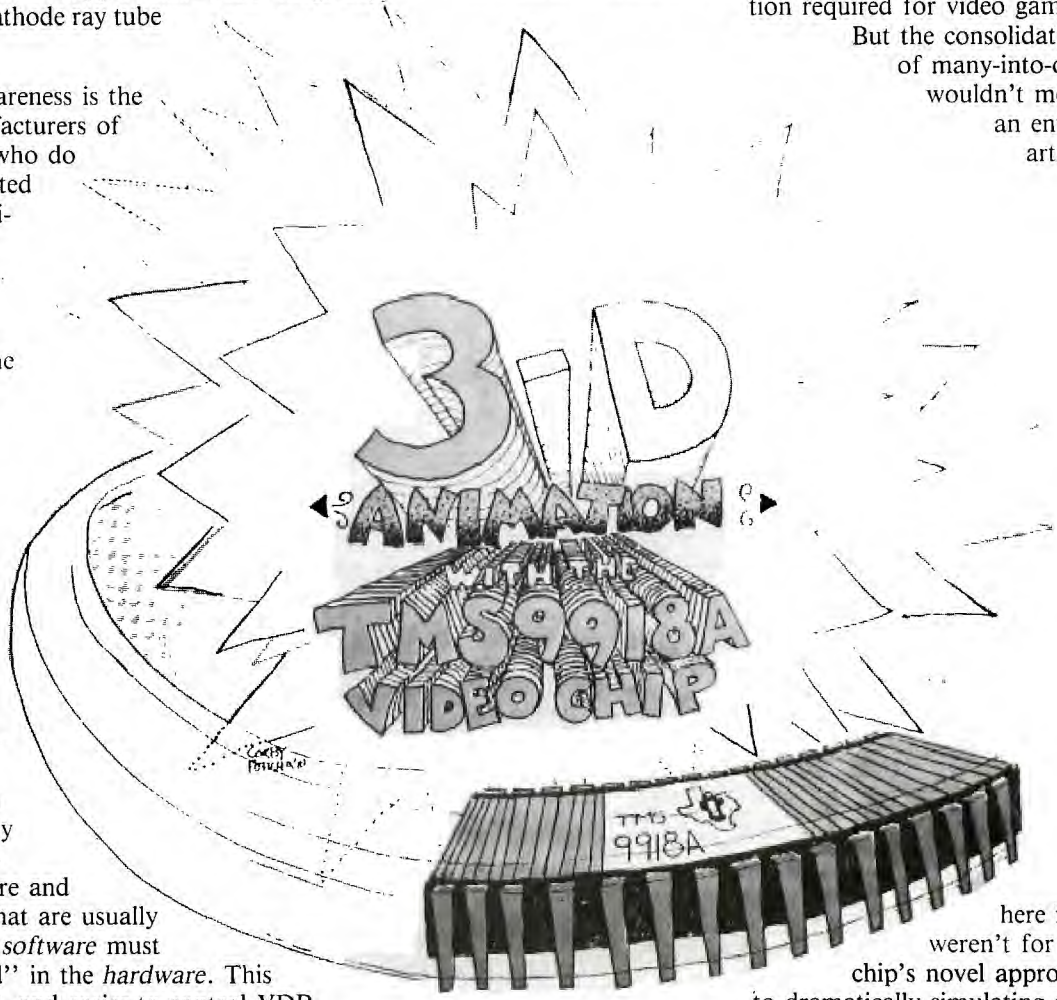
This heightened awareness is the death knell for manufacturers of consumer computers who do not provide sophisticated color graphics and animation capabilities. Fueled by *TRON* (and the horde of video clones that are destined to follow), the public's demand for, and expectation of, more visually spectacular video games and educational displays will surely take quantum leaps. How can computer manufacturers and software houses ever hope to satisfy this demand? That's one tough technical question that some of the finest design teams in the world are currently tackling. One thing is obvious, though—more and more special effects that are usually implemented through *software* must instead be "integrated" in the *hardware*. This means more powerful, and easier-to-control VDP (video display processor) chips—the silicon workhorses responsible for the displays.

The easier-to-control requirement doesn't necessarily mean easier for highly-trained, professional *programmers* to control. There will have to be a way for people such as as artists and "graphic gurus" with fantastic imaginations to interact directly with the display system—a way that requires only a bare minimum of "programming" experience to implement sophisticated visual effects.

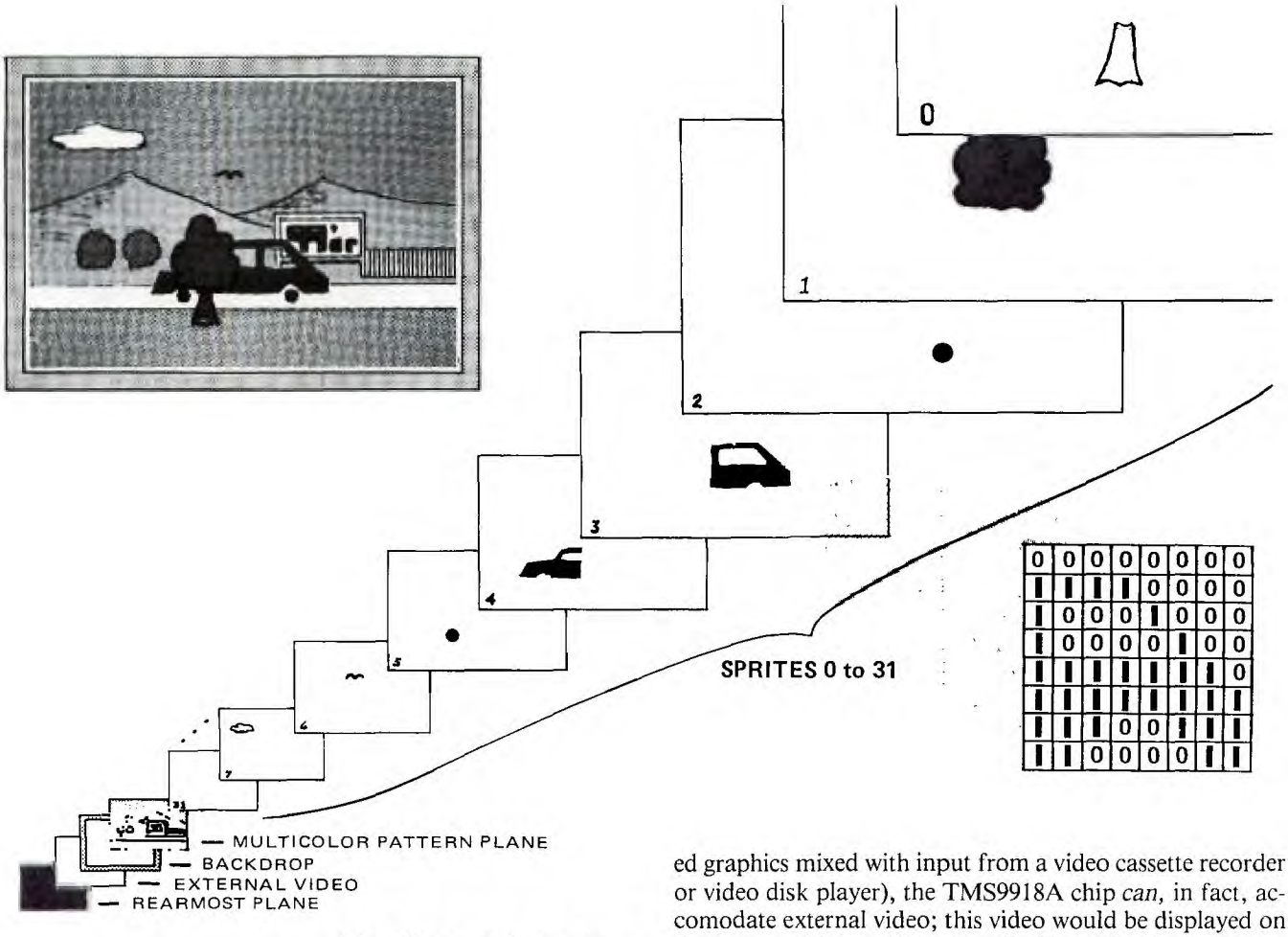
To anyone familiar with the interactive graphics capability of the Texas Instruments 99/4A Home Computer, it is obvious that TI has already made great strides toward this design goal—great enough, in fact, to cause at least two other well-known computer manufacturers to attempt to emulate TI with their "newly-discovered," smoothly moving graphic patterns now known universally as *sprites*. Color sprites as implemented on the TI-99/4A, however, have yet to be equaled in their versatility and ease of use in a multi-language environment. (Extended BASIC, TI LOGO, UCSD Pascal, 9900 Assembly Language and TI PILOT).

A Flat, Yet 3-D Sandwich

The wonder VDP chip behind sprites and other video affects that the 99/4A is capable of producing is called the TMS9918A. This complex LSI (large-scale integrated) chip represents the next generation beyond the many small- and medium-scale integrated circuits that formerly had to be assembled to achieve a display with a minimal level resolution required for video games. But the consolidation of many-into-one wouldn't merit an entire article here if it weren't for the chip's novel approach to dramatically simulating a 3-dimensional animated graphic display: It does this by creating nearly three-dozen flat, "stacked" geometric planes that are sandwiched one on top of the other onto the picture tube of your TV or color monitor.



On *each* of the first 32 planes (numbered 0 to 31), we can define the image of *one* sprite, give it one of the 15 standard colors (the 16th is transparent), and then set it in motion quickly and smoothly. We do *not* have to redefine the imagery over the screen to simulate motion, because once set in motion, a sprite can continue to move without further program control. When a sprite on a *lower* numbered plane (closer in the foreground) comes into contact with another sprite on a *higher* numbered plane, it progressively



blots the second one out and creates the illusion of passing in front of it.

For example, in the figure shown here, the moving car that is composed of four sprites set in motion together on plane numbers 2-5 will pass *behind* the stationary tree (composed of 2 sprites on plane numbers 0 and 1) and *in front of* the billboard which is drawn on the plane immediately behind the rear-most (number 31) sprite plane. By the same design rules, the cloud (plane 7) will mask the color of the sky behind it, and a bird (plane 6) both mask the sky behind, and appear to fly *in front of* the cloud. And since sprites move in a transparent surrounding, the scenery in the background behind the car may be seen *through* the "windows" of the moving vehicle! The entire scene has the appearance of depth and simulates a 3-D animated color movie.

The Multicolor or Pattern Plane is used for textual and fixed-graphics images. It is this plane (containing the sky, mountains, bushes, billboard, fence, roadway and grass) that the sprites on the remaining 32 planes appear to pass directly in front of.

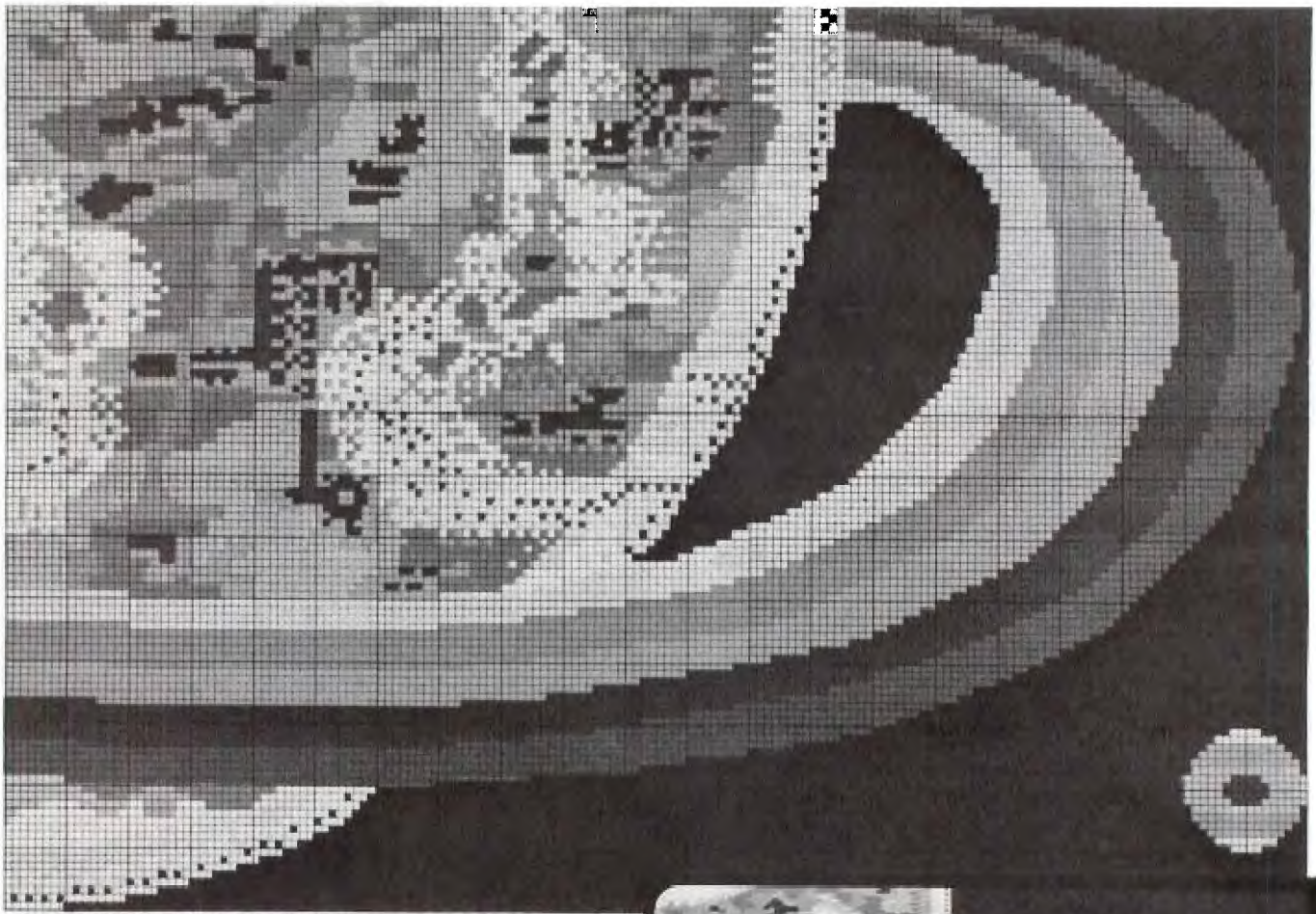
Immediately behind the Multicolor Plane is the Backdrop Plane—solid-colored and slightly larger than the other 33 planes in front of it, so that it forms a rectangular rim around the other elements on the display.

The rearmost plane is pure black, so that when the other planes are set to transparent, the screen appears to be black. Although there is no provision in the current version of the TI-99/4A for *simultaneous* on-screen mixing of external video with computer-generated graphics (e.g., sprites or fixed graphics mixed with input from a video cassette recorder or video disk player), the TMS9918A chip *can,* in fact, accomodate external video; this video would be displayed on the rearmost plane with part or all of it masked by computer-generated graphics until needed (e.g., as subtitles for the deaf or foreign language translation or perhaps a "real-life" video-taped space movie scene viewed through a scanner screen of a computer-generated starship command center). Add to this the capability of chaining together *multiple* 9918A chips, and you have the potential for a visual gaming or educational environment (in future versions of the TI Home Computer) that is simply mind-boggling!

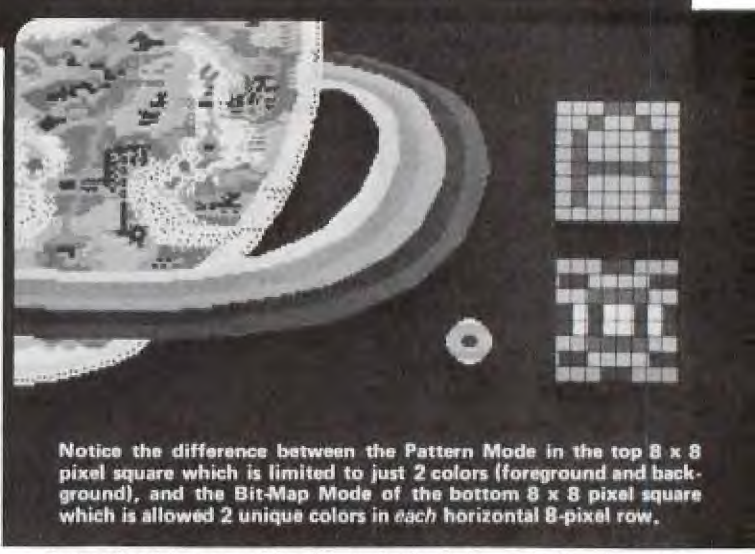
Those Magical Sprites

When sprites are on the screen, the 9918A chip organizes the display into a high resolution pattern of 256 by 192 little boxes or picture elements called "pixels"—the smallest controllable elements on the display. Each one of these 49,152 pixels represents a possible address for a sprite to reside at, or pass through when moving across the screen.

The shape of a regular or standard sprite is defined by an 8 × 8 bit pattern stored in memory. Each of these 64 bits correspond to one of the 49,152 screen pixels mentioned previously—with each being a single color whenever the bit pattern contains a 1 (is thereby "turned on"); a zero designates transparency ("turned off"). We can specify a larger sprite by either (a) using a 16 × 16 bit pattern ("a double-sized unmagnified sprite"), (b) magnifying the existing sprite by a factor of four, ("single-sized magnified"), or (c) using both techniques together to create a sprite sixteen times normal size ("double-sized and magnified"). This size feature allows screen objects to grow and shrink at will—with virtually *none* of the programming effort that would be required in more conventional VDP systems.

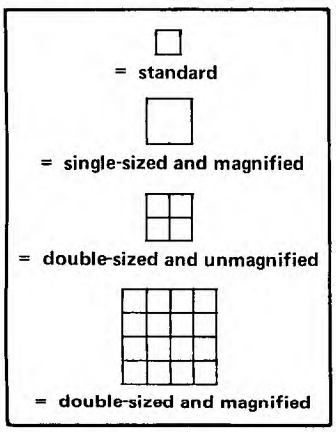


This high-resolution scene is shown over-sized to illustrate the color combination possibilities in Bit-Map Mode. Notice the color blending and shadings that the eye perceives when viewing the *same* scene on the screen-size display of the lower figure.

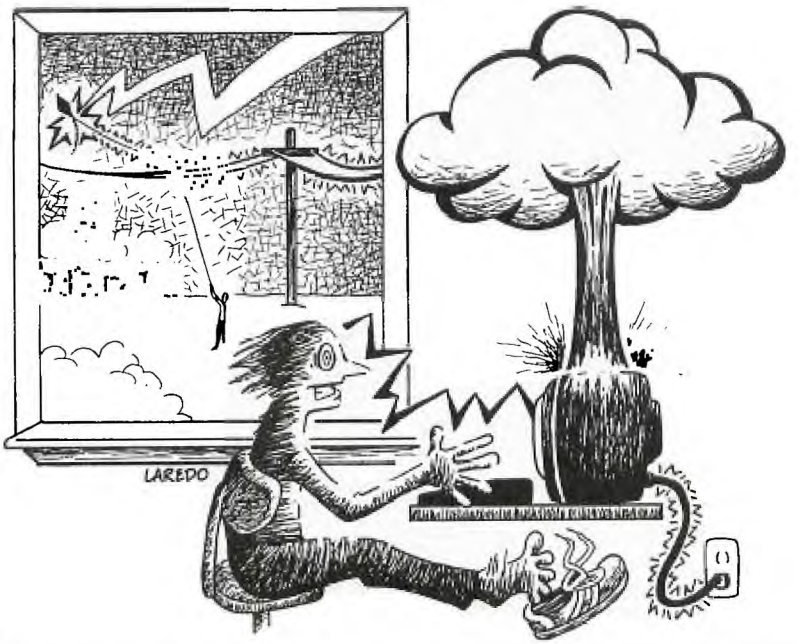


Notice the difference between the Pattern Mode in the top 8 x 8 pixel square which is limited to just 2 colors (foreground and background), and the Bit-Map Mode of the bottom 8 x 8 pixel square which is allowed 2 unique colors in *each* horizontal 8-pixel row.

Each sprite carries four attributes: The first two specify its horizontal and vertical position; the third defines its shape "name" (according to the bit-pattern concept described above); and the fourth specifies its color. Moving a sprite is simply a matter of changing its position indicators; it will continue moving smoothly on its own. The high-speed, smooth motion of a sprite compared with a conventional moving-graphic element is due to the smaller, more precise "steps" (higher resolution) that the sprite can take while moving. Animated secondary motion—for example, rotating wheels or an asteroid tumbling through space—is achieved by defining ("naming") several similar looking sprites in different secondary positions (e.g., states of rotation). Then swapping the sprite names causes what appears to be a *single* sprite to move smoothly across the screen.



The TMS9918A VDP chip has four modes of operation: (1) Graphics 1 or Pattern Mode, (2) Graphics 2 or Bit-Map Mode, (3) Text Mode, and (4) Multicolor Mode. The Pattern Mode consists of a 32-column by 24-row grid of 8 × 8 pixels in each 2-color grid square. Bit-Map Mode (shown in the above figures) allow *each* of the 8 horizontal rows within an 8 × 8 grid square to have 2 *unique colors.* In Text Mode the screen is a 2-color *single* plane (so sprites aren't available) of 40 columns by 24 rows composed of 6 × 8 grid squares. This allows an ASCII character set with each character formed from a 5 × 7 pixel grid, with 2 pixels between characters and rows. Multicolor Mode [see the "Super Crayon" article in this book for more information] divides the pattern plane into an unrestricted 64-column by 48-row color-square display, with *each* 4 × 4 pixel square allowed to take on any of the 15 colors or be made transparent.



POWER LINE PROBLEMS IN PERSONAL COMPUTERS

Although glitches, crashes, errors, false printouts, memory loss, and other forms of erratic microcomputer operation are usually blamed on software and hardware, most of these annoying problems actually come to you courtesy of your ordinary 120-volt powerline! These problems are directly traceable to three general causes: (1) processor-memory-peripheral interaction, (2) power line noise/hash, and (3) transient voltage surges. Fortunately, serious computer users don't have to live with these problems, because many types of corrective devices are available.

Powerline Coupling

The fact that microcomputer systems are so easy to hook up—just plug the computer and peripherals into the wall socket, and connnect the components with a few convenient male/female preassembled cables—makes them susceptible to power line noise. Connecting them to powerline strips that are integrated with RFI (radio-frequency interference) filters will effectively isolate the computer and peripherals from each other and from the power line—thus providing a convenient solution to the problem.

Hash

Hash is another problem altogether. When your favorite space-war game gets fouled up by "glitches," or your previously-proven-to-be-faultless program "blows up" or creates erroneous printout, externally created hash is the probable cause. Elimination of hash at the source is the most desirable solution. But with hundreds of potential sources (arcing in tools, motors, appliances, and other small electrical devices, plus loose, defective, or corroded light sockets, wall sockets, line-cord plugs, or wire connections), pinpointing the offender is often most difficult. That's where hash filters are most effective. They often can completely eliminate the interference.

An alternate approach to the hash problem is first to make certain that all equipment covers and shields supplied by the manufacturer are securely fastened in place. If that doesn't work, you might try building and installing your own shield. Also don't forget to make sure that you have an adequate grounding system with *direct ties* to a good ground rather than *ground loops* (which often provide a home for system hum that can induce glitches).

Transient Voltage Surges.

Transient voltage surges (transients) are certainly not friends of microcomputer circuitry. Semiconductor components are easily damaged by these momentary spikes—often 5 or 10 times the normal AC line voltage. And industry studies indicate that some transients have pulses up to 5,600 volts!

Common causes of destructive powerline transients include (1) demand power load switching by utility companies, (2) nearby lightning strikes, (3) static discharge, and (4) on/off switching of inductive motors, power supplies, air conditioning and refrigeration units. Any of these can cause a Differential Mode powerline surge—one in which short surges of extremely high voltage are developed *between* the AC lines. Anything connected to the AC lines will get a dose of this damaging voltage. The resulting "domino effect" could wipe out large sections of microcomputer memory.

A Common Mode surge occurs when *both* AC lines are brought to a very high voltage—a situation usually caused only by lightning. This high voltage may cause arcing between conductors and ground, destroying the insulation of power transformers (rendering the units worthless) and cables. Damage to switches and controls is also a frequent occurrence in this situation.

Besides the surge damages that are immediate and permanent, there are some harder-to-detect damages as well: deteriorated performance and shortened life-spans. These damages can be the most irritating since equipment will require repeated servicing and will often seem to be falling apart.

Fortunately, a large measure of surge protection is possible with clamping devices that can be placed across the AC line and between each AC line and ground. These devices are frequently built into special AC line cords, and thus, like the other protective devices mentioned, can be attached without altering any equipment.

Murphy's Law and the Home Computer

"ANYTHING THAT CAN GO WRONG WILL GO WRONG"
— Murphy's Law

"And at the worst possible time."
—Pincus's Corollary to Murphy's Law

The significance of Murphy's Law is etched into the hearts and minds of all professional programmers. As an outgrowth of their crucial need to prevent costly errors and protect their sanity, these professionals have developed a set of rules for minimizing the expected problems inherent in any programming project. In this article, I'll acquaint you with a few of these "tricks of the trade" that *you* can adopt when working with your own home computer.

Rule #1—Name that Variable

Most beginning programmers select simple two-letter variable names like X1, X2, X3, etc. The major problem with this kind of naming scheme is that you tend to forget what each one represents. Fortunately, both of TI's BASIC languages allow you to use long variable names. Unfortunately, using long names tends to slow down a BASIC language program. The solution is to pick a 4- or 5-letter name that adequately identifies the variable it represents. For example a program that needs a loop counter could have the name "LOOP" for the counter. [When picking names for variables, you should also be careful to avoid reserved words used in BASIC. If in doubt, check your *User's Reference Guide*.—Ed.] Keep a running list or chart of variable names and their meanings in front of you while you program. Avoid the temptation to add a variable to your program without updating the chart.

Rule #2—Save that Program

Get used to SAVEing your program after typing in about 50 program lines if your storage medium is a floppy disk, or 75 lines if cassette tape. Keep your labels up-to-date or else you may get confused in case of trouble. I always place the version number "V.XX" on the cassette before recording on it. Flipping the tape over and scratching out the old version number as soon as the program has been SAVEd makes sure that I don't accidently record over my last SAVE. For disk users, I suggest that they SAVE their programs under the names V01 and V02 *alternately* so that they don't fill up a disk.

Rule #3—Walk, Don't RUN

Computer-induced heartburn is one experience you'll definitely want to avoid. Yet how many of you court this malady by meticulously entering the last 50-75 lines of your program, then typing in RUN? If you're guilty of this practice, don't be too surprised if Murphy pays an unexpected visit and causes your computer to "freeze" on you—effectively wiping out everything in memory. The best way to avoid this is to always "back up" your *entire* program (on tape or disk) *before* you RUN it the first time. Then verify your recording. The next step is to *plan* what you want to test: Don't expect to have the program execute successfully the first, second, or even third time that you try it out. Instead, take a blank sheet of paper and write down what you want to test and how you will do it. Then RUN your program following your plan. If you notice a problem that does not halt your program, write it down on your test sheet and keep going. Don't stop to figure out what program line caused the problem; there will be time for that later. And above all, avoid the temptation to correct the error right away. Doing this will cause the following:

- If you change a program line, TI BASIC resets all the variable data (that you entered during your test) to zero or spaces. Subsequently, you will have to retype all that data back into your program.

- You may lose your train of thought when you stop your test in the middle. It's possible to miss testing a *major* item because you spent your time fixing a *minor* problem.

After you have done as much of your test as possible, stop the program and begin fixing all the "bugs." Don't forget to mark down the statements that you change. This will come in handy if you must restore a previous version because of a problem in making corrections. There are various way to correct (debug) a program, some of which you'll encounter as you read through this book. One of the best, however, is Rule #4.

Rule #4—Test, Test, TEST !

Any professional programmer will tell you that it's extremely difficult to produce a bug-free program—maybe even virtually *impossible* to do with very complex programs. So even after extensive testing, *your* program may have minor flaws in it. Therefore, the only way to produce a relatively clean program is to test it as much as possible. This brings up another corollary to Murphy's law: "The only programs without bugs are the ones not yet written." This says it all, and should be reason enough to TEST, TEST, TEST !