THE CREATIVE TRS-80

Edited by Ken Mazur

The Creative TRS-80

The Creative TRS-80

Edited by
Ken Mazur

Creative Computing Press
Morris Plains, New Jersey

To Maureen, who passed too quickly.

Library of Congress Number 82-74425
ISBN 0-916688-36-4

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Creative Computing Press
39 East Hanover
Morris Plains, New Jersey 07950

Contents

Chapter V. Programming Tips

Chapter VI. Games

Chapter VII. Graphics and Music

Chapter VIII. Word Processing

Chapter IX. Education

Chapter X. TRS-80 Strings

Preface

If you've ever read an issue of *Creative Computing* magazine, you know that our commitment to the growth and development of the TRS-80 has been a serious one. Every month, *Creative Computing* presents the TRS-80 owner with valuable information on the latest developments involving their machine as well as interesting tutorials on how to use the TRS-80 to its fullest potential.

But perhaps you've never picked up an issue of *Creative Computing* or maybe you've tried to get a hold of a back issue which contained an article of particular interest and found that the issue you wanted was one of the many issues of *Creative* that are now out of print. The *Creative TRS-80* was compiled for the *Creative Computing* newcomer as well as for the *Creative Computing* reader who needs a single reference source of valuable TRS-80 information.

While containing many hardware and software reviews, *The Creative TRS-80* also covers applications in business, the home, and the school. There are articles on programming tips to help you in many varied application areas as well as an entire chapter of game listings which you can type directly into your computer.

We think you'll find *The Creative TRS-80* to be a necessary addition to your TRS-80 library but remember — for the absolute latest in TRS-80 information, pick up an issue of *Creative Computing* every month!

John Anderson
Associate Editor
Creative Computing

Ken Mazur, has been a writer in the computer field for the past four years. He is a college instructor, a consultant to many computer publishers, and is also presently publishing his own computer magazine.

Special thanks to Jim Klaproth for technical assistance.

Chapter I
Hardware

Many Capabilities, Few Weaknesses

Why I Like the TRS-80

Stephen B. Gray

It's become fashionable in many personal-computer circles to call Radio Shack's machine the "Trash-80," to speak of Model I's hardware as poorly designed, and to cite various inadequacies of Level II Basic.

Yet, despite all these hardware and software problems, Radio Shack has somehow managed to sell over 200,000 (perhaps 250,000 by the time you read this) of these "poorly designed and inadequate" computers.

How did Radio Shack manage to fool so many people? How were a quarter of a million people hoodwinked into buying such an inferior piece of merchandise?

What They Wanted

One answer is that Radio Shack provided what many people out there wanted, at a price they felt was right, and at thousands of outlets all over the country where a person could go try it out before buying.

Once the TRS-80 caught on, the name became as magic in its own field as IBM's in the mainframe business. IBM may not make the best computers, or the fastest ones, but it knows, better than all the rest, the importance of service and support.

Radio Shack made a lot of mistakes with the TRS-80, as would any company marketing the first popular ready-to-run personal computers. But they've learned a lot, they've made tens of thousands of free fixes, and they've brought out three more TRS-80 computers that alone may well outsell both Apple and PET.

My TRS-80

Nobody who's used a personal computer for more than a few weeks is completely satisfied with it. There are always some features on other machines he'd like to have on his.

I've had a 16K Level II Model I since December 1977. I've been writing the TRS-80 column in this magazine since the Nov-Dec 1978 issue.

It took me months before I realized that some of my dissatisfaction with my TRS-80 was due to my not completely understanding how it works, and exactly how to program some difficult tasks.

Once I began to realize what my TRS-80 could do, and could not do, I began to appreciate it much more.

There's a great deal I still don't know about the TRS-80. I'm not all that much into machine language, preferring to use Basic, (which I'm still learning about,) in areas such as strings, matrices, and TRS-80 graphics.

But the more I use the Level II computer, the more I like it. I know just about what it can do, and can't, and I recommend it to most of the people who ask me what personal computer to buy.

Most, but not all. The TRS-80 can't please everybody, which is why the Apple II, PET, Atari and Sorcerer computers sell as well as they do.

What I Like About the TRS-80

Service and support are two of the main reasons for my liking the Level II TRS-80.

As for service, when I had problems with my RAM memory, and also wanted the lower-case modification installed, along with the free cassette-loading fix, all I had to do was take my keyboard unit down to a Computer Center in lower Manhattan, where a skilled technician took all of 55 minutes to fix the memory problem (a faulty RAM IC) and install the two mods.

Who else has over 225 Computer Centers around the country? How many other personal-computer manufacturers require that you *mail* the computer to them for service?

As for support, I'm talking about the vast amount of absolutely fascinating Level II software available.

I don't mean Radio Shack's programs, most of which have shown a great deal of conservativism and lack of imagination. (Although they're beginning

Stephen B. Gray is a frequent contributor to *Creative Computing* magazine.

to break away from the mold, and have brought out some good programs lately, mostly written by outsiders, and including Scripsit, Astrology and Dancing Demon.)

Although a great many poor programs are being sold by people whose main interest seems to be in making a fast dollar, some very clever software is being written by programming geniuses. Leo Christopherson, who wrote Dancing Demon, has written several outstanding games. Lance Micklus is another master gamesman.

The pages of *Creative Computing* and other computer magazines are full of ads for some highly imaginative TRS-80 games and some very well thought out utility and business programs for the TRS-80. There are programs for fighting your way through a dungeon full of demons, playing music, drilling children in math, balancing a checkbook, communicating on a network, performing advanced math, writing paychecks, word processing, playing baseball, simulating lab experiments, playing the horses, turning on household appliances, creating and using a database, printing a mailing list, controlling inventory, working in double-precision math, managing a budget, tracking stock trends, generating a horoscope, drawing animated movies, playing chess and backgammon, and hundreds more.

Yes, the other popular personal computers have a lot of programs, but nowhere near the variety and number written for the Level II TRS-80.

More publications specialize in the TRS-80 than in all the others put together: *80-US, The Eighty, 80 Microcomputing, PROG/80, S-80 Bulletin, Insiders,* and probably a couple more I don't know about. That's in addition to the magazines that regularly run TRS-80 articles.

What I Don't Like About the TRS-80

There are some things I don't like about the TRS-80, although several of these have been taken care of with free modifications.

I got terribly annoyed when extra letters started showing up on my screen, as in NEXXT, FFOR and RNND. That can be fixed by prying up the keys and cleaning the spring contacts; the newer keyboards don't use spring contacts.

The lack of lower-case letters was a nuisance until I had the lower-case mod installed. There were problems loading some tapes, until I had the free cassette-loading mod installed, which enabled me to load all but the very worst tapes.

The Level II TRS-80 Model I doesn't have color. But now there's the TRS-80 Color Computer. Several things were left out of Level II Basic. But they are in Microsoft's Level III Basic.

The Level II manual is really a reference manual, and as such is missing a great deal of helpful information. But Radio Shack promises to publish its own Level II user's manual some day. And several fine Level II manuals have been written outside Radio Shack.

Using cassettes for storing programs used to require a lot of cable-plugging and unplugging. But then I discovered a switchbox (Dick Fuller's RF-II) that eliminates all the cable-handling, also provides a speaker for listening to the bit-stream, and permits easy copying of tapes from one cassette recorder to another.

The TRS-80 Model I has no software-definable keys like the Exidy Sorcerer. But the TSHORT program from Web Associates provides that capability, in addition to several others.

A good letter-quality printer costs about $2,000. Well, that's really a problem, and my only solution is to save up for one.

Conclusions

After three years of using a TRS-80, I've learned its many capabilities and few weaknesses, and have learned to live with them. Occasionally there are some problems, such as when the Scripsit word-processing program doesn't work the way I want it to, but that's mostly because I don't use it enough to be fluent in all its idiosyncracies.

I wouldn't trade my Level II TRS-80 for any other personal computer made, except for Radio Shack's Model III, with integral disk drives and keyboard.

If there's a peripheral or program I want that doesn't exist, and it's not too far out, somebody will be selling it before long. □

Expanding the TRS-80 Model I

Exatron MM+

Harley Dyk

If you are considering memory expansion, floppy interface, serial I/O, etc. for your TRS-80 read on.

If you own a 16K Level II TRS-80 Model I, you own a very cost effective computer. This does not necessarily mean, however, that you are content with your computer.

Harley Dyk, 1644 Grant, Grand Haven, MI 49417.

If you are a programmer you are aware that programs often grow to fill (or exceed) available memory. If you are a serious user of your system you probably long to add a disk drive or alternative, such as the Stringy Floppy or the Beta-80. In either case you may need more memory and/or a floppy controller.

The MM+ (memory plus interface) by Exatron and the System Expansion by LNW Research provide quality alternatives to the Radio Shack Expansion Interface and either could save you some money depending on your needs.

MM+

The MM+ has just been introduced by Exatron (the Stringy Floppy company). The unit comes assembled and is made to fit under the TRS-80 monitor. Standard features are: 32K of memory, built-in power supply, parallel printer port (Radio Shack/Centronics compatible), serial printer port (RS-232C), light pen port, real-time clock, and general parallel port (IBM Model 50 compatible). The unit was designed with Stringy Floppy owners in mind, and this accounts for the fact that a floppy controller was not included as a standard feature.

The MM+ has room for an additional board and its power supplies run at or under 50% capacity. An additional 32K (bank 2) and floppy controller will be the first options available on the second board. Exatron is polling its Stringy Floppy owners to find what other options they would like to have available on the second board. The company plans to work on the options in order of preference indicated by their customers. Some of the other things under consideration are: color graphics, hard disk controller, RS-232C serial I/O, IBM Model 50 bidirectional interface (use typewriter keyboard), multi-port parallel I/O, A/D and D/A interface, TRIAC/SSR/OPTO-Isolater control interface, port FF audio output circuit (for sound effects), IEEE-488 Interface, and a communications modem.

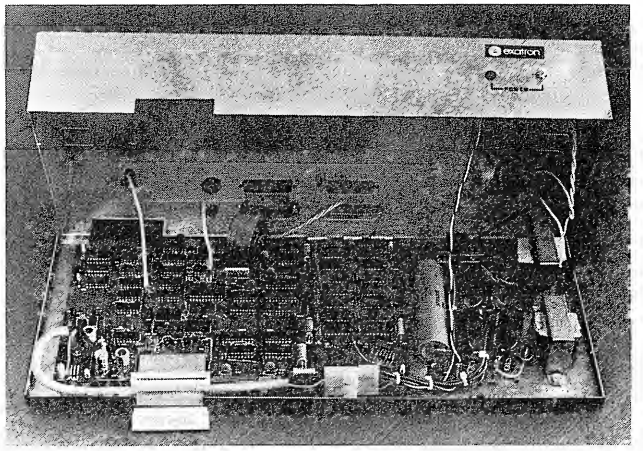
A unique feature of the MM+ is the light pen port. This port is designed for use with the "Photopoint" light pen made by MicroMatrix. The light pen can be used with a cassette recorder serving as an amplifier, but the light pen port makes the amplifier more convenient and leaves the recorder free. The port should work with any light pen that normally connects to the Radio Shack cassette recorder.

The MM+ is guaranteed to run at double CPU speed (3.55 MHz). This makes the MM+ compatible with the TRS-80 speed-up kit offered by Exatron.

Comparisons

The key to selecting one of the two expansions lies in the answers to the following important questions. Do you want to build your expansion unit? Do you need a floppy interface immediately? Do you need serial I/O now? (The MM+ has output only at this time.) Do the additional features being considered for the MM+ interest you? These questions address the basic differences between the two units. Table 1 can also help you compare the major features at a glance.

Both units have performed well for many users. Both work well with disk alternatives such as the String Floppy and Beta-80. Both units are of top quality and are produced by reputable companies. At a minimum, either unit should fix your OM errors and provide many additional features. □



The Memory + Interface (MM+) by Exatron.

	MM+	LNW
Size	17" x 7" x 3"	10" x 12" x 3" (in LMB 10123 chassis)
State	Assembled only	Bare board only
Cost	$199.95	$270-$300 including power supply and 32K
Memory	32K only	16K or 32K
Floppy Interface	No (but an option soon)	Add $50 plus cable
Real-Time Clock	Yes (can use with Level III Basic)	Add $4
Serial I/O	Printer output only (300 and 600 baud	Add $22
Parallel printer port	Yes	Add $3
Dual cassette port	No	Add $10
Light pen port	Yes	No
Bus extender	Yes	Yes
On-board power supply	Yes	Yes (minus transformer)
Dealers	No, mail-order order through program chairman (active Stringy Floppy owner)	None established, however some dealers may stock this board, otherwise mail-order
Warranty	Year/30-day money-back	90 days (board only)
Toll free number	Yes	No
Contact	Exatron 181 Commercial St. Sunnyvale, CA 94086 800-538-8559	LNW Research 8 Hollowglen St. Irvine, CA 92714
Misc.	Guaranteed to run at 3.55 MHz, has memory bank select circuit so can add another 32K, has on-board memory-mapped address decoding.	Prices of options above are accurate only if built in the order listed. Any other order could change prices since parts are shared in many sections.

Table 1.

Another Expansion Alternative

The LNW System Expansion

Richard Zatarga

This article addresses those computerists who are ambitious, industrious, and capable of reading a schematic diagram; possess a better than average ability to use a soldering iron; and have a desire to upgrade a TRS-80 Model I computer and save over $100 in the process.

The above mouthful may sound like science fiction, however, if I have just described you and you are willing to spend a few—well actually, quite a few—evenings with iron and solder in hand, you can have an Expansion Interface for two-thirds of Radio Shack's price, and with a serial RS-232C/20mA interface thrown in as a bonus.

"Sounds too easy!" "What's the catch?" you ask. Well read on and I'll tell you how I did it. First, I parted with $69.95 plus $3.00 for shipping and handling for the LNW Research System Expansion printed circuit board. Please note that this is a bare P.C. board. What you are paying over $70 for is a meticulously traced and silk screened epoxy circuit board and LNW's electronic expertise.

Ten days after I placed my order for the P.C. board, UPS delivered the board and the assembly/user manual. After opening the box, I inspected the board for damage. The board was fine, but what I noticed during the inspection was the very tight and dense component layout. I've built a few electronic kits in my day from a simple speaker system to a complex color television, but I had never run across a you-build-it circuit as tightly packed as the System Expansion. This project is definitely not--I repeat, *not*—for the novice solder jockey or the sweat solder expert who works with copper tubing and a propane torch. Construction of this unit requires time, patience and precision.

With the board inspected, I sat down in my favorite easy chair and began to read the manual. Quickly thumbing through its 67 typewritten pages, I was initially impressed. However, after reading it thoroughly from cover to cover, I found the manual to be a bit of a disappointment due to the lack of detail, especially in sections on assembly, testing and trouble-shooting.

The next thing I did was to collect all of my electronic component catalogs and a few back issues of some computer magazines. Armed with the component checklists provided in the LNW assembly manual, I perused the catalogs and magazine advertisements looking for the best buys on the various components I needed to construct the System Expansion.

I found that resistors were a bargain from one supplier, while integrated circuits were better purchased from another. Another vendor had great IC prices, but his capacitor prices were outrageous. The results were separate orders to six vendors in four states. With my orders for parts in the mail, I sat back and impatiently waited for the components to arrive.

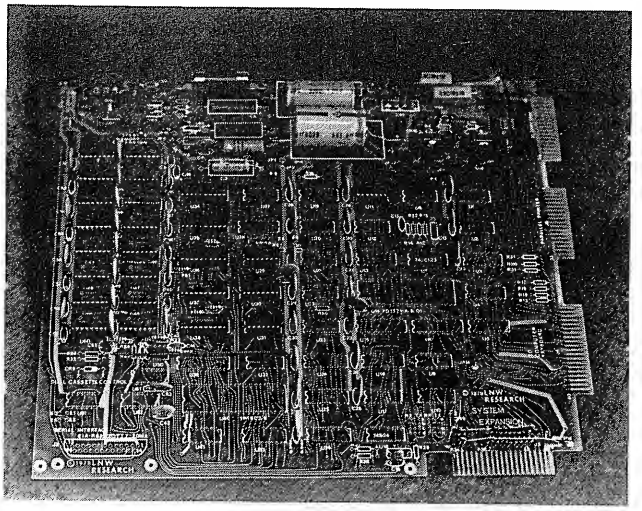
A stroke of luck—the first order to arrive consisted of some integrated circuits and all of the IC sockets I needed for the interface. Actual construction began with mounting and soldering all of the sockets on the PC board. Next, the resistors and capacitors were added. Finally, all the diodes were inserted and soldered in place. I worked on the interface a few evenings a week over a period of two months.

It was just three days shy of the second month when my final parts order arrived. If all the components had been readily available, eight or nine evenings would have been all the time needed to complete the board and thoroughly check my handiwork.

Testing

The main power to the System Expansion is provided by a TRS-80 computer transformer. The LNW onboard power supply takes the raw transformer voltages and provides the regulated +5V, -5V, +12V, and -12V needed to activate the rest of the board. These voltages are isolated from the main part of the System Expansion by five jumpers, and the LNW assembly manual has a procedure for testing them out before the jumpers are added and power is supplied to the rest of the board.

When I first powered up the System Expansion, I expected something to happen, such as blowing both onboard fuses or at least a little puff of smoke. Nothing! I proceeded with LNW's test procedure taking voltage readings at designated test points with a DMM. Everything in the power supply section checked out fine.



The LNW Research System Expansion with power supply and 32K. Transformer not shown.

Richard Zatarga, 800 Towner Swamp Rd., Guilford, CT 06437.

Next, I added the jumpers providing power to the rest of the board. Please note, all of the IC sockets were empty at this time. I saw no reason to test a fully loaded board and take the chance of incinerating some expensive integrated circuits.

I proceeded with LNW's next test procedure. All voltage supplies checked out except for one of the +5 volt sections. I traced the +5 volt supply through the schematic and onto the board, and found a couple of terminating resistors bridged together with solder and loading down the +5 volts to less than 3.2 volts. A light touch with the tip of a soldering iron rectified the problem and all voltages checked out.

Verifying the power supply voltages is the extent of the testing procedures provided in LNW's manual. Still being cautious, I decided to test the rest of the System Expansion one section at a time. The first section I tested was the Dual Cassette Control.

Using the parts list by section, I inserted the required IC chips into their proper sockets, and attached two cassette recorders to the DIN connectors. I powered up the System Expansion and the keyboard, and loaded blank tape into each recorder. I wrote a short Basic program and entered CSAVE#-1'TEST'. The first recorder responded. I then entered CSAVE#-2'TEST' and the second recorder fired up. To complete the test of the Dual Cassette Control, I CLOADed the test program from each recorder and ran the program. Both recorders saved and loaded data perfectly. So far, so good.

The next section I tested was the 32K memory. I tested this section in 16K increments. Why annihilate 16 RAM chips at once when I could do it in two easy steps. The first eight chips were inserted and power was applied to the system. I entered ?MEM from the keyboard and lo and behold a number greater than 15,572 magically appeared on the screen. I ran a RAM test routine and all the memory checked out. I was feeling pretty good at this point and inserted the other eight RAM chips.

PRINT MEM yielded 48,340 this time. The RAM test confirmed that all, including the new 32K memory addition, was functioning properly. Now my ego was really soaring. It must have been up to eleven points on a ten point scale. Confidence in my construction ability was at an all time high, so I decided to forge ahead, even though it was 1:30 in the morning.

Next on the list for testing was the parallel printer port. The relevant chips were inserted and a printer cable connected between the System Expansion and a borrowed printer. I powered up the entire system and CLOADed the test program mentioned earlier. I entered LLIST and *Eureka* the program listing was output to the printer. I modified the program by changing all the PRINT statements to LPRINT. RUN ENTER produced a nicely formatted report on the printer. Three sections tested and I was batting a thousand. I decided to check one more and call it a night.

I inserted the integrated circuits required for the Floppy Disk Controller section. The 40-pin FD-1771 disk controller chip took some effort to get into its socket. There always seemed to be one or two pins that slipped out of alignment. Finally, the FD-1771 was properly inserted, and I connected a borrowed disk drive to the interface and applied power to the system—again.

I inserted a diskette into the drive and pushed the reset button. Nothing happened! What was wrong? I checked the power switches. Everything was on. I checked the floppy cable and that looked fine.

I read the DOS manual (When all else fails, read the instructions. Right!!) and discovered that DRIVE 0 must be the terminal drive, i.e., the last drive on the cable, and it must be the drive farthest away from the interface. Also, the connector nearest the interface must always be attached to a drive. My borrowed disk drive and cable came from a friend with a two drive system and he only lent me one drive. I moved the drive to the first connector on the cable, and this time when I pushed the reset button the drive activated, the CRT screen went blank for a second, and *voila*! DOS READY appeared on the screen. I ran the TEST2 utility provided on the TRSDOS diskette to stress test the floppy controller. The test was successful and I decided to pack it in for the night. I'd test the RS-232C/20mA interface in the morning.

The last of the ICs was put on the board. The 40-pin UART went into its socket without a hitch. It's amazing what a little experience or a couple hours of sleep and four cups of coffee will do for one's manual dexterity. I entered a serial interface routine LNW provided in the appendix of their manual. The RS-232C/20mA interface worked like a charm. Testing of the System Expansion was complete. All sections worked and I had an expansion interface equal to Radio Shack's with the added plus of an RS-232C/20mA serial interface.

The Bottom Line

Did I really save money by going the construction route to upgrade my Model I? My answer has to be a definite yes. Was the completed unit worth the time, effort and, on occasion, aggravation required to construct it? Again, I must answer in the affirmative. Permit me to elaborate.

My total cost for the printed circuit board, sockets, resistors, capacitors, power pack, miscellaneous hardware, integrated circuits, including sixteen 4116 memory chips, was $310. I built a case for the completed board and two power packs—one for the System Expansion and one for the CPU—from some scrap lumber I had in my workshop. If you don't have access to any scrap lumber, another $10 or so can be added to the overall cost.

A substantial investment indeed, but still quite a bit under Radio Shack's price.

Check the discount mail order advertisements in this magazine for the cost of a Radio Shack Expansion Interface. The cheapest one I found was $249. Check out the prices on 4116 memory chips. The best value I found was $40 for eight chips. That totals to $349—only $29 more then I invested and no construction required. But hold on for just one second, the System Expansion includes an RS-232C/20mA I/O section and my total cost includes the components required for this serial interface.

Check the advertisements again, and you'll find that $89 is about the best buy you can find for Radio Shack's RS-232C option. Now your cost is up to $418. A $100 savings should be worth the time and effort required for anyone to build the unit. It was for me.

An added advantage of constructing the LNW System Expansion is the ability to repair any problems that may develop with the unit. Armed with the schematic diagram, the sectionalized parts list and the circuit descriptions provided by LNW, a minimum of time and effort should be all that is needed to locate and fix most troubles. Please note that this last statement assumes some electronic and trouble-shooting ability.

Conclusion

I've been using my System Expansion for the past four months. I have my own printer and disk drive attached to the unit. You can borrow hardware from friends for only so long before they start forcing lease with option to buy contracts on you. Well, the System Expansion has been performing very well. I haven't experienced any crashes or erratic operation. Disk I/O has been impeccable. Everything has been functioning perfectly.

So, if you possess the skills I mentioned earlier, want to save some of your hard-earned money and want the satisfaction of building a sophisticated piece of computer equipment, then I recommend the LNW System Expansion. You won't be disappointed. □

Aimed at the Business Market

The TRS-80 Model II

Stephen B. Gray

Radio Shack, manufacturer of the best-selling TRS-80 personal computer, introduced its big brother, the new TRS-80 Model II, at a New York City press conference in late May.

The new computer is said to perform as a general-purpose dataprocessing machine, an intelligent terminal, or a word processor. Software is available for general ledger, accounts receivable, inventory control, mailing list management and payroll.

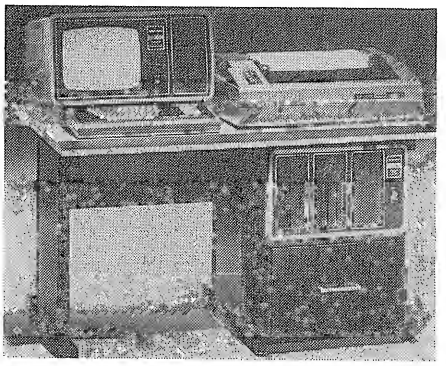
The TRS-80 Model II microcomputer system, designed and manufactured by Radio Shack in Forth Worth, is not intended to replace or obsolete what is now called Model I, but, according to John V. Roach, Radio Shack executive vice-president, is "specifically designed to take up where the original TRS-80 leaves off—a machine with increased capacity and speed in every respect, targeted directly at the small-business application market. A market which has been estimated from $1 billion to $2 billion by 1983. This machine bridges the gap between personal computers and the low end of the commercial market such as the IBM 5100 series, Dapapoint, ADDS, etc. Yet we do not plan to emulate these companies' sales techniques but rather plan to market this equipment in much the same way as the TRS-80 to those customers who are willing to come to us for sales, training, and maintenance."

Software-compatible (only Basic Programs) with the Model I, and resembling the smaller computer in several respects, TRS-80 Model II has the same 12-inch video monitor, but with 24 lines of 80 characters, with both upper and lowercase, as compared to 16 lines of 64 characters, uppercase only, in the Model I.

The TRS-80 Model II has a built-in 8-inch diskette drive at the right of the screen, providing half a million bytes of storage in addition to the 32k or 64k bytes of internal RAM. The drive is manufactured by Shugart, and second-sourced by two disk manufacturers whose names Radio Shack executives would not divulge.

Additional diskette drives are available in an expansion chassis that holds up to three drives, and which mounts in or on a system desk.

The new keyboard includes Control, Escape, Caps, Hold, Repeat and two software-programmable Special Function keys not found on the Model I keyboard.



Prices for the TRS-80 Model II start at $3499 for a 64K byte computer with it's built-in diskette drive. The disk expansion with one drive is $1150, with two drives $1750, with three drives $2350. The Model II system desk is $350.

The Model II uses the same Z80 processor as the original TRS-80, but operates it at twice the speed. A direct memory access feature, according to Dr. John D. Patterson, director of Tandy Systems Design, "further enhances throughput by controlling the data transfer between memory and disk, allowing the CPU to perform other tasks simultaneously."

An enhanced Level-III version of the TRS-80's Level II Microsoft BASIC and TRSDOS operating system are automatically loaded in memory when the machine is turned on. In addition, each time the computer is turned on, it tests itself.

Built-in I/O capabilities include two RS-232C channels and one Centronics parallel port. Future expansion is provided for via four plug-in slots for optional PC boards.

As for software, Jon A. Shirley, vice-president of the Radio Shack Computer division, noted that "About the same time we start shipments we will have the first of five business packages. All of these will run on the basic one disk version of the Model II. There will be a General Ledger capable of handling 500 accounts and a Payroll system that will handle up to 500 employees in up to three different states. An Accounts Receivable package will offer a variable number of accounts versus number of transactions ranging from 300 accounts with up to 8,000 transactions per month, to 2,000 accounts with up to 1500 transactions per month.

Another new package is a very capable Retail Inventory system that handles 3000 items. Finally, we will offer a mailing list system that can handle up to 4000 names. Mailing list management has proved a popular item for churches, school, groups, as well as for business. Prices for these programs will range from about $150 to $400, keeping them in a low price range for software of this capacity.

Also announced were several programs for the TRS-80 Model I, including a Payroll program for up to 100 employees, a Retail Inventory Control system for up to 1000 items, Accounts Receivable, Advanced Statistical Analysis, and Real Estate packages 1, 2 and 3. The real estate packages, which will be eventually expanded to an eight volume set, sell for $30 each and, according to Shirley, "represent an excellent example of how an under-$1,000 computer can provide a small business with very sophisticated software dedicated to a specific industry. These packages range from simple calculations of mortgage payments to the present value of income streams and other very complete calculations for the real estate investor." Also coming is a $99 FORTRAN, and a stock analysis package developed in conjunction with Standard & Poors.

The Radio Shack TRS-80 Model II will be sold at Radio Shack Stores and Computer Centers, and participating Radio Shack dealers, nationwide.

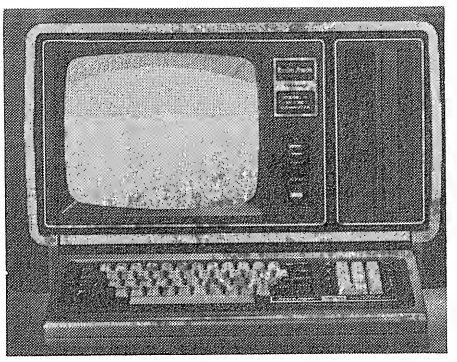
Radio Shack "will fall slightly below our goal of 50 centers by June

Stephen B. Gray is a frequent contributor to *Creative Computing* magazine.

30th," according to Charles A. Phillips, senior vice-president, operations. "As of today we have 38 open with the prospect of five more in June. The majority were opened in March and April 1979, and despite high, nonrecurring opening costs, many have turned profitable and we are optimistic about their sales and profit contribution to the company for fiscal 1980."

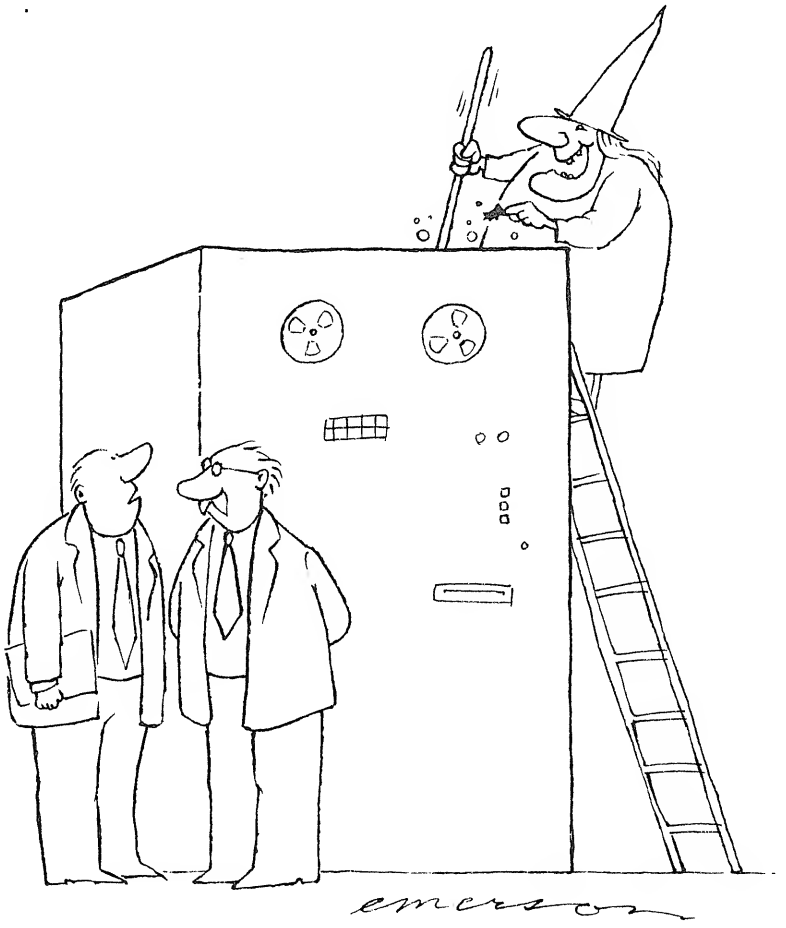
The Radio Shack Model II, with a basic price of $3450, falls in between the Model I, with a basic price of $599, and the Tandy 10, a relabeled ADDS System 70, with a minimum price of $8995.

In comparing the Model II with other computers, Roach asked, "How does the Model II compare to other small business computers? An IBM 5110 in comparable configuration would cost the end user between $15,000 and $20,000. Hewlett-Packard 9800 System 45 desk-top computer with printer sells for around $20,000. The Wang WCS-15 is over $10,000 without printer. The new Datapoint 1400 is around $6,000 without printer and is almost identical to the $3450 Model II configuration.



Radio Shack's TRS-80 Model II has a built-in 8-inch diskette drive, 12-inch monitor displaying 24 lines of 80 normal or 40 expanded characters, and a keyboard with 76 keys including functions such as Control, Escape, Caps, Hold, Repeat, and two software-programmable Special Function keys, plus a "calculator" keypad. From one to three additional diskette drives fit into an expansion chassis, mounted on or in a Model II System Desk. Price for the system shown, with 64k bytes of internal RAM memory: $6,599.

The TRS-80 Model II is a multiple-function machine. It can, on a small scale, be a general purpose data processing machine doing the traditional general ledger, payroll and accounts receivable type of applications. It can also serve as a low-cost word processing machine with a variety of print qualities available depending on the printer you place in the system. Additionally it's a terminal device with a standard communications interface that permits you to hook up to a telephone through a low-cost modem. Therefore, in one package you have an affordable intelligent terminal word processing computer system. The acronym guys can have fun with that." □



"New programmer?"

The TRS-80 Model II and Pickles & Trout CP/M

Andrew Klossner

Early in 1981, I needed a system on which to run CP/M version 2. My requirements included a fast Z80, a full complement of memory, large memory-mapped video display for word processing, a detachable keyboard with a good feel and upper and lower case, and an 8" floppy disk drive.

As a hardware novice, I needed service facilities. As a regular working bloke, I needed something inexpensive. My choice: a Radio Shack TRS-80 Model II, for which I paid $3499. After six months of daily system use, I have yet to regret it.

The Hardware

The design of the Model II is far better than the Radio Shack reputation led me to anticipate. It represents an excellent tradeoff between cost and features. It is also one of the first desktop computers to pay serious attention to human engineering and ergonomics.

In one package, the Model II incorporates an 80 x 24 character display and an 8" double density floppy disk drive. Inside there is a Z80 microprocessor ticking at 4 MHz, 64K of usable RAM, a Centronics printer parallel port, two RS-232 serial ports, and two real time clocks. The keyboard connects via cable. Since you pay for a single cabinet and power supply, you save a great deal over the cost of separate components, although you lose the ability to upgrade your system selectively.

External Organization

The keyboard features upper and lower case in the standard QWERTY layout. Special keys include Tab, Esc, Backspace, Break, Hold, Enter, F1, F2, and four cursor control keys. A numeric keypad lies to the right. The tilde, backslash, vertical bar, and rubout characters are typed by holding down the Ctrl key and striking "6," "9," "0," or "-." There is no way to type an accent grave; this annoying deficiency means that 127 of the 128 possible ASCII codes can be entered.

There are no "typeamatic" keys. Instead, a Repeat key is provided. Both a Lock and a Caps key are included; the former works like a typewriter Shift Lock, while the latter locks only the letters to produce all upper case. Both keys glow red when active, thanks to embedded LEDs.

The display uses a white phosphor. Warmup of the tube takes less than four seconds; it comes on before CP/M finishes booting. The character set is well designed and is pleasing to look at, with true lower case descenders. Besides blank and the standard 94 printable characters (including accent grave), there are 33 "business graphics" characters, useful mostly for margins and bar charts. Each character can be displayed in normal or reverse video.

The disk drive is mounted to the right of the display, facing the operator. My unit came with a Shugart 801 which seeks reliability at six milliseconds per track. This means that there is no meat grinder sound; head movement is whisper quiet. A disk expansion unit with up to three additional drives can be connected at the back.

The Power and Reset switches are located between the display and the disk drive. This position of the Reset key is far superior to a place on the keyboard (where it can be accidentally struck, to disastrous results) or on the back of the chassis (where you have to fumble to find it). To reset the computer, you must lift the key, so a falling object will not affect it. This is, unfortunately, not true of the Power switch, which is On when raised and Off when lowered. Power status is indicated by a red light between the two switches.

The Centronics and RS-232 connections are at the back of the chassis. After removing two screws and breaking the warranty seal, the cover can be pushed back and swung up and away. This is harder than it sounds; in learning the trick, which is described nowhere in the available documentation, I broke two of the four plastic pins which hold the cover flush with the back chassis wall.

Internal Organization

The system is based around a custom bus, which is designed to make full use of the Z80 family of parts. All peripheral interfaces are interrupt driven, and the disk controller is capable of performing direct memory access, so that disk I/O and computation can be overlapped. Boards are plugged into an eight-slot backplane, of which only four are used in an off-the-shelf unit.

The power supply is not extravagant; when the disk head is loaded, the display image contracts. A ventilation fan spins constantly when power is on.

The CPU board contains a Z80A, along with a DMA chip, two SIO ports, a counter/timer chip, and a 2716 ROM. The ROM is mapped into the lower 2K of memory upon power up or Reset. The bootstrap code performs extensive diagnostics, then loads the operating system from disk, which disables the ROM with an OUT instruction to leave RAM throughout the memory space.

The floppy disk controller board features a Western Digital 1791 controller chip which can manipulate single or double density disks. Although Radio Shack doesn't advertise it, the board contains all the necessary circuitry to run a double sided drive. It also has a PIO chip for the Centronics port.

The video/keyboard interface board holds an additional 2K of RAM, which can be mapped into the top 2K of memory at the flip of a bit in an I/O port. Each of the first 1920 bytes corresponds to a character position on the screen. A reverse video (black on white) character is selected by setting the high bit. A Motorola MC6845 chip drives the display and supports various sizes and speeds of blinking cursors.

This board also has a 60 Hz clock and the keyboard interface, both of which signal interrupts by pulling on the NMI (non-maskable interrupt) line.

The fourth board contains either 32K or 64K of dynamic RAM, depending upon which option you order. Bank select circuitry is provided, allowing any of up to

Andrew Klossner, P.O. Box 283, Wilsonville, OR 97070.

fifteen 32K pages to occupy the upper half of active memory. Further memory boards can be purchased to expand the system to as much as 512K, but neither TRSDOS nor CP/M supports more than 64K RAM.

The Software

The Model II comes with TRSDOS and a Microsoft Basic interpreter at no extra charge. However, since I bought my computer to run CP/M, I rarely boot the Radio Shack disk.

After researching the available implementations of the CP/M operating system for the Model II, I bought CP/M version 2.2 from Pickles & Trout. The BIOS (Basic I/O System, the code which must be customized to each new machine) in this package is well thought out. It includes several practical features, while eschewing the expensive flash incorporated into CP/M by many hardware manufacturers. Contrary to published reports in some usually reliable periodicals, no hardware modification is necessary to support CP/M on the Model II, and there is no point in asking a Radio Shack dealer to install the CP/M "option."

Perhaps the most useful aspect of this software is its support of virtual disk drives. This allows a program on a single-drive machine to run as though up to four drives were on-line. When the program attempts to switch to a new drive, CP/M displays a message asking the operator to mount the appropriate disk, then waits for Enter to be typed.

Another extremely important feature is a software type-ahead buffer, which allows keystrokes to accumulate until called for. Many word processing packages also incorporate this feature at the application level, but, unless it is present in the operating system, keystrokes entered during disk I/O activity can be lost.

P&T CP/M supports both the standard single density disk format and its own double density format which provides 600K bytes per volume. This latter is implemented with 512-byte physical sectors, and is a great improvement over the 480K available on a formatted TRSDOS diskette. Only a double density disk can be made bootable.

The operating system allows the baud rate of the serial ports to be selected independently. Any of three different handshaking techniques can be specified, to allow slow devices to request a pause in the flow of data while they catch up. I connect a NEC Spinwriter and let CP/M perform XON/XOFF handshaking; the printer sends control-S to stop output, and control-Q to resume.

The console output routine in the BIOS emulates a CRT terminal by accepting characters and placing them into video memory. Control characters such as carriage return, line feed, backspace, and tab cause the standard display result. Other control codes are defined to perform functions such as clear screen, line insert/delete, toggle reverse video mode, toggle graphics mode, and direct cursor addressing. A program can also request that the video RAM be mapped in for access, allowing near-instantaneous screen update by word processors.

Besides the standard assembly language program entry to the BIOS through location 0, P&T CP/M supports an alternate entry point at location 40 through which additional services can be requested. These include configuring the I/O ports, performing serial or parallel I/O, setting and reading the time of day clock, and selecting display options such as whether lines wraparound or the cursor blinks.

In addition to the usual utilities, P&T CP/M comes with programs to format and certify diskettes, configure I/O ports, and select disk seek rate. The IOFREEZE program allows the operator to set default attributes, which apply when the system is booted. AUTOEXEC can be used to select a command for execution upon every cold or warm boot. TRS2CPM reads a file from a TRSDOS disk. FASTCOPY copies one disk to another, and is optimized to function well with a single disk machine, where it fills all of RAM with data before asking the operator to switch disks.

The major drawback to Pickles & Trout CP/M is that the source code for the custom BIOS is not included. Even a listing would be valuable to the assembly language programmer trying to make full use of the hardware. For example, a program can request that a specific routine be given control when the counter/timer ticks, which happens 100 times per second. This routine must return via the RET instruction. There is no way to determine the contents of the user's registers at the time of the interrupt, or to modify the PC before returning, as a debugging routine might need to do.

Another problem, which may be either a hardware or software fault, is that activation of the Reset key when the disk is active does not completely reset the computer. Typically the display screen fills with garbage, the disk remains active, and subsequent use of Reset is ineffective. In these cases, the only way to recover is to power down the machine for fifteen seconds. This is especially inconvenient when a lengthy, disk I/O bound program must be halted.

For those who are handy at connecting different disk drives, P&T offer versions of CP/M which support double sided drives, for a capacity of 1.2 megabytes per volume, and versions which work with Winchester hard disk units.

The Documentation

The TRS-80 Model II comes with an owner's manual, which contains brief operating instructions, a TRSDOS manual, and a Basic manual. For thirty dollars you can get a technical reference manual, which is extremely complete and well written. It includes fully annotated schematics for every circuit and a separate chapter for each board and major component, as well as the manufacturers' data sheet for the LSI chips. It could be improved only by the addition of a listing of the boot ROM.

Pickles & Trout CP/M comes with the usual five manuals from Digital Research. In addition, a well written user's manual introduces the reader to CP/M and details the additional features provided by the custom BIOS.

A Personal History

When I first decided to buy a Model II, I approached the local Radio Shack Computer Center about the possibility of leasing, which they were pushing in a newspaper advertising campaign. It turns out that the lease program is intended only for businesses, not for consumers; my application was turned down.

I then did some shopping around to see what range of prices was available. All the stores that I visited which carried the Model II charged the list price of $3890. The hobbyist magazines, on the other hand, were full of ads from mail order houses with prices as low as $3400. I selected a dealer who is located within a comfortable drive of the factory in Texas and sent him my life's savings; shortly later, my new computer appeared at the local airport.

The system worked flawlessly for a month, but then began to show some flaky behavior patterns. These were characterized by a failure of the self test upon power-up or Reset.

In addition, the disk drive head began squeaking at the six milliseconds step rate, and I had to increase it to ten and put up with the meat grinder sound.

Eventually the machine failed so completely that the power-up sequence ended with a display of garbage, and I had an opportunity to investigate Radio Shack's claim of readily available service. I took the machine to the local Computer Center for maintenance under the terms of the ninety day warranty.

This first experience with the service department was extremely frustrating. To

begin, they promised to have my computer ready in 24 hours, or "48 at the outside." A week came and went while it sat unattended in the back of the repair depot despite daily telephone calls. In accordance with Murphy's Law, when they finally began to run their diagnostic package, it worked flawlessly, so they didn't bother to open the box. Eventually I retrieved it and spent the rest of the warranty period making regular impact adjustments with my shoe.

On the day that the warranty expired, a hardware-wise friend popped the cover and immediately spotted the problem. The bar which holds the boards against the backplane had too much give, and the boards were working their way out. Also, many of the socketed chips were not well seated. A bit of re-seating eliminated the flakies, and a spot of lubricating oil on the disk drive ended the squeaks. (The lubrication procedure is described in the P&T manual.) If the service technician had been at all experienced with the Model II, he would have recognized my symptoms and done this himself.

Several months later I had cause to return to the service department when the machine failed again. I was prepared for another nightmare, but this time I came away satisfied. There had been a complete turnover of service personnel, and the new technician diagnosed a bad disk interface board, swapped it for a good one, and made a variety of mechanical adjustments in less than two hours while I read science fiction in the front office. He also ran the P&T disk certification utility to check out the new board; TRSDOS has no analogous program. Although the computer was out of warranty and my bill was over $100 (of which $70 was for the board switch), I was happy with the experience and will not hesitate to return if another problem develops.

In all honesty, I must shoulder some of the blame for my computer's failure. For several months I used it both at home and at work, and it commuted with me. The resulting jostling was doubtless the cause of the loosened boards. The Model II is not advertised to be portable, and I have no doubt that if I had installed it in one spot and never moved it, it would not have failed.

Conclusions

The TRS-80 Model II is an excellent CP/M system for the person who is not well versed in hardware. To date, it is alone among inexpensive all-in-one units in offering a standard 8" disk drive instead of 5¼" minifloppies.

When purchasing a Model II, the buyer should ensure that the seller is an authorized Radio Shack dealer, and should get the dealer's "authorized dealer number" for presentation when obtaining service under warranty. If a Model II is bought from an unauthorized dealer, the warranty is voided.

Pickles & Trout CP/M represents one of the best jobs of customization in the CP/M domain. The resulting programming environment is head and shoulders above the more mundane efforts produced by many hardware manufacturers in support of their systems.

CP/M version 2.2 is available for $185. from Pickles & Trout, P.O. Box 1206, Goleta, CA 93017. □

GLOSSARY

Non-maskable interrupt—An interrupt is a signal that stops the normal program flow and causes execution of a specific vectored routine, after which control returns to the original program. Some interrupts can be disabled by setting a bit mask. A non-maskable interrupt cannot be stopped.

Controller chip—Any processor dedicated to control of a peripheral such as a disk drive.

Handshaking—Communication between a computer and peripheral or other computer, indicating whether a device is ready to send or receive data.

Wraparound—The continuation of a word from the right margin of a screen to the left margin of the next line.

Format (verb)—To place output in a predetermined form or arrangement. Also, to prepare a disk for data storage.

Source code—A program in human-readable form before compilation or assembly. Source code usually includes comments and meaningful labels.

Assembly language—A mnemonic representation of machine language. Assembly language uses op codes such as JMP (for jump) rather than the actual hex value that represents the instruction.

Register—A location in a microprocessor that is capable of holding a value. Used to load and store memory, and for internal operations.

Port—A hardware location used for input or output of information.

ASCII codes—The American Standard Code for Information Interchange, which represents alphanumeric characters and special controls as numbers.

White phosphor—The coating on the inside of a television picture tube which produces a white dot when exposed to an electron beam.

Bus—The lines connecting processors, memory, and other portions of the computer that send or receive data.

Z80/Z80A—One of the more popular microprocessor families, containing a powerful instruction set.

Interrupt driven—A device or routine that operates during interrupts.

CPU—Central processing unit. This is the microprocessor on which the computer is based.

DMA Chip—A chip which can read memory using Direct Memory Access rather than requiring information to be passed by the CPU.

Bootstrap—A routine that allows the computer to read other programs. In essence, the computer is pulling itself up by its bootstraps.

PIO chip—A chip dedicated to Programmable Input and Output.

Bank Select Circuitry—A system allowing a CPU to address more memory than it is designed for by switching between different banks of RAM.

512K Bytes Per Digital Cassette

Beta-80 From MECA

Eric VanHorn

Despite the numerous changes in computer technology, many peoples' everlasting impression of computers is large, spinning reel-to-reel tape drives in old science fiction movies. And despite many predictions of their demise, tape drives are alive and well, even in the small computer industry. MECA makes a number of tape drives for small computers, among them the BETA-80 for the TRS-80.

The BETA-80 has little resemblance to the slow and unreliable cassettes that most people have used at one time or another. Its shell is an unobtrusive gray, sheetmetal box that measures about 8″ on a side and 4″ high. The only control is a rear-mounted on/off toggle switch with a front-mounted on/off LED. The BETA-80 uses digital cassettes that load in the top of the box.

Interfacing depends on how the BETA-80 is to be used. The simplest configuration is to use the tape drive with a 16K Level II TRS-80, in which case a ribbon cable is plugged in to the edge connector at the left rear of the keyboard. Like the TRS-80 disk system, the operating system automatically boots when the keyboard is turned on.

The BETA-80 can also be used with an expansion interface as either a stand-alone bulk storage device or in conjunction with a disk drive as disk back-up. This does, however, require a minor jumper modification in the expansion box. If you are squeamish about making any hardware modifications, MECA will make the change for you. As a stand alone device, the BETA-80 automatically loads and executes the same way it does with the basic TRS-80. As a backup device, a port must be addressed using the OUT x,y command in Basic.

Because of its high speed operation, the BETA-80 requires digital cassettes. In a pinch it will work with regular audio cassettes, but they will not be reliable. The cassettes must be formatted, and contain two tracks with up to 999 256-byte blocks on each track. This gives a tape capacity of 512K per tape, although tapes can be formatted for a shorter number of blocks. Anyone who has had to work with the 56K capacity of TRS-80 disks can imagine what a luxury this kind of storage size is.

The primary commands in Basic are:

SAVE	(filename)	(track)
KILL	(filename)	(track)
NAMES	(track)	
LOAD	(filename)	(,R)
MERGE	(filename)	(track)

Filenames are up to 5 characters long. The track numbers, 0 and 1, are indicated the same way drive numbers are in disk systems. Save "TEST:1" would save the file test on track 1.

NAMES is the directory command. Up to 66 filenames can be saved on each track, so 132 files can be saved on a tape without running out of directory space. Because the directory is stored in RAM, the directory need not be read every time a NAMES command is issued. The Directory is automatically read in on start up and must only be reread if the tape or logged-in track is changed.

The BETA-80 operating system also allows for automatically loading and

executing programs. Load "TEST,R" will load the program TEST and Run it. This, among other things, makes it possible to have menu driven tapes.

The SAVE command also can be used for file handling. All files are saved as arrays. The information to be saved is stored in an array, then simply saved using the array variable. Even though Level II Basic only recognizes 2 byte variable names, the BETA can still use a five byte file name by only looking at the variable name and the first 2 bytes of the specified file name. The OS is "smart enough" to automatically convert from one to the other. File array names are all followed by an asterisk (*). Despite the fact the tape hardware is slower than a disk system, this simplified file handling process can actually make the overall speed faster.

Finally, file merges can be performed to concatenate Basic programs. This is not only useful for development work, but data statements can be merged during program execution to provide a different means of doing file handling.

For Basic programmers, this is as far as you will probably go, but for machine language programmers, the BETA-80 really shines. By getting out of Basic and into the operating system, a variety of versatile and simple instructions allow you to load, save, merge, and move machine language instructions. An OS command is also provided to dump memory. All these commands can also be used in Basic. The syntax is similar to that used in Newdos for accessing utility programs in the operating system. EX 4000 will execute a machine language program starting at a memory location 4000H. In Basic, CMD "EX 4000" will perform the same function.

Generally, the BETA-80 has been very reliable. Occasionally there will be a boot failure, but the system has always booted on the second try. I have been using MAXELL Digital Data Tape (about $8 each) and have never had a media problem. My only real complaint about the hardware is that the tapes can be hard to mount in the drive.

The software is extremely good — much better than I expected. Formatting tapes is slow — it takes about 20 minutes — but since the storage capacity is so great this is only a minor inconvenience. The software includes the OS, a debugger, Star Trek, and a mailing list program to demonstrate file handling. I also understand Electric Pencil is available on tape.

My only other complaint is the speed. The BETA-80 works at 4800 baud, which is certainly better than 600 baud cassettes, but I did wonder why they didn't go ahead and set it up for 9600 baud. That extra speed would make a big difference, particularly when compared to disks.

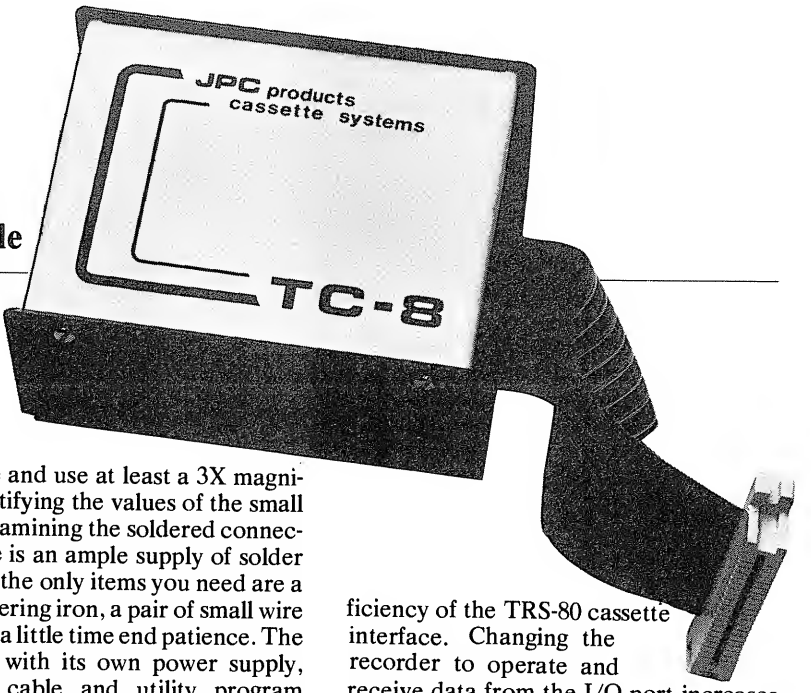
MECA tape decks are currently available for TRS-80, Apple, Sorcerer and S-100 computers, and can be ordered with an optional printer port. Anyone with one of the above computers should certainly consider tape as an alternative or complement to disk systems.

For more information, contact MECA, 7026 O.W.S. Road, Yucca Valley, CA 92284 (714) 365-7686. □

The Poor Man's Floppy

TC-8 Cassette Operating System

Robert C. Kyle



First came the expansion interface and disk drives for the TRS-80. Next came the Exatron Stringy Floppy (ESF) to serve as the "poor man's disc." And now there is the TC-8 Cassette Operating System (COS) to serve as the "poor man's floppy."

Manufactured by JPC Products Company of Albuquerque, NM, the TC-8 COS is sold in kit and and assembled form. The kit is easy to assemble, and the $30 saving is well worth the hour or two of effort required. I would strongly recommend that anyone in the bifocal days of his life also acquire and use at least a 3X magnifier for identifying the values of the small parts and examining the soldered connections. There is an ample supply of solder in the kit so the only items you need are a 30-watt soldering iron, a pair of small wire cutters, and a little time end patience. The unit comes with its own power supply, connecting cable and utility program tape. Yes, this device requires 550 hex bytes of memory to operate.

Nothing needs to be said about the inefficiency of the TRS-80 cassette interface. Changing the recorder to operate and receive data from the I/O port increases the speed of data transfer almost three times and increases the reliability of the transfer. The TC-8 COS connects to the

Robert C. Kyle, 3940 Oakland Ave., South, Minneapolis, MN 55407.

COMPARISON TABLE

EXATRON STRINGY FLOPPY	TC-8 CASSETTE OPERATING SYSTEM
COST $300	COST $120 (kit $90)
Special wafer cassettes w/ continuous loop	Standard audio cassettes Continuous loop cassettes available
4 bytes computer memory used	550 hex bytes computer memory used
Programs saved by numbers (1-99)	Eight character ''names'' allowed for saved programs
Booted on power up	Load short utility program (approximately 15 seconds) Disconnect recorder and plug into unit
Supports 7 units with individual addresses	Supports 2 recorders with individual addresses
Supports 7 commands—4 file commands	Supports 21 commands—4 file commands
Data files specified	Data files not specified
Requires special Data program (1K)	Included in utility program
SAVE and LOAD speed approximately 1K/second	SAVE and LOAD speed approximately 1K/second

I/O port with the included cable.

I had a bit of difficulty loading the utility program tape, but after 15 minutes of fiddling with the volume control on the recorder I got the program loaded. The TRS-80 format tape of that same program now loads in slightly less than 15 seconds. The original program tape is in three parts; the MONITOR, the full UTILITY and the modified UTILITY which is a short version of the full UTILITY. The modifed UTILITY and the full UTILITY programs are loaded as one program. The short version has its own starting address. If your memory is precious and you need every byte you can lay your hands on then just type KILL and ENTER and you have your TRS-80 back to its old self again. Even though you KILL the utility program, you still leave the debounce program in high memory (45 bytes).

The utility program supports 21 commands, four of which are file commands (OPEN, CLOSE, INPUT#, PRINT#). Basic programs are saved and loaded under the SAVE and LOAD commands. The SAVE command must be accompanied by a "filename," which can be any group of eight or less alphanumeric characters. This means that all of your programs can have real names which are recognized by the computer. The LOAD command does not require a filename, but if there is one, the program will search for it on the tape while listing the names of all the other programs it encounters. For example, let's say you have a game tape (TC-8 format) with BLACKJAK, POKER, CRAPS, and STARTREK programs stored. If you type LOAD "CRAPS" and ENTER (assuming you are starting at the beginning of the tape) your screen would look like

```
LOAD"CRAPS"
$BLACKJAK   $POKER   $CRAPS
READY
```

which would mean that the CRAPS program was loaded and ready to run. You could also type RUN "CRAPS" which would enable the program to be loaded and executed immediately.

What if you have a TC-8 format tape and don't know what's on it? Easy: just load it in the recorder and type LOADN. On the screen will be printed all the program names that are on the tape, but the programs will not be loaded. To exit the LOADN function just hit the BREAK key. System tapes are stored and loaded under the PUT and GET commands. Since they will only work with system tapes you can store system and Basic programs on the same tape. The "$" signifies a Basic program and the "%" is used for the system programs. If you GET or PUT a Basic program or SAVE or LOAD a system program you will get an SN ERROR message.

The TC-8 COS supports two recorders which can be addressed separately. The default is to recorder 1.

On power up MEMORY SIZE? is answered with 31400 and the utility program is entered under a standard SYSTEM procedure. The full program is executed by typing the / ENTER at the second ?*. If you want the short version just type /32150 and the screen will show BOOT READY. At this point you unplug the recorder from the TRS-80 cassette port and plug it into the TC-8 unit. If you have many programs to convert to the TC-8 format I would suggest you buy, beg, or borrow another recorder, connect it permanently to the TC-8 unit and leave the original recorder connected to the TRS-80 cassette port. That way there will be less wear and tear on the plugs and cord. Be sure to buy another cassette connecting cord for the second recorder. It is not necessary that your second recorder be a CTR-80 since the TC-8 supports a RESET command which frees the rewind and fast foward on any recorder that has these functions tied to the motor control jack. When the RESET command is used, the recorder motor remains on until you hit the BREAK key.

File management is similar to disk except that files cannot have names. The user controls the name by making it the first entry he writes to the file. Only one file can be OPEN at a time so there is no interchange between files directly. This can be handled through software subroutines.

The TC-8 COS stands up to comparison with the ESF admirably. If one considers the $90 kit price, the comparison becomes weighted excessively in favor of the TC-8. Therefore the comparison of features is based on the assembled price ($120) of the TC-8, which is about one half the cost of an ESF unit.

The manual accompanying the unit is excellent and the company is very receptive to any comments, suggestions, or complaints you have. Their address is JPC Products Company, 12021 Paisano Ct., Albuquerque, NM 87112. (505) 294-4623. □

Exatron's "Stringy Floppy" for the TRS-80

Fred Blechman

Even though I saw it advertised several times and read about it in two articles, I had no interest in the Stringy Floppy until I got my hands on one. Now I'm a believer! I'm not sure if the ads were too general, or the articles too technical, or that there's nothing like having the real thing and using it to really know what it can do.

Cassette Systems

Let's talk for a moment about the difference between cassette systems and disk systems. Cassettes are relatively slow. The TRS-80 Level II operates at about 500 baud — that's approximately 62 characters per second when loading a program onto a cassette from the computer, or loading a program from the cassette into computer memory. Cassette tapes are tricky to load, with head alignment problems, speed variations, tape variations, dropout, tape wrinkles, oxide flaking and such. You really have to CSAVE and verify at least twice for reliability. External DATA handling is too slow for most practical purposes. Changing programs requires making new copies, rewinding and then verifying with CLOAD? — just too time-consuming. However, cassette recorders are inexpensive. The tape cassettes are cheap (about 75¢) and are really great for "archival storage" — information you're going to keep for a long time and you're not going to use every day.

Disk Systems

Now look at a disk system. They're fast and wonderful — great for DATA handling, and extremely fast for loading and saving. But they're expensive! An Expansion Interface, with an additional 16K memory (which you almost have to get, since the disk operating system uses 12K all by itself — and with a 16K machine, that would only leave you with 4K) costs $448 from Radio Shack. The disk drive is another $449 — for a total of $937 (perhaps less if you have another source or use non-Radio Shack devices). The blank disks are about $5 each. And the disk system is also complicated, creating new problems for those who are not willing to spend the time and effort to learn it.

Fred Blechman, KGUGT, 7217 Bernadine Ave., Canoga Park, CA 91307.

The Stringy Floppy

Now there's another alternative, Exatron's Stringy Floppy for $300— a "poor man's disk."

Let me tell you some of the advantages. It's fast. (I mean fast for me. Maybe not fast for you people from disk-land.) It runs at 7200 baud, which is 14 times faster than the Level II cassette. Actually, it's 14.4 times faster. That's about 900 characters per second as compared to around 62 characters for Level II cassettes.

You don't need an Expansion Interface. The Stringy Floppy plugs right into the wall socket for AC power (no power stolen from TRS-80). It plugs right into the keyboard expansion slot, and has an extra connector to share the expansion port if you've got something already plugged into it.

You can put up to seven Stringy Floppies in daisy-chain fashion on one system, address them individually, and have them talk to each other — as compared to the normal maximum of four disk drives.

Automatic Keyboard Debounce

None of Exatron's literature or advertisements even mention what I'm about to tell you. When I discovered this I called up Bob Howell, Sr., the President of Exatron, long-distance to confirm it. He said, "Oh, yes, the Stringy Floppy automatically debounces the keyboard. I guess we should mention that . . . "

The Stringy Floppy requires *no RAM memory* from the TRS-80! It has its own EPROM — erasable programmable read-only memory. While it utilizes space in some operating system areas, it does not interfere with normally accessible memory.

It's self-verifying on loading and saving. When you tell the Stringy to save a program, it puts the program on tape, then goes back and checks every single byte. So you don't have to make two copies of everything, then rewind and CLOAD? verify.

Wafers

The Stringy Floppy uses little "wafers," $2 each, and smaller than a business card. In fact, I store my wafers in the jackets of plastic business card holders. Each can hold over 48 thousand bytes. Not bad for something this small. It's only 3/16 of an inch thick, and looking down from the top it's 2¾ inches by 1½ inches.

Inside, the wafer is a *continuous loop* of 1/16 inch wide tape — so narrow it looks like a string, hence the name "Stringy Floppy."

The wafers come in four different lengths: 5 feet, 10 feet, 20 feet and 50 feet. Just remember the numbers 4-5-6 and it's easy. A 4K wafer — that is, it will hold 4000 bytes — is 5 feet long and runs around its whole length in 6 seconds; 4K, 5 feet, 6 seconds. Now if you extrapolate that, 8K uses a 10 foot wafer and runs 12 seconds; 16K uses a 20 foot wafer that runs 24 seconds; and 40K fits on a 50 foot wafer that runs 60 seconds. Actually, I've found the 50 foot wafers really hold over 48K, so either the tapes are longer than marked, or the byte density is somewhat higher than 4K on 5 feet. (The $2 price is the same for all lengths.) Somewhere on it there's a little metal foil about ½ inch long, to indicate the end-of-tape/beginning-of-tape location to a pickup in the tape drive.

On top of each new wafer is a small ½ inch diameter silver paper reflective disk. If this is removed, or covered over with black paper, the Stringy Floppy will not record. In other words, if you want to protect a program on the wafer from being recorded over, remove or cover the silver disk. This wafer would then be called "write-protected." This is like removing the break-away tabs at the back of a cassette.

Other Things

The Stringy Floppy is fast enough to make DATA handling practical. An

internal buffer spits out 256 bytes of DATA about every second, just like that, into your computer memory — or from the computer to a DATA tape.

You can also load and save machine language programs, and a monitor program is available for machine language geniuses. (As for me, I've got my hands full with just Basic.) Incidentally, the Stringy Floppy does not interfere in any way with your regular cassette operation — you can CSAVE and CLOAD just exactly as you did before.

Installation

The actual unit is four inches wide, six inches deep and only two and a half inches high, and weighs about two pounds. The black and gray metal and plastic cabinet is a perfect match for the TRS-80. That's all there is to the installation. The wafer just pushes into the slot on the front of the unit. There are no controls on it; just two light-emitting diodes, one to show that the drive motor is operating, the other to tell you when it's writing on tape.

Using The Stringy

Use of the unit is simplicity itself. When it's connected and the computer is turned on, the display will show MEMORY SIZE? If you need to reserve memory for some other use — printer driver or whatever — type in the number you need in the normal fashion. When you press ENTER you'll be in Basic with a READY on the screen. Type in SYSTEM and press ENTER, then type in /12345 and press ENTER. The screen will now come up with:

EXATRON STRINGY FLOPPY
VERSION 3.2

and READY. You're in Basic and your keyboard is debounced! Check your memory with ?MEM and you'll get the same number you would without the Stringy Floppy on line (15572 for 16K unit with no memory reserved).

New Commands

You have three new commands when you've done this (see Table I). @NEW (and you can use an upper or lower case @!), @SAVE and @LOAD. These commands can be entered from the keyboard or can actually be placed in Basic programs.

The @NEW command initializes and verifies the wafer by turning on the drive motor (right LED goes on) and searching the tape for the beginning-of-tape foil. When a sensor spots the silver foil on the tape, the left LED goes on and the Stringy writes on the wafer tape with a special code. The tape is a continuous loop — it pulls out of the center, goes past the recording/playback head, and then winds around the outside of the tape pancake, like the common 8-track music tapes you have in your car or home. You don't ever have to rewind — in fact, you can't. As soon as the beginning-of-tape foil is located, the left LED goes out, the unit continues running, reading and verifying every non-byte on the entire tape! This assures you that the tape has no dropouts, snags, wrinkles or other nasty things.

Meanwhile, the screen says "ERASING . . . " (Shouldn't it say "VERIFYING"?) When the tape has been completely verified, the number of available bytes on the tape appears on the screen, followed by "DONE."

Saving A Program

The @SAVE command is similar to the cassette CSAVE command, and must be followed with a number from 1 to 99. 99? Yes, you can save up to 99 numbered programs on a single wafer. (You can do the same sort of thing on a cassette — in Level II, anyway — but who bothers? It takes so long for the tape, running at normal speed, to find the numbered program that everyone I know uses the tape counter and fast-forward manually.)

You must give the program a number, starting with "1" for the first program. The drive moves the tape forward until it finds the next available space. If you already have, say, two programs on that wafer, you would command @SAVE3. Once the tape has moved to the next available space, the record head writes the program, then the tape continues around and verifies with the computer every byte of the newly-written program before stopping.

In one operation you have accomplished what you usually do with a CSAVE, rewinding and a CLOAD? using a cassette.

Loading a Program

The @LOAD command is like the cassette CLOAD. If you don't follow with a number, it will load in the next program on the wafer. Give it a number, like @LOAD3, and it will seek and load that specific program only. Give it a number not existing on the tape and it will seek endlessly. (This wastes time but it is not otherwise harmful.)

The BREAK key stops the Stringy Floppy at any time.

You verify the loading two ways. The screen says "READING . . . " during loading and follows with "DONE" when completed. If there's an error, a "CHECK-SUM ERROR" or "PARITY ERROR" will appear on the screen — rare, in my experience, and not likely to occur if you try again. The second verification of a good load is to LIST the program. I've never had a bad load when the screen said "DONE." What a pleasure compared to cassette loading in Level II.)

Timing Comparisons

Getting down to the nitty-gritty, I have a chart that shows the timing

@NEW(n) — Verifies Ability of Tape to Hold Bits Along Entire Unused Portion. (n) Optional.
@SAVE(n) — Writes Numbered Program and Verifies Each Byte. (n) Required.
@LOAD(n) — Loads Next (If No (n)) or Specified Program Into Memory With Parity & Checksum Verified.
(Note: @ May Be Shifted or Unshifted)

Table I. Stringy Floppy Commands

PROGRAM	BYTES	SECONDS TO LOAD	
		LEVEL II CASSETTE (500 BAUD)	STRINGY FLOPPY (7200BAUD)
TRS232 Printer Driver	1734	32	2½
Telephone Toll-Charge	2853	48	3½
Simplified Bookkeeping	3163	54	3½
Telephone Dialer/Timer	5139	86	6
Distributor Records - Amway	7687	127	10
Order Verification - Amway	10417	171	14

Table II. Loading Time Comparisons

comparisons in loading several programs I use frequently (see Table II). The Amway Products Distributor Records program contains over 270 DATA statements (one for each of my distributors) and it needs to be updated every month. This used to be a real bother with cassettes, since every change required making a new cassette copy of the program, CSAVEd and verified twice. Each CSAVE or CLOAD? took over 4 minutes plus rewinding time. With the Stringy Floppy it takes under 45 seconds to @SAVE and verify —and I only have to do it once. That's over 16 minutes for cassette, versus under 45 seconds with the Stringy Floppy.

The Telephone Dialer Program is another example of how speed can be important. It offers the convenience of dialing numbers stored in memory — but can take several minutes to load if you have a lot of names in memory. With 67 names in memory it takes 86 seconds to load from a cassette, but only 6 seconds with the Stringy Floppy. Obviously, it gets used more often now than before I had a Stringy.

Data Handling

Some programs require data be stored outside of the regular program itself. Inventory, mailing lists, accounts receivable and many other data bases are usually handled this way. With cassettes it's a bummer. Loading external data into a program from a cassette, can take 30 minutes or more, since it's usually done line-by-line.

However, a special data I/O program is supplied for the Stringy Floppy. It lets you operate on 256 bytes at a time, with no serious loss of speed. The program occupies less than 1K and loads quickly from a wafer (taking about one second to load).

Table III. Data I/O Commands

@OPEN(n) — Open Specified Data File
@PRINT — Records Data on Wafer Tape
@INPUT — Reads Data Into Memory
@CLOSE — Closes Data File
(Note: @ May Be Shifted or Unshifted)

The data I/O wafer gives you four new Basic commands (see Table III). These are similar to cassette or disk file commands, and can be directed to any of up to seven Stringy Floppies on line. The special I/O commands are normally imbedded in Basic programs.

For example, I have an order checking program I use almost daily in my Amway business. It holds 260 DATA statements which are loaded into a two-dimensional, 6-column by 260-row array with READ statements in the program. Because the resident DATA statements take up about 6500 bytes of my 16K memory, I'm limited to 260 stock numbers and prices. Once the data items are loaded into the array by the program, the data is just occupying memory for no purpose. I found I could use a data cassette, but it took almost 30 minutes to load the data into the program. However, using the Stringy Floppy data I/O program, reading the data into the array from a wafer takes only 45 seconds and frees 6500 bytes of memory — which allows me to put almost 500 stock numbers and prices in an array instead of 260! Now that's what I call an improvement.

Machine Language

You can also @SAVE machine language programs if you know the starting address and byte length. An autostart address is optional. A monitor wafer is available for machine language debugging; it includes a memory relocator and separate manual. Level III Basic is also available on a wafer.

Manual

Although I've had no experience with disks or exotic peripherals, I followed the user's manual easily. It's so very explicit, with examples and explanation of error messages. It even has a selection on Assembly Language Operations for those of you who understand that stuff. And for the hardware types, a parts layout and complete schematic are also included.

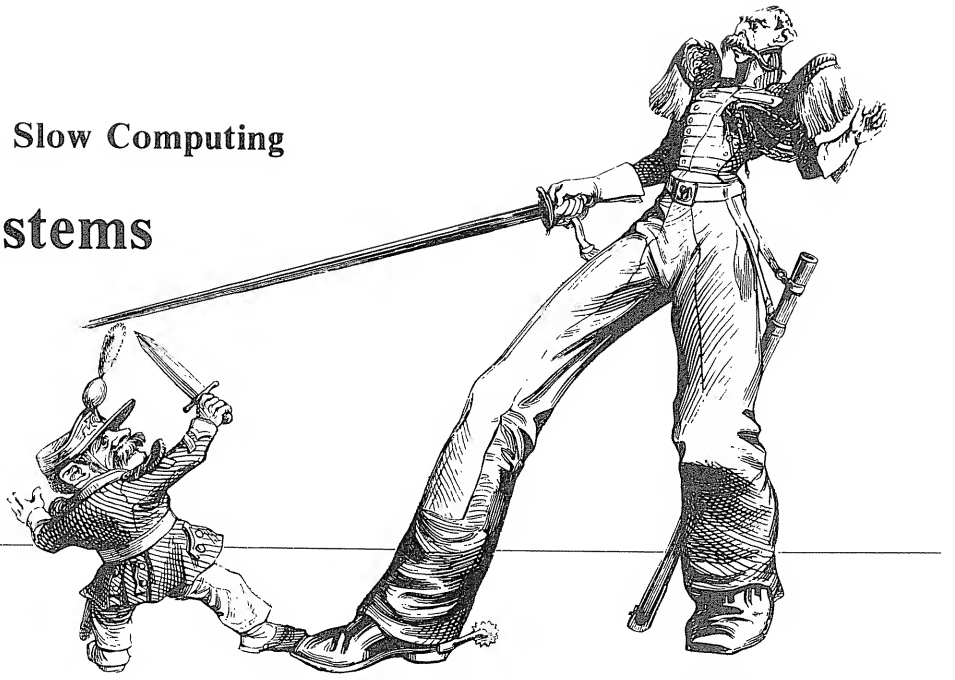
Guarantee

Exatron sells the units with a 30-day unconditional moneyback guarantee. Besides the TRS-80, Stringy Floppies are available for the SS-50 and S-100 busses as well. The cost for the TRS-80 version is $300, with the other units comparable.

Exatron doesn't have any dealers, so you'll have to contact them directly. Their address is 181 Commercial St., Sunnyvale, CA 94086, and they have a hot-line toll-free number: (800) 538-8559, except in California, where the number is (408) 737-7111. □

Winning the Battle Against Slow Computing

Floppy Disk Systems



Ken Knecht

I just got my first disk drive for my TRS-80 and I'm very happy with it. Ran perfectly ever since I unpacked it.

I'd been using Level II BASIC and was anxious to have the much faster saving and loading of programs with the disk, and since I write business programs for part of my livelihood I was anxious to try those I had written while waiting for the disk.

I've previously used MITS 4.0 disk BASIC and will first give a few comparisons. Incidentally, this is version 2.1 of the TRS-80 DOS (TRSDOS).

Comparisons

A random record is 128 bytes long in MITS, 255 in TRS. The maximum number of records that can be stored on a disk in MITS is 2046, 329 on a TRS disk (remember though, the TRS records are essentially twice as long). You can store 255 files on a MITS disk, 48 on a TRS disk.

It's obvious that the TRS-80 disk will hold much less data, roughly 1/3 as much. The total capacity of a MITS disk is 250K bytes, a TRS-80 55K or 85K bytes. We'll discuss this "or" later.

I'll have to admit I'd sure like more disk storage for some programs, for instance when I have to keep a very large data-base on line, but in most other programs I find I have more than enough room. For one program I find I'll need four drives, and would like to use one more, but TRSDOS will only handle 4 drives. Perhaps later on we'll have a dual density drive available, or full size floppies, or whatever, to choose from.

The disk drive runs much more quietly than does the MITS; no fan and the head loads very gently. You can't hear the motor run unless you listen for it. Also, the motor only runs when the drive is being used, not all the time.

Ken Knecht, 1890 W. Colorado St., Yuma, AZ.

Storage Capacity

I mentioned earlier that the total disk storage area was 55K or 85K bytes. Well, if the disk is in drive 0 it has 55K bytes of storage, in drives 1, 2 and 3 it has 85K bytes. The reason for this is the DOS system keeps most of its utility programs and the extensions to Level II BASIC on the disk until needed. Thus 30K of disk space is pre-empted by this software. This disk must be in drive 0 to use the system. I'm wondering if all the extensions to Level II BASIC might not run with a blank disk in drive 0 after disk BASIC is loaded. I can't try this until I get my second drive and can FORMAT a disk.

Now you are probably wondering if you are going to be stuck with only one usable disk if you only have one drive. No problem, you can copy this software onto another disk quickly and easily, thus making as many drive 0 disks as you wish. Yes, you can copy a disk even if you only have one drive!

I wish that the records were 128 bytes, as in MITS disk BASIC, instead of 255 bytes. Note 255, so you can't fit in two 128 byte records. This means I'll have to do a lot of rewriting to use my old MITS programs using random disk files on the TRS-80. Rats!

The Basic

However, the disk BASIC is very much like that of MITS, almost all BASIC commands are the same, with the TRS missing a few present in MITS disk BASIC, such as SWAP, RENUM, ERASE, WAIT and possibly a few more. The only one I really miss is RENUM, and that is available on tape as a System program. So if you are used to MITS BASIC you will find the changeover an easy one. I suspect it would be from most other BASICs as well, as the dialects aren't all that different. If you write your programs in modules to be loaded off disk as required, as I do, then you can generally use CLEAR to replace ERASE.

Two commands conspicuous by their absence in the TRS disk BASIC are MOUNT and UNLOAD. Nope, no equivalent either. Just put the disk in the drive and go, or pull it out when you wish. Of course, don't remove the disk if the LED on the drive door is on, indicating it is reading or writing. So much for MITS. On to the DOS. Now it gets interesting.

The Disk Operating System

The DOS is more like you'd expect a DOS to be. That is, like a micro version of a big computer's operating system. And that's what it is. BASIC is just one of the programs it calls. There are many other interesting programs and features as well.

The TRSDOS permits file protection — at three levels. You can protect a file to be available in any of the following categories, with or without a password.

Execute
Read, execute
Write, read, execute
Rename, write, read, execute
Total Access

You always have total access with your password. Of course if you forget the password you have problems. But each disk also has a password, allowing you to change the file passwords if

you know the disk password.

Of course you can change the passwords and protection whenever you wish. You can also make a file invisible to a normal DIR (file name listing) if you wish. This DOS command is ATTRIB with the various options.

You can use the DOS command APPEND with the names of the two files to add one sequential file to the end of another. This is very useful to extend a sequential file without having to write a subroutine to re-record it.

Then there's the DOS DIR mentioned earlier. This is like FILES in MITS disk BASIC only more so.

DIR prints out all visible BASIC file names (remember the invisible files under protection?). If you add S (DIR : (S)) then it also displays System files. An I gets you all invisible and non-system files. An A gives you the disk allocation for the files displayed. So DIR : (S,I,A) gives you everything.

Notice two things, the disk drive number in all cases is optional and spaces, especially in DOS commands, are critical. Even in BASIC some are critical. For example, FIELD3,128ASC$ is illegal, it has to be FIELD3,128 AS C$ because ASC is a reserved word. Earlier versions of MITS disk BASIC had this problem too, but not 4.0.

You might have noticed the Systems files I mentioned. These are machine language program files, used by DOS or user generated.

The following is a list of more DOS (not BASIC) commands:

The DUMP command permits loading a file located in a consecutive series of memory locations, for example a machine language program. You also specify the loading address for when the file is read.

FRE lists the free disk space on all used drives in terms of files available and granules (1.25K bytes, or 1/2 track) by drive number.

LIB lists all available DOS commands in that version.

LOAD loads a machine language program file into memory. This is different than the disk BASIC LOAD.

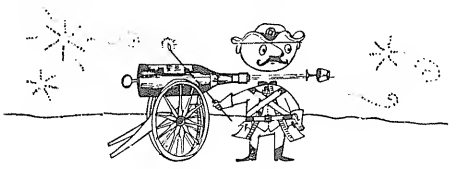
PROT changes file and disk passwords. It permits changing all the passwords to one, or to remove the passwords, or change the disk password (all only if you know the disk password).

RENAME permits changing a file name. This does not affect that file's password or protection.

VERIFY(ON) causes DOS to verify all BASIC disk write operations. That is, read them after they have been written to be sure they were recorded properly. The DOS system disk writes do this automatically. The default is VERIFY(OFF).

TAPEDISK is a DOS utility program which allows loading Radio Shack (and other?) System tapes into RAM, then into a file on disk. Programs must load after 54F4 hex in memory. Which eliminates most programs.

DISKDUMP/BAS is a BASIC program (run under BASIC, not DOS) which permits you to make a sector by sector examination of any specified user file. It prints out each sector in hex, then with corresponding ASCII code, if printable. This shows how data is stored on the disk. Educational.



CLOCK turns the clock and display on and off; it is displayed in the upper right corner of the CRT screen. The DOS commands to set the time and date are TIME and DATE. The clock and date can be read (and printed) under BASIC with RIGHT$(TIMES$,8) for the time and LEFT$(TIME$,8) for the date.

TRACE turns on or off a display next to the clock showing the program counter contents in hex. I haven't figured out a use for this yet. This is different than the BASIC TRACE command.

AUTO is another DOS routine. This forces the DOS to load any DOS program upon power on. This could be used to load BASIC automatically when you turn on the computer. It can't force a BASIC program load because BASIC hasn't been loaded yet. Only one argument allowed.

BACKUP is the DOS program used to copy disks. It will also copy itself, and can copy with only one drive. It automatically FORMATs (initializes) diskettes if this has not already been done. It will only copy to a disk with no data stored on it. Therefore you must bulk erase a disk before you copy to it. That makes it pretty difficult to erase a disk by mistake.

When BACKUP runs it tells you which track it's FORMATing as it does, then tells you as it reads and verifies each sector, then the copy routine begins. DOS tells you as it loads each track into RAM for a one drive BACKUP. It loads as much data from the disk as it has room for in RAM, then requests you to change disks, then puts the data into the same tracks on the blank disk, and so on. Runs pretty fast too, only a few minutes and disk changes for a single drive BACKUP. MITS takes 30 minutes for a two disk drive copy.

COPY records a file under a new name, retaining the old file under the old name. Passwords and protection remain the same.

FORMAT initializes the disk. You can't FORMAT with only one drive, but if you only have one drive you need to use BACKUP to copy the 30K bytes of software you need to keep on the drive 0 disk anyway, so this is no loss. BACKUP does its own FORMAT. FORMAT takes less than a minute, not 7 minutes like MITS DOS.

The last program is a DEBUG, called by DOS. This is a pretty standard debugger; it permits you to display (in hex or ASCII), or modify, any address in memory, load a register pair, display registers, display memory sections in full screen, set up to two breakpoints, and single step or single step with CALLs completed.

File names are the usual 8 characters, but you can add up to three characters as an extension: thus STARTREK or STARTREK/ONE or STARTREK/#1, etc. Nice for different versions of the same program.

Another nice feature, drive numbers are optional. If you LOAD a program in DOS or BASIC without a drive number all drives will be searched until the program is found. Watch out for the same file name on several disks. Also, if you SAVE a program with no drive number specified it will be saved on the first disk found with enough room to hold the file.

Of course you can return to DOS from BASIC at any time. Just type CMD"S". To get to BASIC type BASIC to the DOS READY.

Disks are dated when they are initialized or copied. The date shows up when you use DIR.

Some Closing Thoughts

A few more interesting items I should mention. When you return to DOS from BASIC you lose the program you had stored in memory. A bit inconvenient at times as you always have to remember to SAVE a new program before you leave BASIC, even if only for a DIR. Of course this also means you can't use some of the more useful (to a program) DOS commands in a program like APPEND, FRE and RENAME.

Another thing you should note, the minimum length of a random or se-

quential file is one granule (1.25K bytes or 1/2 track). Thus one record random files are very wasteful, as are short sequential files.

When you FORMAT a disk you are notified of any bad tracks on the disk. These are locked out automatically. You are also given the option of locking out other tracks, and FORMATing or not these locked out tracks as well.

If you use the reset button on the computer you lose any program in RAM.

Power up is very easy, it's all automatic. Just put a DOS disk in drive 0 and turn the system on. I use a power strip to provide power for everything and use its on/off switch. The disk LED goes on for a few seconds and you get DOS READY before the monitor has warmed up. No loader programs to fool with or switches to flip or buttons to push.

You can also use your Level II BASIC and Systems tapes in two ways without going to the trouble of disconnecting the expansion module. Ask for BASIC2 instead of BASIC when you get DOS READY. Push the reset button on the computer to return to DOS. You can also get Level II BASIC when you power up if you hold down the BREAK key when you turn the computer on.

I'd recommend you purchase 16K of RAM in the expansion module for a total of 32K. With 10K of RAM used by DOS and disk BASIC the 6K left is not very much.

Incidentally, the DOS disk is free with the purchase of the disk drive; MITS charges an extra $200. Radio Shack gets $5.98 for additional blank disks, I can generally find them elsewhere at $3.50. ■

Percom Doubler II and DOSPLUS

A Winning Combination for Double Density on the TRS-80

Ronald H. Bobo

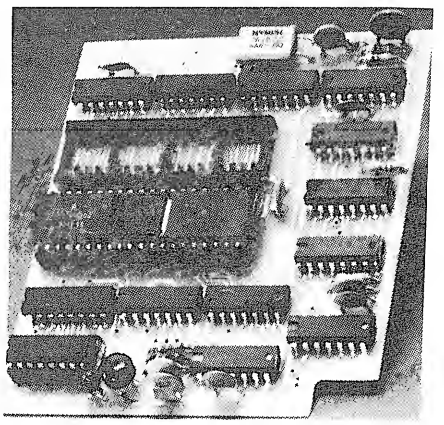
Apparently, the amount of software in one's possession expands to fill the storage media available. As the proud possessor of over 70 disks full of software, I recently decided that the time had come to make some changes.

As the owner of a TRS-80 Model I, I had three choices for storage: tape, floppy disks or hard disk. Since I was already using a pair of mini-floppies, and hard disk drives were out of my price range, another solution was needed.

That solution turned out to be a double-density disk controller.

Without getting technical, I will say that double density pertains not to the number of tracks on a disk but, rather, to the way those tracks are formatted and the manner in which information is recorded on them. Whether you now run a 35, 40, 77 or 80 track drive makes no difference; after adding double density, the number of tracks will remain the same, but more data will be packed into those tracks. Actually, the disk capacity is not really doubled; an increase of about 70% is what should be expected.

After reading advertisements for the Doubler II from the Percom Data Company of Garland, TX, I bought one from



my local dealer. The cost for the complete package, which also contains a disk operating system, was $169.95.

The Doubler II consists of a small printed circuit board containing a double-density controller and supporting chips, including an external data separator. The data separator corrects a design deficiency in the TRS-80 Model I which leads to CRC errors and locked out tracks when formatting the innermost tracks of disk.

The Doubler consists of a small printed circuit board containing a double-density controller and supporting chips, including an external data separator. The data separator corrects a design deficiency in the TRS-80 Model I which leads to CRC errors and locked out tracks when formatting the innermost tracks of disk.

The board also contains one empty 40-pin IC socket. Installation is simple and well documented. Simply remove the bottom cover from the expansion interface and locate the existing disk controller chip. Carefully remove this from its socket and install it in the empty socket on the Doubler board, taking care that Pin 1 (on the end with the notch) faces in the same direction as the other 40-pin chip. Now, plug the board into the socket in the interface, again making sure the orientation is correct.

A word of caution: if your expansion interface is still in warranty, opening it will void that warranty.

On bootup, the controller determines whether the disk is single or double density and automatically selects the correct mode. As with nearly everything, there is an exception to this, the LDOS operating system, which will be covered later.

The LNDoubler

LNW Research of Irvine, CA, also manufactures a double density controller. The LNDoubler is priced at $175 and

Ronald H. Bobo, 3246 Gravois, St. Louis, MO 63118.

includes a copy of the DOSPLUS 3.3D operating system from Micro Systems Software.

Micro Systems Software has run full-page ads in most of the computer magazines offering a reward of $100 to anyone who finds a bug in the system. I can attest to the fact that this is a bonafide offer, since I found one! I reported it by phone, and several days later received a check. The bug, of course, has now been fixed.

Reports from owners of two different systems using the LNDoubler indicate that it will format some disks which have been rejected by the Percom unit. The LNW unit also has an onboard data separator.

DBLDOS

Percom's Doubler comes with a disk containing DBLDOS. For many months, DBLDOS and OS-80 (also from Percom) were the only double-density operating systems available for the Model I. Others are now available.

DBLDOS is a bare-bones system. Apparently similar to both TRSDOS 2.1 and NEWDOS+, it is adequate for getting started, but most people will probably wish to upgrade to a more powerful system.

Although DBLDOS has provisions for transferring programs between single- and double-density disks, it cannot read a single-density directory. When you begin to transfer your files to double density, you had better know what is on the disk.

Percom and other sources are now selling patch programs called Doublezaps which modify the NEWDOS-80 and VTOS 4.0 operating systems for double density compatability. Since I had been almost exclusively a NEWDOS-80 user, I was very much interested in this patch.

After observing it, I was somewhat disappointed. While orders of magnitude better than DBLDOS, the patched NEWDOS-80 is not compatible with single-density disks. It will read a single-density directory, which is an improvement, but any program to be run must first be transferred to a double-density disk. Some of the excellent copy routines of NEWDOS-80 were lost, when the original format routine was disabled and a double-density formatter substituted.

NEWDOS-80

Later, the opportunity arose to examine a NEWDOS-80 which had been patched with Doublezap II. A more expensive patch, this one includes a utility program called ADR/CMD for automatic density recognition. This allows you to have either single- or double-density disks in drives other than 0, which contains the double density system disk.

This appears to be a great improvement over the original Doublezap. The special NEWDOS-80 copy routines appear to be intact, and transferring files between single- and double-density disks is much simplified.

Making a backup, however, has to be done in two steps. First the disk must be formatted using a utility called DBLFMT/CMD. It is then copied using the NFMT option which, of course, is No Format. At some point in this operation, another utility, FIXBOOT/CMD must be run to apply the proper Boot sector to the backed-up disk.

While these patched programs do work, I am sure that NEWDOS-80 Version 2.0, which should be available by the time you read this, will be much better. Apparat is apparently doing a complete rewrite on the DOS and, as a bonus, Model I and Model III owners will be able to exchange files if both are using their respective versions of NEWDOS-80 2.0.

I have had no chance to observe the patch VTOS 4.0, so will reserve comment.

Both DBLDOS and the patched NEWDOS-80 will abort a format if five tracks are locked out. Since few of my disks are certified for double density, this happened several times.

In particular, the Doublezap II for NEWDOS-80 would not format anything in its original configuration. The format routine used puts a pattern of 6DB6's on the disk sectors. This is a worst-case pattern for double density. DBLFMT/CMD would not format any disk I had, including those rated for double density, until a zap was applied which changed the formatting pattern to the more common E5.

After installing the Doubler II, I tried again with the 6DB6 pattern. It formatted and verified 40 tracks on everything but a very cheap, unbranded diskette. Even this could be formatted with 35 tracks.

Two of the later entries in the double density disk operating system field are LDOS and DOSPLUS.

LDOS

Some purchasers of LDOS were in for a surprise. Advertising for this close relative of VTOS 4 claims to support double density, and perhaps it does if one owns a Lobo expansion interface. (The DOS was originally marketed by Lobo Drives International of Goleta, CA. A separate organization, Logical Systems, has now been set up for this purpose.) Owners of the Radio Shack interface, however, found it necessary to obtain an additional driver program in order to activate the double density mode.

Activation is somewhat cumbersome, requiring that the system be booted in single density after which the driver may be loaded in one of two ways and, after this ritual is complete, one may switch a double-density diskette into Drive 0.

Once activated the system will read both single- and double-density diskettes automatically.

Perhaps it is not entirely fair to call LDOS a close relative of VTOS, since many improvements are said to have been made. With the skimpy VTOS documentation, who knew what capabilities that system had, anyway? Lobo has corrected this, issuing a manual of some 250 pages for LDOS.

One of my greatest objections to LDOS, whether single or double density, was its extreme slowness, a carryover from VTOS. If you called up a directory with the original version, it helped to have your lunch handy. A fix for this has now been implemented and LDOS has become a much better system.

Otherwise, LDOS appears to be a great operating system, containing many features usually found only on larger computers. Its owners swear by it.

SOFTWARE PROFILE

Name: DOSPLUS 3.4D
Type: Disk Operating System
System: TRS-80 Model I or III
Format: Disk
Language: Machine Language
Summary: Impressive and useful
Price: $99.95
Manufacturer
Micro Systems Software, Inc.
Hollywood, FL

DOSPLUS

DOSPLUS seems to have been written with double density in mind from the beginning, and is the only operating system evaluated which gave me that impression. No patches or zaps to apply here, just stick the disk in the drive and go.

Priced at $99.95, DOSPLUS 3.4D is available from Micro Systems Software, Inc. of Hollywood, FL. A version is also available for the TRS-80 Model III.

DOSPLUS is compatible with all single-density operating systems for the TRS-80 Model I. What this means is that, as long as the double-density DOSPLUS disk is in drive 0, it doesn't care whether the other drives contain double- or single-density disks. DOSPLUS will read from or write to either one.

LDOS and DOSPLUS will also read one another's disks, but are not compatible with any other double-density system I have checked.

Some impressive utilities are resident on the DOSPLUS system disk. CLRFILE zeros a disk file without reallocating the file space. COPY1 is a file copying routine for single drive users. CRUNCH is a compression utility which removes unnecessary spaces and REMarks from Basic programs, the better to conserve memory.

CONVERT modifies TRS-80 Model III TRSDOS data disks to allow reading by the Model I. Since Model I and Model III DOSPLUS data disks are already compatible, no conversion is necessary.

DISKDUMP allows display and modification of a disk sector, accessed by disk file name. DISKZAP is similar in capabilities to the many zap programs on the market, but can be toggled to operate in either single or double density. Access by file name is not available in DISKZAP—the desired track and sector numbers must be specified.

The FORMAT utility allows formatting a disk with any number of tracks from 35 to 80 in either single or double density.

PURGE is just that, a means of rapidly deleting unwanted files from a disk. If you purge too many, RESTORE will recover them, or any other file which has been inadvertently killed, so long as it has not been written over. The exception, files which have been killed under TRSDOS are completely expunged from the directory and no recovery is possible.

RENUM is used for renumbering all or part of a Basic program.

SCRIPFIX is a patch which modifies your existing copy of Scripsit or SuperScript to operate under DOSPLUS.

SPOOL is a printer spooler that works. You can allocate a block of memory to the printer, and go ahead and do something else with the computer while it prints out what is in the reserved block.

TRANSFER copies all non-system files from one disk to another—very handy when you are moving files from single- to double-density diskettes. MAP prints out a listing of file allocations showing just where each one is located on the disk.

Library Commands

In addition to the usual Library commands, DOSPLUS contains a few which may be new to you.

BUILD creates a file composed of a chain of commands, which is then started executing by DO. By using DO in conjunction with AUTO, it is possible to perform a great many tasks in sequence simply by booting the disk.

CLEAR zeros memory above 5700H. CONFIG is used to set disk drive parameters, as well as telling the computer whether or not to send graphic characters to your printer.

CREATE allocates space on the disk for a file. Reserving this space in advance will usually cause your file to be recorded in one continuous block, rather than being scattered over the disk.

FORCE lets you set parameters for printer drivers. When you print out a LISTing with this system, it is very neatly divided into pages.

Typing FREE while running DOSPLUS does not produce the customary line telling how many granules are available on each drive. Instead, a disk map is printed out showing the status of each granule. "X" signifies that the granule is in use, a period indicates a free granule, "L" is a locked out granule, and "D" shows the directory granules.

The primary use of PAUSE is to stop execution of a DO file until the operator presses Enter, as when a disk must be changed. By means of the CMD function, it may also be used from Basic.

RS232 reads the switch settings on the RS232 serial port if one is installed, and allows changing them in software.

Extended Z-80 Disk Basic

DOSPLUS Extended Z-80 Disk Basic is compatible with Radio Shack Disk Basic, but minor syntax changes may have to be made if the following are used in a program statement:

PRINT &O for octal conversion is not supported.

CMD "S" will return an error message. To return to DOSPLUS, it is only necessary to type CMD and press Enter.

CMD "I" has been eliminated. To execute DOS commands from Basic, type CMD, enclose the desired command in quotes, and press Enter.

Unlike the Radio Shack system, no file buffers are reserved automatically upon entry into Basic. Both number of files and protected memory must be specified at the time Basic is called. For instance; Basic filespec-F:3-M:61000 will load Bas , open three file buffers, protect memory above 61000, and load and run the specified Basic program.

DI, when followed by two line numbers, as DI 10,100 will delete line 10 and insert it as line 100. DU is similar, except the first line is not deleted, it is duplicated at the new location.

Several new one- and two-character abbreviations have been added to make editing easier.

OPEN "E" opens a sequential disk file for output and places a pointer at the end of the file. Data may be added to the file without having to load it into memory first.

Variable length records are supported in the random file mode, up to a maximum of 256 bytes. Simply use the following format:

Open "R",1,"filespec",51

Your record length will be 51 bytes. If you specify a length that is not evenly divisible into 256, file space is not lost; records will span across sector and track boundaries.

The trace function in Basic, turned on and off by TRON and TROFF, is considerably different from what TRSDOS users are accustomed to seeing. The DOSPLUS version is actually a Basic single-stepper. Statements are executed one at a time by pressing either ENTER or the space bar.

CMD "REF",K,L,V prints out all references in your program to keywords, variables and line numbers. They may be referenced either all three at once, as above, or one or two at a time. Adding a "P" outputs the display to the line printer.

CMD "M" displays all variables currently allocated, along with their values.

Would you like chaining capability between Basic programs? Adding a comma and "V" will save the variables from your current program, and carry them over to the program "TEST/BAS." Variables set as a literal or through DATA statements are not saved.

A global search and replace feature is included for ease of mass editing. Activated by CMD "SR", it will search the entire Basic program and display or replace any string variable or expression. From an example given in the DOSPLUS manual, CMD"SR",":",CHR$(10)+":" will search the whole text and insert a line feed in front of every colon, thus making the program easier to read.

As if all that were not enough to make a great system, Micro Systems Software has also included TBasic on the DOSPLUS disk. This is a tiny Disk Basic which will give users approximately 3K of extra memory space.

While DOSPLUS documentation is not extensive (it consists of a 48-page manual), it is written in an easy to understand style and appears adequate.

If you have the impression by now that I think double density is great, you are absolutely right.

Transferring most of my files to double density allowed recovery of more than three boxes of disks. Now, if the amount of software doesn't expand too fast... □

Telecommunications for the TRS-80

The Ultimate Connection

Frank H. Marz

No, it wasn't love at the first sight, but after a few days with the Microconnection, it became a realistic and real affair. This is an unbiased evaluation of the Microconnection from the Microperipheral People for which I also relied upon CompuServe's Information service (formerly called MicroNet).

The basic computer arrangement used for the review, consisted of the TRS-80 computer, Level II with 48K of RAM, three disk drives, Microline 80 and IP-125 printers, using at various times TRSDOS 2.2 and 2.3, followed by NEWDOS+ and the latest, most powerful NEWDOS 80 from Apparat, Inc.

For providing maximum support in computer communication for the hobbyist and general computerist, CompuServe Information Service (whose Director of Application Software, Mike Ward, was most understanding of my immediate needs and quickly supplied a special evaluation ID number) deserves all the thanks of the author.

The Microconnection allows telecommunication for all TRS-80 computers, be it a Level I, a Level II, Model III, or the Color Computer. No interface is actually required for the connection to CompuServe, Forum 80, or most other community bulletin boards. The Microconnection also has a serial I/O for a lineprinter.

A serial I/O socket (female DB25) protrudes from the back of the Micro-

Frank H. Marz, Rt. 4, Box 1, Delavan, WI 53115.

connection along with other screw-terminal I/O arrangements, which permit the use of Ham communications or a tape recorder. Simplex/duplex mode is selectable at 300 or 110 baud. The basic package measures approximately 7.5" x 4" x 2". A cabled modular phone plug of approximately six feet is provided, allowing connection to your modular phone master box. Another wire connects to the power transformer, which is of sufficient watt rating to eliminate any possibility of overheating or other overload problems.

Two LEDs protrude from the top of the instrument package, one to indicate power-on condition and the other to indicate when the carrier has been established. Artwork describes the arrangements for voice/data and simplex/duplex modes, which are selected by two on/off front switches. The 40-pin ribbon connector to the TRS port comes from the front of the Microconnection. The author feels that this is a poor design as the connector cable must be bent at an angle, and causes wear and tear on the ribbon cable.

Upon mentioning the above point in a recent conversation with the Microperipheral Corporation V.P. Don Stoner, I was advised that design changes have located the 40-pin connector port on the side, nicely blending in the cable connect functions. These units should be available soon.

The Micro connection allows computer-to-computer communication over Ma Bell's wires, the transmission of electronic mail in the form of letters and messages, others), stock market data, news, bank services, main-frame computer networks, and also the ability to program in other computer languages. You can also swap programs over the phone, transmit data over Ham radio stations, and make TRS-80-to-TRS-80 communications if the other TRS-80 also has a modem—a heck of a lot of ability for a Level I computer!

With the $249 Microconnection, a dumb (S-80) tape terminal program is supplied, which provides immediate access to a community bulletin board, time-sharing network, and the various networks available, one of which is CompuServe Information Service. Basically the program is dumb, but its simplicity and reliability make it perfect for the new addict to telecommunications. Once loaded under the SYSTEM command, only one question must be answered, which is to select half or full mode (simplex/duplex). When this information is given, a prompt is displayed and you are ready to communicate with the outside world.

For disk owners, the program can be transferred from tape to disk with the Tapedisk command. Use the following Tapedisk data to transfer to disk, assuming you use drive zero (0): F S80/CMD:0 6000 6209 6000. One note of caution, however: the S-80 terminal program will not always work with NEWDOS80, so use TRSDOS 2.2 or 2.3 or NEWDOS+. No tests were conducted with VTOS or other DOSes.

The supplied Manual 1.0 can satisfy most computerists, but certain sections could or should have been written with the beginner in mind. For instance, in the section 'The RS-232 connection' it is mentioned that generally only pins 2 and 7 of the DB25 connector need to be connected to ensure serial interface to a printer. While the use of the printer manual is suggested, carefully check before you make any permanent connections. It would be very desirable to have a detailed circuit diagram of the Microconnection. 'Making the micro connection,' paragraph 2, doesn't apply to many of the telephones sold at various stores. The author's phone, for instance, is barely audible if not connected to Ma Bell's umbilical cord, and no keying tone change is noticed.

Enough said about how to hook the Microconnection to the telephone terminals. Various stores have a selection of plugs for standard (old) terminals, not

so old 4-pin plugs and converters, with which you can emulate your proper phone installation. Just inform Mama Bell that you have connected the Microconnection, ringer value 0.0. No charges can be rendered by Mama, except if you ask her to install a terminal jack should you lack the ability to do it yourself. You will be charged installation cost and a monthly jack fee. If you follow the instructions in the manual, no unpleasant surprises should result, as tone frequencies are permanently locked in.

A few pages in the manual list networks which can be accessed using the Microconnection. At the conclusion of this article, a list of active community bulletin boards and their phone numbers is given. Access is immediate and free—not counting Ma Bell's charge.

Based on the experience the author gained from using the Microconnection for many hours (phone bills are the best proof) only one aggravating problem exists: for all practical purposes, the most unsophisticated part supplied with the unit. The ribbon cable connect should be terminated with more suitable connectors of the more gentle springy type.

The connectors supplied with the author's unit are pressure-type, contact-to-contact devices, which means that every time you have to disconnect, you must yank on the cable itself. The connectors expand when pushed onto the PC board's mating connection, then their plastic memory permits them to return to their natural molded position, i.e. they seem to be welded to the PC board terminators. While this may be permissible under normal circumstances, the author's residence in the Dairy State includes murky, humid weather conditions. Under the slightest humidity, the connection between the Microconnection and the computer will deteriorate, oxide will form, and the reliability of an otherwise perfect connection will suffer. I am assured that the Microperipheral People, who created an otherwise sound product, will not stake their reputation on a troublesome connector cable.

It is the author's opinion that the Microconnection is of great value for the communication-minded computerist. For reasons of quality, simplicity and useability, the unit can be recommended without any hesitation. Constant use by the author over many hours did not produce a single problem with either the TRS-80 or the networks.

It must be remembered, however, that Terminal programs specifically written for a standard TRS-80 setup (RS-232 interface, acoustical modem), will not work with the Microconnection, nor will CompuServe's Loader program (to retrieve the Executive Terminal program), if you selected CompuServe as your data base. While this may not be a consequential matter (many good Terminal programs exist for the Microconnection), it is worth mentioning. For the electronically inclined, Radio Shack uses ports 240-241 for data and status, whereas ports 208 and 209 are used by the Microconnection for the same purpose.

Once you're ready to communicate over the phone systems using your computer and the Microconnection, you can access networks with such services as the latest news, stock market results, banking service and credit card service with instant card verification.

Using the Network

With the data/voice switch in the out position (voice), dial the phone number which has been assigned by your network. You will hear the dial tone, then the connect tone. When this occurs, push the VOICE switch on the Unit into the DATA position, and the carrier-LED will come on. Hang up the phone,—don't worry, you're connected. A few seconds later you will be questioned about your ID number or assigned code. All the information will be displayed on the screen. If a printer is connected, and you used the code for printer activation (as prescribed in your terminal program), all your communications will result in hardcopy. The nice thing is that the Microconnection provides a serial port for a lineprinter, so even the TRS-80 Level I computer can use a printer without any interface or other optional gadgets. Type in your number or whatever information you are asked to supply. Figure 1 shows sample pages from Compuserve.

```
     CompuServe Information Service
On at xx-xx-xx      XX:XX (Date and EDT)
CompuServe            Page 2
1 News, Weather, Sports
2 Finance
3 Entertainment
4 Electronic Mail
5 Compuserve User Information
6 Special Services


9 MicroNET

Enter your selection number
Or HELP for more information:
!
```

Let's assume you selected 9 MicroNET, CompuServe's personal computing component. The following text will appear on your screen after your selection:

```
CompuServe            Page 25
If you proceed to the next page,
you will enter MicroNET. If you
don't have a MicroNET User's
Guide, we suggest you return to
the previous menu (Key: M)
Once in MicroNET, you can return
to the main menu by entering
R DISPLA
from command mode.
The next page will take you into
MicroNET...
Key <ENTER> for next page
!
```

(you just press ENTER)

```
Welcome to MicroNET.
..A new command has been added to make
  MicroQuote access easier
For more information enter: NEWS
For an index of the MicroNET Program
Library (MICRO.NET), enter: R INDEX
To access the CompuServe information
service from MicroNET, enter: R DISPLA
OK
```

From then on you select your choice from the information center residing within the network. Obviously it would be easy to write page upon page of how to, when to, and for what reason to access CompuServe or any other network or bulletin board, but those choices are up to you. As far as the author is concerned, the MicroNet component provides an excellent choice of services. Their rates are excellent.

If you have the Microconnection, the Micro Peripheral People have a new program, The Dow Jones Connection, with which you can access Dow Jones & Company. It allows you to organize your stock portfolio, and update via the Dow Jones computer in Princeton, NJ.

Available Services

Telenet carries only digital telephone line signals but should be explored by the small entrepreneur. General Electric Information Services have tremendous MARK II data bases. The Micro Peripheral People have a software package called 'Mailgram Connection,' which allows you to send telegrams via your computer to U.S. and foreign recipients.

Another useful service is the Telex Connection which uses Databridge, a service of ITT. By using the Microconnection and your computer, you connect to ITT's Telex message center.

Yet another service is provided by the Peripheral People: If you access Compu-

Serve or the Source with the micro connection, you may also access their network-within-a-network. Once you have purchased their product they automatically initiate you into The Microconnection Users Group. It provides the latest information on the Microconnection itself, hardware and software; the Microconnection Owners Bulletin and message center. This is free with the compliments of The Peripheral People. You can leave messages, retrieve messages, and download written programs (record these on your own tape recorder) via the Bell wires. This in itself will justify part of the Microconnection's purchase price.

The latest update on the manual includes terminal programs and other utilities. All in all, the author is impressed with The Peripheral People's interest in supplying a reliable product within the price range of the average computerist, and with their dedication in giving the purchaser additional and free services once he is connected.

The following are addresses of companies and services mentioned in this article:

The Peripheral People
P.O. Box 524
Mercer Island, WA 98040

Computer Information Service
CompuServe Inc.
5000 Arlington Centre Blvd.
Columbus, OH 43220

The Source
Telecomputing Corp. of America
1616 Anderson Rd.
MacLean, VA 20102

Telenet Communication Corp.
8330 Old Courthouse Rd.
Vienna, VA 22180

General Electric Information Services
401 N. Washington St.
Rockville, MD 20850

For listings of data bases contact:
Cuadra Associates
1523 6th St. Suite 12
Santa Monica, CA 90401 □

The 80-Grafix Board

High Resolution Graphics

David Holtz

If you've longed for the graphics capability of Apple, Sorcerer, or Pet microcomputers, then it's time to put high resolution graphics frosting on your TRS-80 Microcomputer cake.

The 80-Grafix board from Micro-Labs, Inc. (902 Pinecrest, Richardson, TX 75080) puts 384 x 192 resolution at your fingertips. You can easily program up to 64 characters while your creative imagination races merrily along. In a 6 x 12 matrix format, you can create whatever characters or symbols you need for your special programming, hobby or business applications. If you need larger symbols, you simply concatenate the characters—or you POKE or PRINT the symbols individually as a concatenated variable, or with conventional program loops.

Maybe you have a use for Greek, French, or Arabic characters. Or playing card or game symbols, space ships and other unique characters would add an exciting new dimension to your programs. Perhaps you have special graphic needs for your business profession. If any of these spark your interest, then you don't have to look any further. You can even have inverse video or lower case!

David D. Holtz, 91 Valley St., Rochester, NY 14612.

And there is software from Programma that will patch the 80-Grafix board-generated lower case directly to Radio Shack's Scripsit word processor. If you've taken a look at the price of hardware-generated lower case with descenders and have planned to take the step, then for a modest additional investment in the Grafix-board, you can link your software-created lower case character set to the Scripsit package and realize high-resolution graphics capabilites besides.

All of this high-resolution capability is wrapped up in a 2 1/4" x 5 1/4", double-sided circuit board that fits snugly in a recessed "well" in the bottom of the TRS-80 case. It requires only two solder connections plus mounting of male DIP connectors over the top of four ICs. You can either push on the DIP connectors or solder the connections to the ICs.

The Package

I'm impressed by the comprehensive 30-page manual that leaves little to your imagination as you methodically step through the installation. There's even a trouble-shooting section if you need it (and you shouldn't if you followed instructions carefully, for the board has been "burned-in" and tested at the factory). An illustration and a labeled sketch show where the board and cables go.

You get a good look at the theory of operation, and then learn how to use the board's capabilities. Seven pages of how-to-use-it instructions are supplemented by application programs and a lower case character set on an accompanying cassette tape. It is helpful to study the program listings in the manual and from the tape to add to your application ideas.

Available Modes

There are three different operating modes, all controlled by the OUT 255 command in your Basic program, but the TRS-80 automatically powers up in conventional graphics mode. OUT 255,32 provides standard TRS-80 characters and graphics. OUT 255,160 puts you into hi-resolution mode, and lets you use the special characters you've programmed into the 80-Grafix board. Standard alphanumeric characters are not affected, but cannot intermix conventional TRS-80 graphics characters and hi-res characters unless you've programmed the standard configuration into your hi-res set. OUT 255,96 is the programming mode for setting up and storing character patterns into Programmable Character RAM (PCR). This mode is used only when programming new character information into the board.

Mode and character programming control, of course, are available using assembly language, too, and Programma has thoughtfully included a section on assembly language techniques. The section includes a complete source listing for a keyboard driver which displays unshifted and shifted (upper and lower case, respectively) characters or whatever character set you happen to have programmed into the 80-Grafix board. The cassette has a SYSTEM program and a disk version which provide a lower case character set with video drivers.

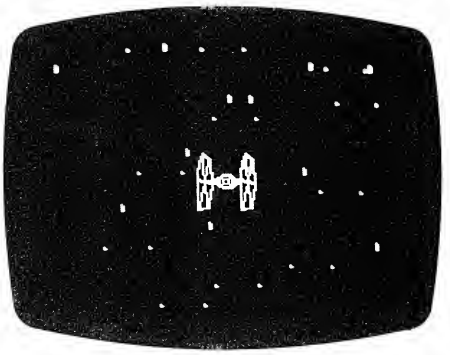
How It Works

Instead of the 5 x 7 dot character matrix that exists in the standard TRS-80 character generator ROM, the 80-Grafix board expands the graphics cells to a 6 x 12 matrix (or resolution of 384 horizontal by 192 vertical). The board, incidentally, has its own 1K (byte) by 6 (bit) memory so you don't have to sacrifice any RAM to obtain hi-res graphics.

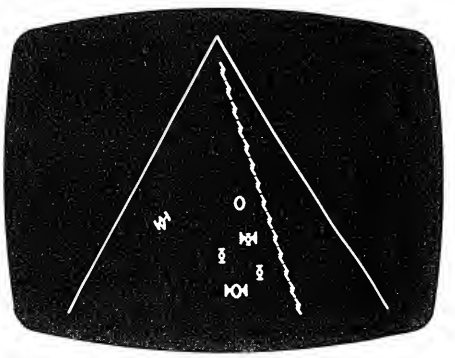
This 1K by 6 memory block of PCR is organized into 64 fields, each consisting of individual character data when information has been programmed into it. Four of the 16 bits in each field are ignored, so in effect you are using the first 12 bytes for the 12 data lines of each character.

The video memory (a separate 1K block of RAM) acts as a buffer when you enter the programming mode. According to the manual, the hi-res character data is moved to the screen, then read as the image of the PCR, and stored into PCR memory. Once data for all hi-res characters is stored, one exits to high resolution mode (OUT 255,160). All of this, of course, is under software control, so it is being done automatically for you.

Characters are transferred to the screen using PRINT, PRINT @, or POKE statements (or with a video driver via the keyboard).



Imagine adding the excitement of high resolution 382 x 192 graphics to your programs. Note the fine detail possible in the spaceship design.



Unique symbols and characters can be quickly created and used via several alternate programming methods you can choose. Or you can use standard TRS-80 graphics with a simple OUT command in Basic.

Although you can't access every single dot on the screen using cursor control for free-form driving, for example, you can access any dot on the video screen depending upon how each of the 6 x 12 elements of the 64 hi-res characters is programmed. So you can do full screen plotting, but access to each dot on the screen is not supported, according to Programma.

Programming Characters

You can program characters into the PCR in several ways. A very simple but somewhat time-consuming way involves designing a character with the aid of a handy 6 x 12 grid form in the back of the manual. A photo-copying machine provides an economical worksheet supply for further programming needs.

First you darken the spaces in the 6 x 12 grid to correspond to the image you wish to create. Then, think of the 6 horizonal blocks in each grid line in terms of binary numbers. That is, a lighted block would be a one; an unlighted block, a zero. With a little practice, one becomes very facile adding the six digit binary numbers, and standard patterns, of course, produce the same sum as you move from character to character. You add each of the 12 lines in the grid block of the character, then simply insert the 12 decimal figures into a DATA line in your Basic program.

As mentioned before, you set the board to hi-resolution via OUT 255,160 in your Basic program. Then a simple FOR-NEXT loop programs the board using READ and POKE statements.

Clear instructions and program syntax are included in the manual and illustrative programs you receive. After data is read, board programming accomplished, and the computer switched to high resolution mode, you are ready to use the special characters in your program.

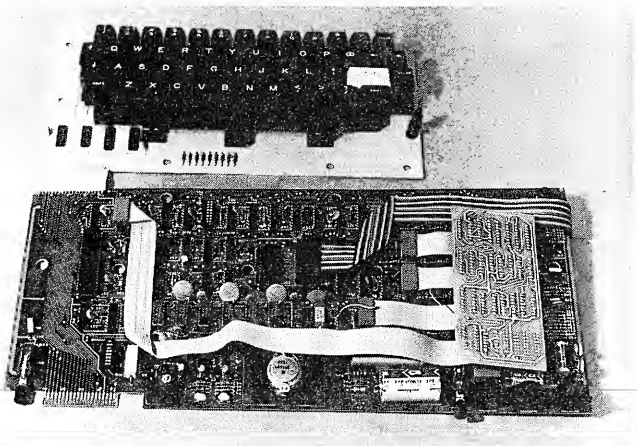
If your computer is disk-based, character data lines can be saved for handy reference and merged at any future time to incorporate the same character data into other programs. Otherwise, the character DATA figures will have to be typed into new programs.

An easier way is to use the Basic language "Create" program, supplied on the accompanying cassette and documented/listed in the manual. By inputting X-Y coordinates for the 6 x 12 cells, the program prints out a decimal figure to enter into your DATA line—much easier. Each of the cell coordinates has to be entered individually, but it is a marked speed improvement over the manual grid sheet method.

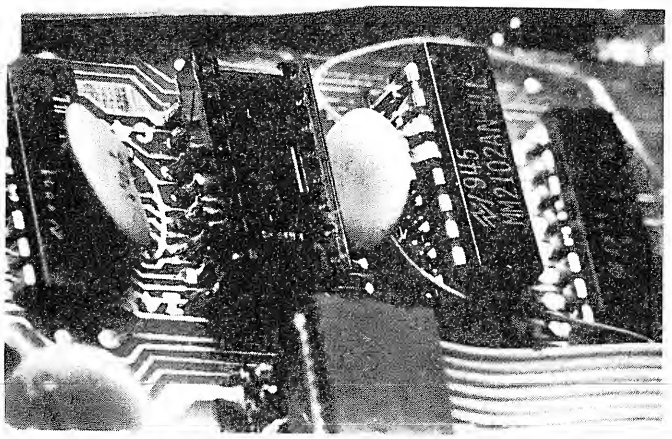
The best way, and I recommend it to anyone who purchases the 80-Grafix board, is to order from Programma International their HIRES80 high resolution graphics editor, a fast machine language program that adds great versatility to your character programming. Its strongest feature is the speed it contributes to your character design. In addition to preprogrammed characters sets (among them an upper/lower case set, inverse character set, fractions, Arabic and Hebrew sets), there is a separate utility program that provides a special video driver for your keyboard. Inverse characters can be obtained by simply pressing a key as you create your character. In my opinion, the HIRES80 graphics editor is the best way to go.

Installing the Board—Decision Points

There are three decisions you will have to make before proceeding. If your warranty is still in effect, you'll have to bite the bullet—decide to break the computer warranty seal and install now, or wait until the warranty ends. In my case, I installed 16K memory chips from Ithaca Audio the same day I bought my computer, so I proceeded full speed ahead.



Four "push-on" male DIP connectors mount on top of existing ICs. Two solder connections are necessary, plus one IC lead to be clipped.



One can solder low-profile IC sockets onto the pins of the four ICs that will receive the DIP-pin ribbon connectors. Special care is necessary when soldering. The Radio Shack lower-case modification (see piggy-backed ICs) is next to one of the ribbon connector points.

Another decision is whether to install the board yourself or have a qualified electronics technician do it for you. If you are not familiar with digital electronics, then your answer is pretty clear cut. You should probably seek assistance.

Another choice is the method of installing the four male DIP pin ribbon connectors on top of appropriate ICs. Programma's manual recommends that connectors be pressed directly onto the top of the ICs, and suggests a drop of glue (or double-faced tape) on top of the ICs for proper adhesion of connectors. The manual says that tests have shown connections such as these to be reliable.

But Programma gives you an alternative. You can mount low profile IC sockets over the top of the ICs and securely solder the socket and IC pins together. It's a bit tricky soldering in tight quarters, but there is real assurance your connection will stay in place if you tote your computer around.

Since the manual is so complete, I'll simply cover special points I observed along the installation route.

The most tedious and painstaking part I encountered was making sure I had good solder connections between the pins of the ICs and the pins of the low-profile sockets mounted above. Quarters are frequently close, so you need a good light, a low-wattage soldering iron, a steady hand, and a bit of time.

I found a magnifying glass handy to check out the solder connections and make sure I didn't have any solder bridges. A handy way to check for bridges between the fine traces interconnecting the computer ICs is to position a good light source (a high intensity light, for example) below the PC board. You should be able to see any bridging outlines from the opposite side of the board.

One snare I ran into was my Radio Shack lower case modification. In my unit, a 2102AN-4L chip is piggy-backed over IC Z46. Two of the pins on this memory chip are bent outward, and Z46 is right next to Z47, one of the ribbon connection points. In order to mount my piggy-back socket, I had to carefully remove the Z46 chip and its top-mounted companion to make room for soldering.

Fortunately the local Radio Shack Computer Center has installed IC sockets so the job was easy. If you've had the modification, you probably will have to bend the two pins downward somewhat, taking care that they don't contact the pins of the 2102AN-4L directly below.

With the lower case modification, I also discovered the new character generator chip to be mounted in a new socket. Under most circumstances that would be great. In this instance, I found the socket-IC combination, potentially another low-profile socket soldered to the character generator chip, and a connector to be inserted in the socket unacceptable.

Among other things (aesthetics primarily), the package would simply be too high for available spacing between the computer board and the keyboard. So, I soldered the connector pins directly to the generator chip. I didn't like the prospect of doing so, but I had no option other than to scrap the character generator chip socket and resolder the chip back into the CPU board. That wasn't acceptable either.

That's about it. There are only two solder connections to make if you choose the nonsocket approach, and you will have to cut the pin of one IC which is certainly no big deal. No traces are cut on the board either, so it is a very clean installation.

This is especially so because the 80-Grafix board sits conveniently out of the way in the bottom of the computer case, and adhesive strips are already attached to the small PC board. Before the board is adhered in the recessed well, make sure you have checked everything out to your satisfaction. You will find the strips hold the board firmly in place, and it takes effort and care to remove the board from the well.

An Evaluation

You should be aware that Programma International doesn't provide a schematic with the board. This will be a major obstacle when servicing becomes necessary at some future time. To make things even worse, chip identification has been removed from some of the ICs. Should your 80-Grafix board ever fail, you will have no choice but to remove the board and ship it back to the manufacturer for servicing.

It seems to this writer, that chip IDs and a schematic should be provided to 80-Grafix board customers. I hope the manufacturer will change its policy, for it is really in the interest of customers to be able either to service their own equipment or have it done locally if the need ever arises.

The manual doesn't tell you how to get inverse video. However, after you've fretted a bit, you suddenly discover that's one of the reasons why the demonstration programs accompany the board. The program listing tells the story, and there are inverse video program data lines waiting right there on the cassette.

I'd like to see an applications book from Programma in the future, for it would save one a good deal of time if a number of special character sets with data information were available. Also, a handy reference card would be super, along with a set of preprogrammed graphics design building-block characters that might accompany the HIRES80 software. The Arabic and Hebrew alphabets that come with HIRES80 are neat for demonstration

but have little utility for many people, so I'd like to see some other more practical character sets such as electronic and logic symbols, graphic building blocks, and so forth.

The 80-Grafix board has all kinds of fascinating potential. So far all of my software works perfectly and has not been affected by the installation. The board has a lot going for it; I recommend it highly. At $169.95, it's a modest investment in a major improvement to the TRS-80. You really can have high resolution graphics icing your computer cake — and eat it, too. □

"Gee, this "Space Traveller" game is intense!"

Add a Joystick to Your TRS-80!

Marc Stanis and David H. Ahl

Are your fingers as uncoordinated as mine when you try to play *Scarfman* or *Robot Attack*? Somehow, I just don't seem to be able to remember which fingers are controlling up and down movement and which are controlling right and left. And then remembering to fire with my thumb; by this time I'm all thumbs.

This article describes how to add a joystick to a TRS-80 Model I or III in an hour or so for the bargain price of around $15. The joystick is completely compatible with all software (Basic and machine language) that uses the four arrow keys to move and the space bar to shoot.

Since the joystick parallels the keyboard, it doesn't require a power supply. It need not be disconnected after use, nor does it contain any electronic parts. If, for some reason the joystick malfunctions, your computer isn't in danger. At worst, you might have to replace the joystick.

Marc Stanis, 14930 S. Springfield Ave., Midlothian, IL 60445.

David H. Ahl is the editor-in-chief and founder of *Creative Computing* magazine.

Before you begin construction, read through this entire procedure and be sure you understand it. You will need the parts listed in the table. You should also be aware that this installation will void the warranty on your computer and that it will have to be removed if the computer ever needs servicing by Radio Shack. Even if the modification has been removed the warranty is still void.

Atari joysticks which connect to the ribbon PC connector at the back of the TRS-80 are available from Big Five and Alpha. These do not void the warranty and work with many, but not all games. If you are leery about voiding your warranty we strongly recommend one of these products even though they are slightly less versatile than the one described in this article.

Examine the Joystick

The first step is to remove the four Phillips head screws from the bottom of the Atari joystick case. Remove the stick mechanism. Be careful not to lose the small spring on the firing button. Examine the printed circuit board. The six connectors on the right side should have wires of the following colors connected to them:

Joystick Position	*Wire Location*	*Color*
Right	Top	Brown
Up	2	White
Common	3	Black
Down	4	Blue
Left	5	Green
Button	Bottom	Orange

If the wire colors do not correspond to this list, change the connectors so they do. Reassemble the joystick taking care that the firing button spring is back in place and that no wires are pinched—either between the edges of the case or between the edge of the case and the PC board.

Cut the DE-9 connector off the end of

the Atari joystick cable. Starting from that end of the cable, *carefully* remove 10 to 12 inches of the outer plastic cable sheath with a sharp knife or electrical scissors. Take it slowly and be careful not to damage the six fine wires.

After removing the cable sheath, examine each of the six wires over its entire length to make sure it has not been damaged. If one or more have been damaged, you will have to replace them with fine (24 gauge) flexible wire or remove more of the cable sheath to get 10 to 12 inches of intact wire.

Next, strip off the insulation from the end 1/8" of each wire. Tin the ends of each wire, i.e., with a hot pencil soldering iron, apply a small amount of solder to the tip of each wire. This will make them easier to solder to the PC board in the next step. Apply heat and solder quickly so you do not melt any of the insulation. Inspect your work and set the joystick aside.

Into the Computer

Next you are going to open the computer. Be sure you are working in a static-free location, no rugs, cats, etc. Ground yourself out before starting work by grabbing a water pipe, grounded case of a three-wire power tool, etc.

Remove the screws from the bottom of your computer and remember which screw came from which hole. Label them if necessary. (**Note: Opening your computer voids the warranty.**) If you have a Model I, lift off the bottom of the case and set it aside. *Carefully* flop the PC board in the direction of the wire connector. This exposes the bottom of the printed circuit board of the keyboard.

If you have a Model III, after removing all the screws, turn the computer right side up. Lift the top off and tip it onto the left side leaving all the wires connected. Remove the six screws holding the keyboard cover and set the cover aside. Tip the keyboard forward.

Look at the etching on this PC board. Near each pair of connections (representing each key) is a small silver etching of the name of the key (Model I) or a number (Model III). On the Model I, the second row down on the left, you should find the right and left arrow keys while the rightmost keys in the second and third rows are the up and down arrow keys. Of course, the spacebar is at the bottom. On the Model III everything will be reversed since the keyboard is effectively upside down.

Figures 1 and 2 show the three portions of the etchings on the PC board around the keys of interest. Match this with your keyboard. Solder the colored wires to the connections indicated on Figures 1 or 2.

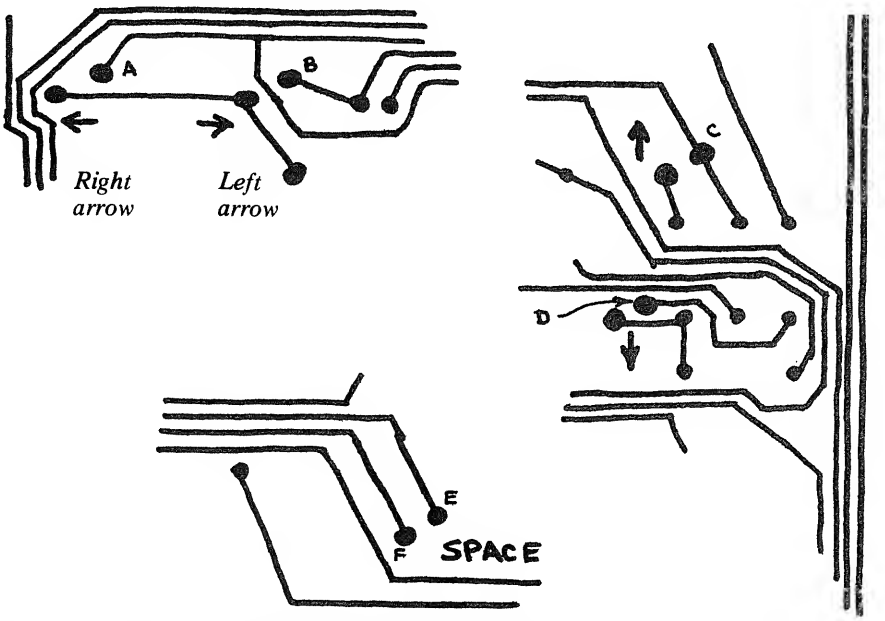


Figure 1. Portions of the TRS-80 Model I keyboard by the arrow keys and spacebar.

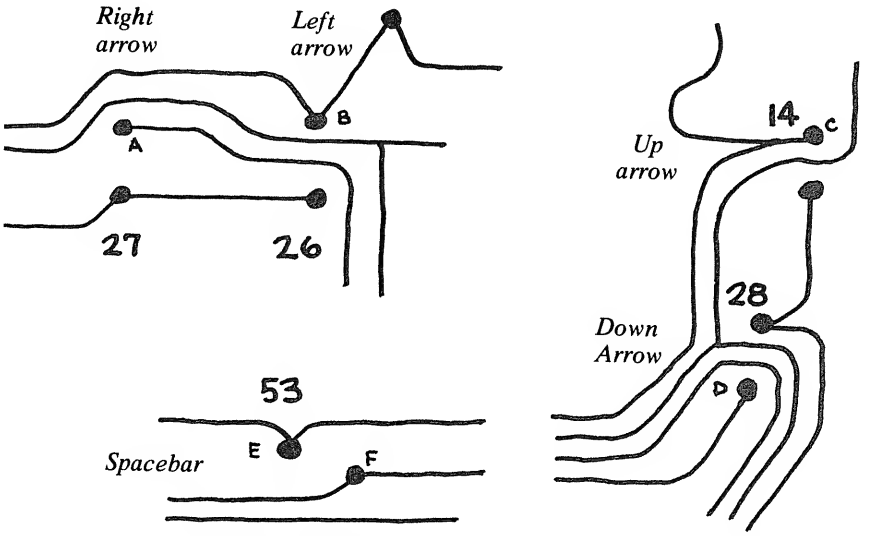


Figure 2. Portions of the TRS-80 Model III keyboard by the arrow keys and spacebar.

Make connections as indicated:

A	*Brown*	*D*	*Blue*
B	*Green*	*E*	*Orange*
C	*White*	*F*	*Black*

Solder these connections by touching the tip of the soldering pencil to the solder blob until it starts to melt (two seconds or so), inserting the tinned end of the wire and quickly removing the soldering pencil. Blow on the connection immediately. Following this procedure should insure that none of the wire insulation melts. If some insulation melts and the exposed wire could possibly touch another connection, unsolder it, cut off the end, tin it, and try again.

Decide where you want the joystick cable to come out of your computer. Model I owners can snake it out through the rectangular ribbon connector hole on the back left of the case although it is difficult to put a strain relief on the cable at this point. A better choice might be the front or either side of the case which choices are also open to Model III owners. With a sharp knife or rat-tail file, cut a small 1/8" notch in the case of the computer.

Back Together Again

Tie a knot in the cable close to the end of the remaining sheathing. Alternatively, you may put a small cable clamp on at this point. Snake the wire to the notch and, with the knot or cable clamp on the inside, reassemble the computer case. When doing this, be sure all the spacers between the PC boards are where they belong, check that nothing is pinched and use the correct screws.

Now comes the moment of truth. Turn on the computer (video display first, if a Model I). If you don't see the normal startup message, immediately turn off the computer; something is wrong. The problem is probably that something is pinched since even gross mistakes in soldering or a shorted connection on the keyboard will not cause a non-startup condition. Re-check everything (keyboard connections, damaged insulation, stray solder blobs, pinched wires, etc.) and reassemble.

After a proper start, try the joystick. Pressing the fire button should print spaces. The up position should print up arrows (Model I) or left brackets (Model III). The left position should print backspaces. The right position should print tabs while the down position should produce a carriage return/line feed. If you get any but these results, open the case and check the connection(s) to the keys which are not responding correctly. Reassemble and try again.

The cable on the Atari joystick is about 40" long; we have used about 12" of it inside the computer which leaves a 28-inch cable. If you feel that this is long enough (it should be for most purposes) and you don't mind having the joystick permanently connected to the computer, then just stop here. You are finished.

On the other hand, if you want a removable joystick or a longer cable, you will have to cut the cable and install male and female connectors and, if you wish, a longer cable. You can choose any one of many in-line connector sets (Cinch-Jones, D.I.N., etc.). Flexible six-conductor cable isn't so readily available and you might have to settle for ribbon cable. Some electronics outlets carry coiled six-conductor microphone cables (for CB and ham replacement use) which is perfect.

Obviously the rule to follow when installing an in-line connector set or longer cable is to maintain the integrity of each wire. When you are done, pushing the joystick left should still act as a left arrow keystroke, the fire button should be a spacebar stroke, and so on.

All done? Load in *Scarfman* or *Robot Attack* or your favorite shoot-'em-up game and you will be amazed. Never thought you could score that high, eh? □

TRS-80 Plug 'n Power Controller

Polly Put the Kettle On

Richard A. Zatarga

I have owned a BSR X-10 control system for several years and have been intrigued with the possibility of somehow controlling the modules with my TRS-80 Model I computer. Recently, several hardware interfaces have been introduced to perform this function. These interfaces range in price from less then $40 to over $200.

The cheaper units usually provide a sonic link from the computer to the BSR Command Console. The more expensive units operate without the console, i.e., they send the signal directly from the interface, through the house wiring, and on to the control modules. Some controllers come complete with the necessary software for them to function. Others offer software as an optional extra.

Richard A. Zatarga, 800 Towner Swamp Rd., Guilford, CT 06437.

In my search for the right computer-to-X-10 controller, I made a list of attributes I felt the unit should possess. It had to be relatively inexpensive. It had to be compact. (I don't have too much room left on my desk.) I didn't want a sonic link type controller. My command console is centrally located in my house, and quite some distance from the computer. I had no desire to move the console every time I wanted the computer to control the modules.

It also had to be permanently installed without affecting the operation of other peripherals; I didn't want to remove and attach cables to use the controller. Lastly, it had to be easy to use and program so other members of the family could readily operate it.

My quest came to an end when I received a TRS-80 Plug 'n Power Controller. The price is only $39.95. It is only 4.5" x 4.25" x 1.5" in size, and the command console is not required—no sonic link. It attaches to the cassette DIN jack on the TRS-80 and the cassette recorder plugs into the controller.

A single rocker switch on top of the unit selects either cassette or controller operation. Finally, the Plug 'n Power Controller came complete with the software to satisfy all computer configurations for which the unit was designed—TRS-80 Models I and II, and the Color computer.

Documentation

The TRS-80 Plug 'n Power Controller comes with two cassette tapes and an instruction booklet. The 36-page instruction booklet exceeds Radio Shack's high standards for documentation. It is clear, concise, and complete concerning the installation,

operation, and programming of the controller.

Set-Up

It only takes five minutes to attach the controller to the computer permanently. You simply unplug the cassette cable from the computer and plug in the cable from the controller. Plug the cassette cable into the DIN jack on the controller and the line cord of the controller into an AC outlet.

In order for the computer to activate the X-10 modules through the Plug 'n Power Controller, a software link or operating program must be loaded into memory. This is a machine language program for the Level II machines and a Basic program for the Color and Level I computers. To load a Level II version of the operating program, you set the rocker switch on the controller to CASSette and enter the following:

SYSTEM ENTER
*? L ENTER

Set the rocker switch to conTRoL and enter a slash (/) after the second question mark (?) to execute the program.

Table 1.

Line Number	Time	House Code	Unit Code	Command
01	07:00	A	03	OFF
02	07:00	B	14	ON
03	07:25	B	14	OFF
04	19:30	A	03	ON
05	07:00	A	03	OFF
06	09:30	B	14	ON
07	09:55	B	14	OFF
08	19:30	A	03	ON

A scenario for the above commands might be:

Day 1
01 7:00 a.m.—Turn off outside security light.
02 7:00 a.m.—Turn on coffee pot.
03 7:25 a.m.—Turn off coffee pot.
04 7:30 p.m.—Turn on outside security light.

Day 2
05 7:00 a.m.—Turn off outside security light.
06 9:30 a.m.—Turn on coffee pot. (Must be Sunday)
07 9:55 a.m.—Turn off coffee pot.
08 7:30 p.m.—Turn on outside security light.

Operation

The controller may be operated in either a direct command or a programming mode. I really can't see any purpose for the direct command mode. It is faster and easier to turn a single appliance or lamp on and off directly than it is to power up the computer, load the operating program, enter the time, and finally, issue a command to a module via the computer keyboard.

In the programming mode, however, the Plug 'n Power Controller is a very useful tool. In this mode you can program a sequence of commands to turn lights and applicances on and off at different times over a period of hours, days or even weeks. Your only limitation is a maximum program length of 45 lines, i.e., 45 separate commands.

Once a program sequence is entered and executed, it will continue to function even after the last line of the program has executed. The sequence of commands will repeat again and again, starting with the first program line. For example, if you program a three-day sequence of commands and are away for five days, the first day's command sequence will begin to repeat on the fourth day. In fact, the whole three-day sequence will repeat indefinitely, unless there is a loss of power and the operating program is lost.

Both operating modes use the following six control commands:

ON: Turns on the selected module.
OFF: Turns off the selected module.

CLR: Turns off *all* modules for a selected house code.

ALL: Turns on *all* modules for a selected house code.

The following two commands only work with the lamp and wall control modules. They will not work with the appliance modules.

DIM1-DIM9: Decreases or dims the level of light intensity for a selected module. The number after the command determines the level of intensity.

BR1-BR9: Increases or brightens the level of intensity for a selected module.

Programming mode commands allow you to (I)nsert, (R)eplace, (E)dit, (D)elete and (L)ist program lines. One command, controlling one module. is allowed per program line. However, up to eighteen commands can be executed at the same time by using the same execution time in two or more program lines.

A program line must contain the following five items of information:

- A line or sequence number.
- The time the command will execute.
- The house code of the module being commanded. (A thru P)
- The unit code of the module being commanded. (1 thru 16)
- The command (on, off, etc.)

Line numbers are automatically displayed by the operating program and do not have to be entered. The other items are keyed into the program line individually, with each item followed by a carriage return (Enter). After the four items have been keyed into a program line, the next line number is displayed, ready for a new command. Table 1 shows an example of a two day sequence of commands to give you an idea of the format.

Conclusion

I have nothing but praise for the unit. It has performed flawlessly since the day I hooked it up. Its only limitation is the 45 program line maximum of the operating program. This limit precludes a multi-week program of commands that contains any degree of time and sequence sophistication. However, I considered this to be a minor flaw in the design of the controller. Since it is the software that is at fault, a little programming and experimentation should overcome the flaw completely.

So, if you want to foil burglars by giving your home that lived-in appearance while you are away, or just want the luxury of having your coffee ready when you get out of bed in the morning, you can't go wrong with this cost effective remote control computer interface from Radio Shack. □

A Dot Matrix Printer For the TRS-80 Level II

Radio Shack Line Printer VI

Roger and Bonnie Koester

After a long period of comparison shopping for a printer to use with our TRS-80 Model I/ Level II, we decided on a Line Printer VI. The first week of September we placed our order, even though the printer wouldn't be ready for shipment until September 30th. On October 21st the Radio Shack dealer phoned saying our printer had arrived.

The unpacking process was quick—partially because of the method of packing, and partially because we were eager to set-up and run the unit.

The printer was securely packed with styrofoam blocks holding it tightly in place. There was a dust cover over the Line Printer, and printer and dust cover were both inside a plastic bag. The manual, which is extremely informative, was packed inside with the Line Printer. The ribbon cable, which must be ordered separately, came in a second package. There are two ribbon cables available—one plugs into the keyboard,—the other plugs into the expansion interface.

Once unpacking was complete, it did not take long before a new sound was coming from our computer room.

The Line Printer VI, a 9 X 7 dot-matrix impact printer, has a very efficient output rate.

1.) It prints NORMAL characters at 100 characters per second, and 33 lines per minute.

2.) The CONDENSED characters are printed at 120 characters per second, and 37 lines per minute.

3.) There is a maximum of 132 characters per line.

4.) It has the capabilities to print in both directions. In the large (elongated) characters, however, it only prints left to right.

The printer has four sizes of print that are software selectable, and simple to use.

1.) Normal Characters

FIGURE #1

```
LPRINT"SMALL";CHR$(31);"LARGE";CHR$(30);"SMALL AGAIN"
PRINTING OF NORMAL AND DOUBLE-SIZE CHARACTERS ON SAME LINE DEMONSTRATED.

LPRINTCHR$(27);CHR$(14);"CONDENSED"
PRINTS CONDENSED CHARACTERS.

LPRINTCHR$(27);CHR$(15); "NORMAL CHARACTER"
CONCELS CONDENSED CHARACTER MODE AND RETURNS TO NORMAL CHARACTER.
```

FIGURE #2

```
LPRINTCHR$(27);CHR$(54)
                    PRINTS LINES AT 6 LINES PER INCH.
                  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

LPRINT CHR$(27);CHR$(56)
                  PRINTS LINES AT 8 LINES PER INCH.
                  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

LPRINT CHR$(27);CHR$(54)
                  PRINTS LINES AT 12 LINES PER INCH.
                  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
                  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXxx
```

FIGURE #3

```
 !"#$%&' ()*+,-./0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]↑←
abcdefghijklmnopqrstuvwxyz{|}~
```

2.) Elongated Characters
3.) Condensed Characters
4.) Elongated—Condensed

To change print size "CHR$( )" command is needed. (See Figure 1.)

The vertical spacing, also software selectable, can be six, eight, or twelve lines per inch. It automatically goes to six lines per inch at power-up. To change vertical line spacing, a two part code is needed. (See Figure 2.)

Roger and Bonnie Koester, R.R. #1, Box 161A, Tennyson, IN 47637.

One of the best features is the choice of tractor feed or friction feed. With tractor feed you can use any standard tractor feed paper ranging from mailing label strips up to 15 inch forms. They can be single, double, or triple sheets, allowing an original and two copies. We'd like to mention that although the specifications say 4 to 15 inch paper, the tracks will go from 2 5/8 to 16 5/8 inches. The tractor feed paper can be entered at two locations. There is a paper insertion opening at the top center. If your printer stand allows, paper can also be fed into the bottom of the printer.

Changing from tractor feed to friction feed is simple. You turn off the power, remove the upper plastic cover, and pull up on both sides of the mechanism. The rear part will unlock and you'll be able to remove it by simply continuing to pull upward. Then you re-install the plastic cover and turn on the power. You're now ready for friction feed.

The friction feed is a real plus. It has all the same capabilities as the tractor feed, and more. The use of single sheets of paper is very beneficial. We've used typing paper to write letters. They look great.

It is very easy to keep track of the line you are on when you need to advance the paper. Mechanically, you use the Line Feed button, which advances by very definite spacing. Manually, the roller can be quietly heard, and distinctly felt, as it advances each line. Many printers simply roll freely in order to advance the paper, and you cannot tell how many actual lines you have moved.

The streamlined, attractive appearance is a plus in our computer room. The Line Printer VI measures 24.2 inches(W) X 6.3 inches(H) X 13.3 inches(D). On the front there are three buttons (left to right reset, line feed, 1/12 line feed), and one switch (on/off). Just above the buttons are three L.E.D.'s (power, alert (out of paper), ready).

There is a self-test switch on the left rear corner. During testing the line printer does not need to be attached to the computer. The test prints the character set (See Figure 3.) Notice, as mentioned before, the graphics capabilities.

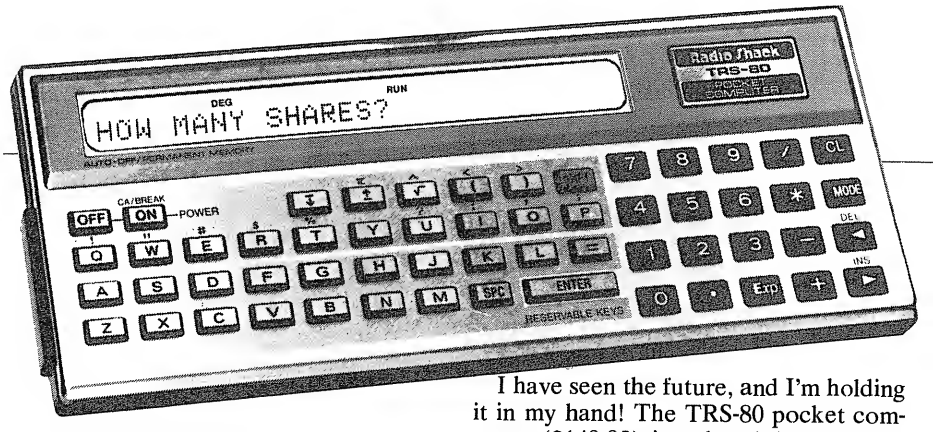
So far we've only touched on the good features. There are a few things with which we are not satisfied:

1. The high-pitched noise which one can hear any time the printer is on.
2. The lack of a method for setting tabs.
3. The lower case g, y, p, and q do not go below the line. It's a shame that a printer of this quality does not do this.
4. The absence of markings by the paper release lever denoting which position is LOCKED for friction feed, and which position is FREE for using tractor feed.

Regardless of these four points of dissatisfaction, we still believe the Line Printer VI is the best line printer we've seen for the money. □

A New Generation of Computing Accessibility

The TRS-80 Pocket Computer



Glenn Hart

I have seen the future, and I'm holding it in my hand! The TRS-80 pocket computer ($149.95) is a breakthrough, the vanguard of a new generation of computing accessibility.

The TRS-80 PC appears at first glance to be either an unusually designed calculator or some sort of language translator. In fact, it is neither, being far more sophisticated than either. It is a complete *computer,* with keyboard, 24 character alphanumeric output display, Basic and monitor in ROM, 1.9K of non-volatile RAM, and an I/O port for storing programs and data on cassettes. Its capabilities would have been reasonably impressive in a first generation desk-top

Glenn A. Hart, 61 Church Rd., Monsey, N.Y. 10952.

small computer (and are far in advance of the early *room-size* computers); in a package small enough to fit in a pocket they are astounding!

The Hardware

Measuring only 6⅞" x 2¾" x 19/32" and weighing a bit less than six ounces, the TRS-80 PC is sleek and attractive. The 57-key alphanumeric keyboard is laid out in a modified typewriter arrangement; the letter keys follow standard "QWERTY" practice while the numeric keys appear on the right side in calculator array. There are also several special purpose keys to handle editing and the generation of symbols, and 13 keys have alternate outputs selected by a SHIFT key. While the keys look rather small and close together and there is no tactile or audio feedback when a key has been depressed, keyboard action is very good and with only a little experience it becomes quite easy to enter text. It is even possible to "type" with both hands for quicker entry, although the non-standard location of the numbers and the need to use the SHIFT key to enter simple punctuation takes a bit of getting used to. Eighteen keys are "reservable," meaning that special program segments can be pre-programmed into a dedicated memory area and called up quickly.

The liquid crystal display is highly legible in normal room light, but, like any other LCD display, it should be viewed more or less directly from above. A 5X7 dot matrix is used to generate upper case only letters and the various punctuation marks and special symbols. No graphics symbols are provided, but they would not be particularly appropriate on a one line display anyway. The input buffer is 80 characters, and if a line longer than the 24 characters displayed is entered the display uses horizontal scrolling in either direction. Cursor control is provided to move around the buffer area for editing and viewing long lines.

A nine pin I/O connector on the left edge of the PC is used to connect the optional cassette interface ($30). Normally protected by a plastic cover, this connector could presumably be used to attach other peripherals even though Radio Shack has not announced the availability of anything other than the cassette interface. The cassette interface itself is a plastic cradle into which the PC is inserted. Various slots and protrusions on both the PC and the interface prevent incorrect attachment. Three cables are provided to connect the interface to the microphone, earphone and remote jacks of any cassette recorder. The new Radio Shack Minisette-8 is an excellent match in size and styling; together they form a complete system which fits in one corner of an attache case.

Radio Shack does not provide any information on the microprocessor or support circuitry other than to indicate that large-scale-integration CMOS devices are used. Rumor has it that TWO *four-bit* microprocessors are used: one to handle Basic instructions and one to perform calculations. At a time when some of the biggest excitement in microcomputing centers on the availability of sixteen bit CPU's (with 32-bit devices in the wings), the TRS-80 PC is graphic proof that important and interesting things can be done with many fewer bits on hand.

CMOS technology offers several advantages, the most significant of which is extremely low power consumption. The TRS-80 PC draws only *one-hundredth of a watt,* and can operate for up to three hundred hours on four small mercury batteries. The cassette interface requires three AA cells and the cassette recorder four.

Most importantly, the computer's memory is *non-volatile,* which means that it is not cleared when the power switch is turned off. Programs and data are retained in memory continuously and are immediately available for use when the computer is turned on. While this continuous memory has begun to appear on advanced pocket calculators, the TRS-80 PC is, to my knowledge, the first true personal computer (other than those with adapted core memory) with this convenient and useful feature.

The 1.9K of RAM available is divided into several segments. Basic program storage is limited to a maximum of 1424 "steps," with each step corresponding to a character in a program line. Note, however, that Basic commands are compressed to one byte tokens, so the *effective* program length is more than 1424 bytes. Twenty-six "fixed" memories are provided. Each can store the contents of one numeric or string variable, with the length of a string variable or constant limited to seven characters. If more variable store is required, up to 178 "flexible memories" can be allocated at the expense of program storage. In addition, there are 48 steps of "reserve memory." These hold up to 18 short program segments which can be retrieved easily for use in manual calculations of programming. The remainder of memory is allocated to the 80 character input buffer, an 8 step "data stack" and a 16 step "function stack." There is no apparent way to expand memory beyond that incorporated into the computer, but a surprising amount of useful programming can be done within the limited memory available.

The Basic Interpreter

The use of a high level language like Basic is what sets the TRS-80 PC apart from the many programmable calculators available today. As the owner of both a Texas instruments TI-59 and a Hewlett Packard HP-41C, I have enjoyed these devices and programmed them to perform many useful functions, but the pseudo assembly language necessary to program them is often a source of frustration at best and a total obstacle to certain tasks at worst. Radio Shack has asserted that the PC renders programmable calculators obsolete, and I would have to agree that machines that seemed advanced until now have suddenly become much less impressive.

The TRS-80 PC Basic interpreter is *not* a direct equivalent to any of the Microsoft-supplied interpreters used on full size Radio Shack computers. I am not particularly familiar with Level I or Level II Basic, but it seems that the PC interpreter has capabilities similar to those of Level I with a few bells and whistles added and some modifications made to handle the specific demands of a computer displaying only one line at a time.

TRS-80 PC Basic allows only 26 directly named variables, corresponding with the 26 "fixed" memory locations. Each location can contain *either* numeric or string information (strings are limited to seven characters) but not both. Thus variables A and A$ occupy the same memory, and only one or the other can exist at any time. Only one array, labeled A(), can be used; its elements point to the same memory locations as a direct reference would — at least up to A(26), which is the same thing as memory location Z. Up to 178 additional locations can be allocated to variable storage (at the expense of program storage). These locations can only be accessed as elements of the A array (e.g., A(145)).

The statements and commands provided are listed in Table I. LET works normally to assign values to variables and is, as usual, optional in most cases. (It is necessary only after an IF statement.) INPUT allows the entry of data from the keyboard, with prompting messages if desired. Prompts do *not* scroll horizontally, so it is necessary to limit the prompt message to what will fit on the display.

PRINT outputs messages and data to the display, as in most Basics. Due to the one line display, PRINT interrupts program execution until the ENTER key is depressed, similar to the interruption caused by an INPUT statement. The PAUSE statement performs the same functions as PRINT except that the information displayed appears for only a little less than a second (the display time could be a bit longer for maximum utility).

I was surprised to find that a limited form of PRINT USING is included (the USING formats can also be used with PAUSE). Only numeric formats can be specified. The normal "###.##"-type format sets the display mode; a carat sign sets scientific notation display. No provision is made for special formatting characters like dollar signs, asterisks, floating signs, etc. The format can be specified for all output with a USING statement by itself or the standard PRINT USING constructing can be used.

GOTO and GOSUB operate normally, except that alphabetic labels can be inserted in lines and GOTO and GOSUB can be instructed to branch to these labels as well as to simple line numbers. This is a nice feature rarely found in *any* Basic interpreters.

The IF statement does not require a following THEN (THEN is synonymous with GOTO), but therefore requires a LET if a variable assignment is to be made. FOR/NEXT loops operate normally, with a maximum nesting level of four. The STEP statement is available, but only with integer values.

BEEP sounds a small tone as many times as its parameter indicates. CLEAR clears all data memory. DEGREE/RADIAN/GRAD sets the mode for angular entries and calculations. AREAD reads a value into a variable prior to the start of execution of a defined program. REM allows comments to be inserted into a program (although heavy use of comments is unlikely with limited memory).

Commands usable only by manual entry include RUN to execute programs, CONT to continue interrupted programs, LIST to list program lines, NEW to clear memory, and MEM to display the remaining available program and flexible memories. The DEBUG command is similar to Microsoft's TRACE; the line number of each program line is displayed after it is executed.

The commands for cassette storage of programs and data follow closely normal Radio Shack Basic practice. The CSAVE and CLOAD commands perform the obvious functions, while CLOAD? verifies accurate saving or loading by comparing a cassette stored program with the contents of memory. PRINT# and INPUT# save data rather than programs. CHAIN loads new programs from tape and immediately executes them, either at the beginning of the new program or at a specified label within the program; this allows programs longer than memory to be segmented into smaller programs and executed sequentially, greatly increasing the power of the computer.

Table II lists the functions provided. A complete spectrum of trigonometric functions is available as well as several normal Basic functions. No string functions or logical operators are provided, although logical operations can be simulated with programming tricks explained in the user's manual.

Statements					
LET	INPUT	PRINT	PAUSE	USING	GOTO
IF	THEN	GOSUB	RETURN	FOR	TO
STEP	NEXT	STOP	END	BEEP	CLEAR
DEGREE	RADIAN	GRAD	AREAD	REM	
Command Statements					
RUN	DEBUG	CONT	LIST	NEW	MEM
Cassette Tape Control Statements					
CSAVE	CLOAD	CLOAD?	CHAIN	PRINT#	INPUT#

Table I. TRS-80 Pocket Computer Basic Interpreter Statements and Commands

Editing of program lines is possible with the cursor movement keys located in the lower right of the keyboard and the line movement keys above the center of the alphabetic area. The cursor can be moved to any spot in a line and the contents of that spot overwritten with new information. Alternatively, characters can be deleted or inserted at will. The line movement keys move the display within a program so any line can be accessed readily. As in other Basics, entire lines are deleted by typing their line number and ENTER and new lines can be added between existing line numbers. No renumbering facility is provided.

Operation and Evaluation

The TRS-80 PC operates in four modes, indicated by small status displays above the main 24 character display. The RUN mode is used both for manual calculations and to execute programs. If several programs are stored in memory at the same time and each is defined with an alphabetic label, the DEFINE mode is used to run the separate programs. PROGRAM mode is used for entry and correction of Basic programs. Most of the Basic statements and commands have abbreviations which make entry programs quicker and more convenient. RESERVE mode is used to enter short programs and/or statements into the 48 step reserve memory. Two small templates are provided which fit over the reservable keys; these can be used to indicate what program or statement has been assigned to the individual key.

Radio Shack does not supply any information on clock speed. In general, Basic programs execute rather slowly, although they seem to run at least as fast as equivalent programs coded for my programmable calculators. The TRS-80 PC maintains up to 23 digit precision internally (although displayed results are limited to 10 digits before the computer switches to scientific notation), so some premium in execution speed is certainly justified by this very high precision.

In practical use, the TRS-80 Pocket Computer is a delight. Manual calculations are a pleasure, since so much of the input is retained on the display and can be corrected if necessary. A user who constantly requires trig functions might find a calculator faster than having to type in the function names, since the necessary function keys are immediately available.

Programming is vastly improved over programmable calculators. Any reasonably competent Basic programmer can put the PC through its paces and generate complex programs *much* faster. The interpreter is surprisingly capable and easy

program; the debugging and editing

tions make program correction simple. The full alphanumeric 24 character display is a pleasure (the alpha display of the HP-41C seemed advanced only a few months ago; now it is impossibly restricting).

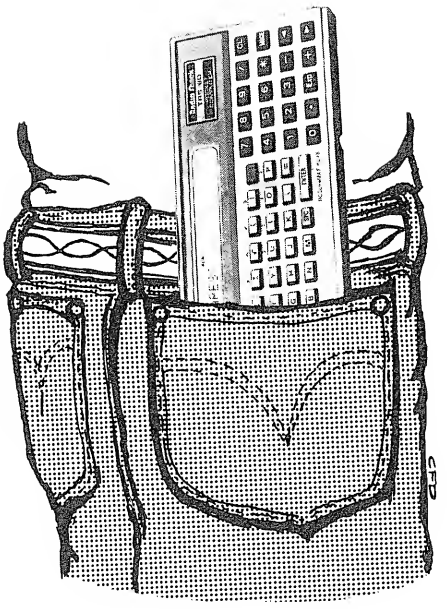
I have already used the PC in several

SIN	COS	TAN	ASN	ACS	ATN
LN	LOG	EXP	SQRT	DMS	DEG
INT	ABS	SGN			

Table II. TRS-80 Pocket Computer Basic Interpreter Functions

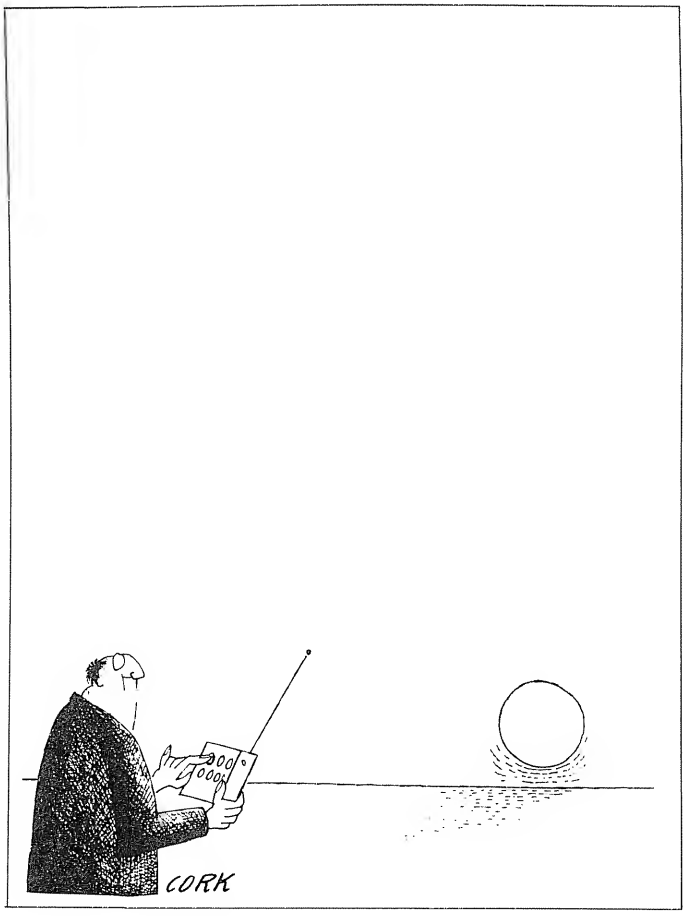
business meetings where I was able to use pre-stored programs to good advantage and even write and execute new programs on the spot to answer complex business problems. Its small size has earned it a permanent spot in my attache case; I am not willing to be without it wherever I travel.

This is not to say that the machine is perfect. Several other manufacturers have announced similar products which would appear to have more capabilities, either more memory or available printers, modems, etc. I am sure that the TRS-80 PC will be surpassed in the future unless Radio Shack makes a major commitment to expanding this end of their business. The presence of that I/O connector would indicate that such expansion is possible. None of the software packages designed for the PC were available at the time this review was written; a large software library will be necessary to wean many



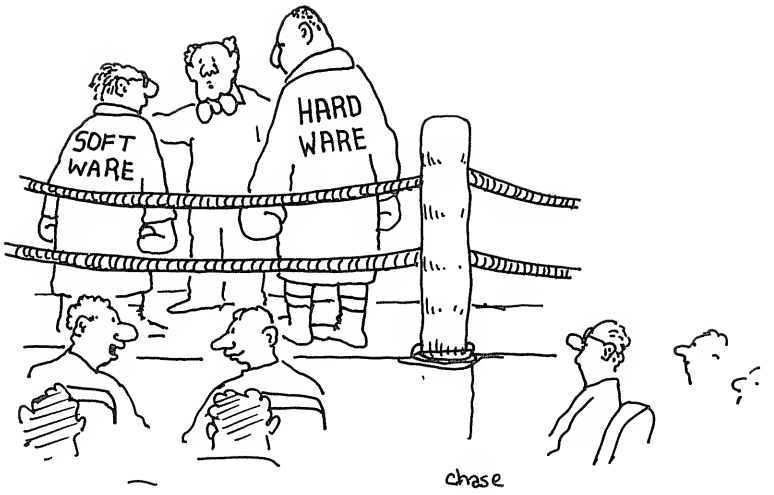
prospective purchasers off their programmable calculators and their large software base and user groups. I guess that we computerists are by nature greedy gluttons; we get a wonderful feast of new technology and all we want is more!

In the meantime, the TRS-80 Pocket Computer is the state-of-the-art. It is not denigrating the usefulness of this computer to also point out that it is a terrific toy and an absolute conversation stopper. The day after I bought mine I had two mainframe-computer specialists, each controlling several million dollars of hardware, in my office to discuss development of a complete business control system. The looks on their faces when I demonstrated the TRS-80 PC more than justified its purchase price. Their chagrin when I pointed out to them that it was the lizards who survived, not the dinosaurs, made my day! □



Chapter II
Software

"This ought to settle a few arguments."



Dumpload

Inexpensive Backup for TRS-80 Disks

David A. Hinton

SOFTWARE PROFILE

Name: Dumpload

Type: Disk-to-tape backup utility

System: Model I TRS-80, Disk drive

Format: Disk or Tape

Summary: Valuable tool for backing up disk libraries.

Price: $16.95 on tape, $19.95 on disk

Manufacturer:
Complete Computer Services
8188 Heather Dr.
Newburgh, IN 47630

Many utility programs have been written and sold for the Model I TRS-80. Most of these are well-thought-out pieces of software that fill the programmer's needs, and a few of them can even be classified as excellent. Dumpload, created for users of disk-based Model I systems, is one of the newest entries into the utility software marketplace and it, too, deserves to be called "excellent."

Any experienced programmer knows the importance of making backups of the frequently used and valuable disks in his library. Some people, myself included, don't feel safe unless they have backups of their entire library. As the program library grows, having a duplicate set of disks soon becomes a very expensive practice. Some programmers resort to using less costly cassettes to make backup copies of seldom used programs, but this is usually a tedious process and does not work well for all types of software.

David A. Hinton, R.R.3, Box 44B, Rockport, IN 47635.

Getting Started

Dumpload allows you to make cassette backups of your disk library—but without the usual hassle. It can copy anything and everything (e.g., DOS, data, word processor files, Basic, Fortran, Pascal, assembly code, object code, etc.). The command options allow the user to copy only a certain track, a group of selected tracks or the entire floppy. When making a complete disk backup to cassette, the process is fully automatic even for one-drive users. No more swapping disks in and out of the drive. Just load the desired floppy, load a blank tape, initialize Dumpload and walk away.

This utility can be purchased on cassette or disk. I ordered the cassette version and received it in about 10 days. The instructions which accompany Dumpload cover the use of both the tape and disk versions. Procedures are included to place the tape version on a disk for easier access or copy the disk version to another disk. Dumpload will work with TRSDOS 2.3 or NEWDOS80 without modification.

A Choice of Speeds

When Dumpload is loaded, it begins by asking if you want to use the standard 500 baud tape speed. What's this, a choice? That's right, the program is capable of backing up a disk to tape at the standard Radio Shack cassette speed of 500 baud or at an optional high speed of about 1800 baud. At 1800 baud, a 40-track disk can be saved on less than 10 minutes of tape.

The written instructions point out that you will have to run at 500 baud if your keyboard contains the Radio Shack XRS-2 cassette modification. If you have this modification, indicated by a keyboard serial number ending with a dash one (-1), don't give up on using the high speed. Another set of instructions included in the Dumpload package gives all the information needed to install a bypass switch which will allow you to enable and disable the XRS-2 modification at will. (This is the same modification required to use B17 sold by ABS Suppliers).

If you prefer not to mount a switch in your keyboard case or you don't have a switch immediately available, the instructions also describe how to disable the XRS-2 circuit temporarily. Neither of these modifications requires any circuit board traces be cut.

Easy to Use

After the tape baud rate question is answered, an introductory message and a menu of three options are displayed on the screen.

Option 1 dumps the disk, which must be in drive 0, to tape. All you have to do is load a blank cassette, set the recorder for record mode and answer the questions displayed on the screen. You are first asked the starting track number.

You may start with any track you desire. Pressing "enter" without giving a value defaults to an answer of 0.

You are then asked, "How many tracks on this diskette?" Pressing "enter" gives a

default answer of 35. If your disk contains more than 35 tracks, or you only want to dump a few tracks, you can indicate this by typing "40" or the number of the last track you want to dump.

Option 2 will restore the Dumpload tape to a disk. All you need to do is load the recorded tape in the cassette recorder, set it for play mode and load any formatted disk in drive 0. The tape contents will then be placed on the disk with each track being restored to its original position without any further action from you. If a checksum error is encountered, the recorder will stop. You can then choose to rewind the tape to the blank area preceding that particular track record and try Option 2 again, restore the track to disk with the checksum error or discontinue the restore attempt.

Option 3 permits you to verify that you have made a good tape. It will read the tape records, looking for checksum errors, but will not write to the disk.

Options 4 and 5, which allow you to exit Dumpload, are mentioned in the written instructions but are not displayed on the screen. Option 4 will return you to DOS Ready, and Option 5 will reboot the system.

How It Works

Dumpload creates a record or series of records on the cassette tape with each record constituting one disk track. The records are separated from each other by a blank area of tape which enables you to position the cassette at the beginning of any desired track record manually. A checksum value is computed for each disk track as it is processed before it is sent to the recorder. This checksum value and the track number become part of the actual record stored on the tape. Therefore, when a track record is being restored from tape, the computer can verify that the tape record is good and where that particular track record is to go on the diskette.

A Personal Experience

I wrote this article using my TRS-80 as a word processor. The article was about half finished, when the power company provided me with a two-second interruption in service.

My first thought was to congratulate myself for having just saved a current copy of my file to floppy. I then rebooted my disk. The drive motor clicked into action but nothing happened. The motor timed-out and stopped. I tried again and got the same results.

That's when I had my second thought: "Oh no, it's gone!" I inserted a different disk, booted, and everything worked perfectly. "Well, that's it. I have lost my article and all the other files on that disk, I thought. But wait, it acts like track zero is glitched and that might be the only problem." Since Dumpload can copy and restore a single track, I figured I might as well give it a try.

I loaded Dumpload, inserted a good disk into the drive, and a blank cassette into the recorder and dumped track 0 to tape. I rewound the tape, inserted the glitched disk into the drive, and loaded track 0 on the disk. I then booted the disk and was back in business again. My article, along with all my other files, was recovered safe and sound in about one minute, thanks to Dumpload.

Conclusion

Dumpload is highly interactive, and, therefore, is easy to use, even for the beginner. Once an option to save or restore a disk is chosen, it is as fully automatic and convenient as making a backup using two floppy drives. I have found it to be a very simple, inexpensive way to protect my large library of disks.

Dumpload is available from Complete Computer Services, 8188 Heather Dr., Newburgh, IN 47630. It is sold on cassette for $16.95 or on disk for $19.95. If you send them a disk containing TRSDOS 2.3 or NEWDOS80, they will install Dumpload on your disk, return it and charge you only $15.95. □

Multidos

Jim Klaproth

Multidos is a new full-featured disk operating system written by Vernon Hester, who created a Basic single-stepping utility called *Boss*. It is available from the Cosmopolitan Electronics Corporation for both the Model I and II computers and can be configured for either single, double, or P (Modified Double Density) Density. The DOS has many unique features, the most notable one is compatibility between different DOS systems. *Multidos,* however, is not a cure-all for all situations and it has its own limitations, but it has sufficient merits to warrant a full description of the system. The library commands will be described first, then the system utilities, and finally the Basic enhancements.

Jim Klaproth, 2012 25th Ave., S.E., Puyallop, WA 98373.

Library Commands

APPEND — Appends a file to the end of another file. When appending Basic programs, all must be in ASCII format and the line numbers must be higher in the second filespec.

ATTRIB — Assigns filespec attributes. There are seven levels of protection, similar to TRSDOS.

AUTO — Automatic command after reset. Multiple commands are supported and the limit is 32 characters after the "AUTO."

BLINK — Disables/enables the blinking cursor.

BOOT — Causes a hardware and software reset.

BREAK — Disables/enables the BREAK key.

BUILD — Creates a "DO" file, similar to Model III TRSDOS.

CLEAR — Zeroes RAM from 5200H to TOPMEM, the limit of free memory.

CLOCK — Displays the real time clock.
CLS — Clear the screen.
CONFIG — Default power-up drive attributes. The parameters that can be set are stepping speed, type of density, and the number of sides on each disk drive. Although these parameters are used to set up the drives, the DOS's automatic density recognition will override the density set in CONFIG.
DATE — Sets the current date and will display it if issued without any parameters. The date is automatically retained on non-powerup reboots and can be eliminated completely.
DEAD — All memory from 4000H upward is zeroed and the system will reset.
DEBUG — Activates the real time debugger.
DEVICE — Lists the current I/O devices and their routine entry points.
DIR — List the disk directory. The options are as follows: "A" — display an expanded directory including password level, date of last update, starting track and sector, end of file sector and byte within the file, number of segments, logical record length, and size in granules of each file. "I" will include all invisible files, "K" will include all killed files provided the directory entry has not been overwritten, "P" directs the output to the printer as well as the video, and "S" will include all system files in the directory. A really handy feature is the shorthand directory. By simply pressing the 0,1,2 or 3 key, the simple directory will be displayed for that particular drive.
DO — Substitute a disk file for keyboard input. This powerful command can cause the computer to perform several operations without operator intervention. DO files can be nested until user memory is exhausted.
DUMP — Write RAM contents to a disk file.
FORMS — Sets the following printer parameters: "I" initializes line counter and character counter to zero. "W" sets the line width, "P" sets the page length, "T" sets the text length, and "S" sets the blank spaces between printed text.
FREE — Displays the number of available file spaces, free granules, and kilobytes of disk space on all mounted diskettes, and will display the total diskette space available.
HASH — Returns the HASH code of a filespec.
HELP — Provides a brief explanation of each library command.
KEYBRD — Sets the keyboard attributes. The parameters are enable/disable the blinking cursor, the CLEAR key, the BREAK key, the graphics convertor, and the Epson printer graphics driver; and, which character to use for the cursor.
KILL — Delete a filespec.
LIB — Displays the library commands.
LINK — Links the printer, the display, and the RS-232 output for simultaneous output.
LIST — Display a diskette file.
LOAD — Loads an object file from disk to RAM.
PATCH — Modifies the contents of a disk file. Currently this is the only way to modify *Multidos*.
PRINT — Print out a disk file.
PROT — Changes the master password and locks/unlocks all visible and non-system files on the disk.
RENAME — Change the name of a file.
RESTOR — Recover a killed filespec. This only works if the directory entry was not overwritten on the killed file.
ROUTE — Redirects one I/O device to another.
SETCOM (Model III only) — Initializes the RS232 interface and sets the parameters.
SKIP — Allows the DOS to read a 40 track diskette in an 80 track drive.
TIME — Sets the correct time or displays it if no parameters are given.

TOPMEN — Sets the upper memory limit to protect high memory programs.
VERIFY — Causes all disk writes to be reread for parity.

System Utilities

Several system utilities are included in *Multidos*. Some take the place of the normal library commands found in other DOS systems. For example, note the lack of the library commands: BACKUP, COPY, and FORMAT. These are included in the system utilities, perhaps to allow easy recognition for creating a minimum size operating system diskette. One current limitation is the lack of a disk zap program, which is forthcoming from the author.
BACKUP/CMD is a menu driven utility that duplicates an entire diskette. It can create a mirror image copy with a different track count, but it cannot go from one density to another, which disappointed me greatly. If a Model 1 owner with single density drives obtains *Multidos* and then later adds a double density module, he must order a P density copy of *Multidos* in order to utilize maximum space on his system disk. A little greater flexibility here would be a great improvement.
COPY/CMD is a flexible single file copy utility that allows copying even from a foreign DOS system or a data diskette. Single drive copying of data diskettes is also supported. FORMAT/CMD is the disk formatter utility. It is menu driven and asks for the disk drive, the diskette name, the track count, the diskette date, the master password, the density to format in, and the directory track location. The diskette is checked for previous data before formatting begins to guard against accidental erasing.
CAT/CMD is perhaps the most powerful and useful utility on the disk. This little gem will pull a directory on practically any Model 1 or Model 3 diskette, regardless of the address marks, density, or sector/granule allocation. This means that if a Model 1 has a double density module installed, it can read a Model 3 directory created by virtually any DOS.
CONVERT/CMD is a utility contained only on the Model 3 version, but is also utilized as a library command (DDAM) on the Model 1 version. This command alters the address marks on a single density diskette in order to be read by *Multidos*. RS/CMD is a utility called the RAM Scanner. It will search the specified RAM locations for all occurrences of a single byte or a 16 bit word. It will optionally inquire about all calls, jumps, and loads to the selected word.
SPOOL/CMD is the *Multidos* RAM spooler. This allows the line printer to run continuously while simultaneously freeing up the CPU for normal operations. It is a very good spooler, with little noticeable slowdown.

TU/CMD is a TRSDOS file transfer utility that allows the user to transfer TRSDOS 1.3 files to a non TRSDOS Model 3 diskette. VFU/CMD stands for "versatile file utility" and it allows for multiple file copying, purging of files, printing a disk directory, and menu based execution of all programs on a disk. This allows purging or copying of system files in order to create a minimum system on a diskette.
GR/CMD is a Model 1 utility that allows producing graphics characters from the keyboard. The KEYBRD command must be configured for graphics in order to use this utility. On the Model 3, the DOS self configures itself for keyboard graphics, hence the lack of this particular utility.

Other DOS Features

Multidos also includes a utility

called *Mighty Multi,* which is similar to *Minidos* in NEWDOS/80. It is limited to only 4 commands, which are COPY files, obtain DIRectory, KILL a file, and LIST a file. It may be invoked during Basic with no disturbance of the resident program. *Multidos* on the Model 1 will automatically disable the interrupts when CLOAD, CSAVE, or SYSTEM is keyed in. The system will not hang, even if no disk is present in drive 0 and it will not start a write to disk unless sufficient free space is present on the disk. Another nice feature is the automatic repeat of the last DOS command by hitting the ENTER key.

Superbasic

Multidos' Basic, called *Superbasic,* is probably the most comprehensive Disk Basic brought out to date, and yet, it is the shortest Disk Basic around with 39979 bytes free in a 48K system. It adds so many new features to the original Basic in ROM that a comprehensive description of all of them is not possible. We shall hit the highlights of this excellent implementation. *Superbasic* is loaded by issuing the command, Basic from DOS READY. You may also specify number of file buffers, upper memory limit, and whether or not you want variable record lengths in random access file mode. You may also issue any valid Basic command, such as RUN"MYPROG/BAS" at the end of the loading command.

There are also 4 other Basic enhancements that can be invoked. Basic * will recover a previous program that was loaded into RAM after a re-boot or return to DOS. Basic ! will allow you to load a Basic program under an alien operating system, boot up *Multidos,* and recover the program stored in RAM. Basic # allows you to recover a Level II Basic program as long as Level II was entered via the CMD"X" call in *Superbasic.* This powerful feature allows a disk based system to develop BASIC programs in a Level II environment, while maintaining the option to save and load the programs on disk.

The final Basic available is called *BBasic,* which is a much enhanced *Superbasic.* BBasic adds all of the *Boss* single-stepping, trace, and variable review functions. Even at that, this version is still shorter than any other Disk Basic, taking only 1309 more bytes than *Superbasic.* With *BBasic,* you can single-step in three ways: to the end of a line, by a single instruction, or with a timed wait. The trace function can be sent to the video, to the printer, or be turned off. Break points can be set by inserting a POKE command in your program as a further aid in debugging. Variables can be reviewed selectively by first choosing which variables are to be reviewed during program execution. By pressing a certain key combination during execution, all variables selected in step one will be displayed in the order that they are entered, and then program execution will continue.

Editing commands have been given a shorthand to speed up editing. Enter a period will list current line, a comma will edit the current line, and a slash will list the "BREAK in" line number after the BREAK key is pressed. Up-arrow lists the previous line, down-arrow lists the next line, shift-up-arrow lists the first line, and shift-down-arrow+Z will list the last line. Single letter commands include [C]ont, [D.]elete current line, [E.]dit current line, [L.]ist current line, [M]ove line, [N]ew duplicate line, [P]age listing, and [R]un program. &H and &O are used for Hexadecimal and Octal constants.

There are a number of CMD functions that have been added to *Superbasic.* CMD"C" causes space compression of the current program, eliminating all spaces and linefeeds that are not in quotes. CMD"D" loads and executes DEBUG, CMD"E" will cause a brief explanation of the last DOS error code to be displayed. CMD"K" will zero a specified array. CMD"L" will delete a specified array. CMD"M" will move program line to a new location and delete the original line. CMD"N" duplicates a line without deleting the original. CMD"O" opens an addition file buffer and can be used within a Basic program for dynamic allocation of memory space. CMD"Q" is a high speed machine language string sort. It will sort 1000 items in about 7 seconds. It will work on a single or double dimension array. CMD"R" enables interrupts on a Model 1 only. CMD"S" returns to *Multidos.* CMD"T" disables interrupts on the Model 1 only. CMD"V" displays all scalar variables and string equivalents in the order that they were created. CMD"X" transfers to Level II while maintaining the resident program (see Basic # section). CMD"xxxxx" allows any valid *Multidos* function to be executed from *Superbasic.*

Superbasic also contains the usual TRSDOS functions of DEF FN, DEFUSR, INSTR, MID$, TIME$, and USRn. The file manipulation commands are KILL, LOAD, MERGE, NAME (which allows a new program to be loaded and run while keeping variable values), RUN, and SAVE. The file access commands are OPEN (5 modes are allowed: random I/O to existing file, random I/O to a new file, sequential output to a new file, sequential output to existing file, and sequential input from existing file), CLOSE, INPUT#, LINE-INPUT#, PRINT#, FIELD, PUT, GET, EOF, LOC, and LOF.

Also included are 4 overlay utilities. FIND will search for all occurrences of an ASCII string in a program. GE, which stands for global editing, allows you to change variable names, items in a data list, integers, strings, etc. You can also create compressed strings, merge lines, split lines, and change reserved words, such as PRINT to LPRINT. There is a fast RENUMBER utility and also a REFERENCE command that cross references all variables and integers.

Summary

Multidos' strength lies in the power of its Disk Basic and in its ability to attain a one-way compatibility between most disk operating systems. The only limitations are that Model 1 *Multidos* cannot read or write to a Model 3 TRSDOS or NEWDOS/80-2 diskette and Model 3 *Multidos* cannot write to a Model 3 TRSDOS diskette. A couple of my pet peeves were that *Multidos* cannot generate system disks in any other density than what is supplied and there is no 8 inch disk support (as yet). Another minor irritation was that although *Multidos* automatically supports C.P.U. speed-up to

SOFTWARE PROFILE

Name: Multidos

Type: Upgrade DOS With New Features

System: TRS-80 Models I, III

Format: Disk

Summary: Offers one-way compatibility between most disk operating systems.

Price: $79.95

Manufacturer:
Cosmopolitan Electronics Corporation
Box 89
Plymouth, MI 48170

5.3 MHz, the system refuses to re-boot while in high speed. The lack of a disk zap utility may bother some; however, Cosmopolitan has announced a utility called EASY ZAP, which may be available when you read this. *Multidos* is certainly the best bargain in a disk operating system considering that it only costs $79.95. For further information write to Cosmopolitan Electronics Corporation, P.O. Box 89, Plymouth, MI 48170. □

Throw Away Those Index Cards

Radio Shack's Information System

Steve Gray

In-Memory Information System This three-cassette $19.95 package from Radio Shack is, according to the TRS-80 catalog, "A collection of three assembly-language programs that can virtually replace any small index-card system. It will file inventory, name and address lists, phone numbers, investment portfolios and more." The Level-I programs are on one side of the cassettes, Level-II on the other.

According to the overview, the system "is designed to allow you to create, save, retrieve, modify and sort any type of data. Data can be made up of any characters: letters, numbers, special characters or any keys on the keyboard."

The nine-page manual provides an overview, tells what data files are and how to prepare them, how to use the three programs, and shows how to create large data files that require more than one data tape.

The **Initialization** program is for entering both the fixed information (record length, key length and field names) and variable information (data). The key is the first field, the one used to sort and retrieve data with. Each field, including the key, can be up to 19 characters long.

After you've entered all the information in the fixed and variable fields, you store it on one of your own tapes, as no blank cassettes are furnished with this package.

The **Retrieval and Update** program, after being loaded, first asks you to read in the data from the tape you've recorded. Then it displays a menu, which is a list of things you can do: add more records, obtain a list of the keys, look at or update or delete records, record data, or end the run.

The menu says that if you want to look at, modify (update) or delete any records, type an S. Then, to access a record, you type its key. The computer displays all the information stored in the records that start with that key. You can change or delete any part of this record, or move on to the next record.

The **Sort** program puts the records in order according to their keys. You load your data tape, press ENTER, and the sort is then performed automatically. As the manual says, "The advantage of sorting is that data is easier to find when it is in some ordered arrangement."

The sort can be on an alphabetic field, or a numeric field, or on a field made up of characters that are neither letters nor numbers, and in which, for sorting purposes & precedes $, which comes before #, etc. You can't use a semi-colon because that's for separating data items.

I'd decided to keep track of the Radio Shack software, so when the Initialization tape was run and the screen asked for record length, I entered 35, which is 7 for the catalog number, plus 20 for the description, 5 for the price, and 1 for each of the three fields. When the key length was asked for, that was 7, the catalog number. The screen then asks for the names of the fields and after that the data.

When I typed in the description of 26-1502, which is In-Memory Information, the program asked me to REDO, because I'd exceeded the field length of 20. I should have counted more carefully. The program lets you vary the length of any field except the key field, but once you've decided on the maximum length of a field, you can't go beyond that. So count carefully.

Then you can save the information by recording it on a blank tape. That's all there is to Initialization, unless you

Steve Gray is a frequent contributor to Creative Computing magazine.

made a mistake and want to record the data again.

In the Retrieval and Update program, when the menu came up, I typed A for adding records, added a couple of new software items to the list, and then recorded this latest version.

With the Sort program, when the sort is finished the screen says SORT COMPLETE, and adds that if you wish to save the sorted data, load a data tape and record.

The manual ends with a section on large data files. If you have large amounts of data, you have to use more than one tape. So if it's an employee file, for example, you divide the alphabet into as many parts as needed. The manual then shows how to compute the number of records your computer can store at one time, by dividing record length into available RAM.

This is a good information system to use if you have a fair amount of data to keep track of, say at least several dozen items. For fewer items than that, a card file would be adequate and would require less manipulation. Even with large files, it all comes down to this: how much time would you save by computerizing them, and are the files important enough to you to make the computerizing worthwhile? If the answers are "a lot" and "yes" then you should check out IMIS, Radio Shack's In-Memory Information System. ■

Versafile by Radio Shack

Versatile File Manager for the TRS-80

Fredrik O. Haarbye

SOFTWARE PROFILE

Name: Versafile

Type: Database management

System: TRS-80 Model I Disk

Format: Disk

Language: Basic

Summary: Inexpensive yet versatile file manager

Price: $29.95

Manufacturer:
Radio Shack
1800 One Tandy Center
Fort Worth, TX 76102

If you are looking for something that will make your computer more useful, *Versafile* from Radio Shack may be the program for you. Enter this program and your computer is turned into an information system.

The program is written entirely in Disk Basic and loads in 5952 bytes (Version 2.2, Model I). At $29.95 it is one of the best buys around. It is very nicely packaged, and includes:

- An 18-page manual (in a hardcover three-ring binder; value $5.95).
- TRSDOS on the program diskette (value $14.95).
- *Versafile* Program ($29.95 - $20.90 = $9.05).

If you know of another program as useful as this for $9 let me know; I want to buy it.

What makes *Versafile* score high is its versatility—its name is very appropriate. Take a look at one example of a *Versafile* Index (Figure 1) and you will recognize some useful "personal" applications. And its usefulness in business is just as great.

The manual provided with the program suggests as examples a used car inventory, an insurance agent's client information file, and a foreign word dictionary. Figure 2 may give you some more ideas.

The program uses sentence-oriented storage and retrieval. File entries are stored under eight different keywords which may be used or changed at the user's option. Any word with seven or fewer characters can be used.

A file entry may or may not contain one of these keywords, and there may be more than one keyword in the sentence. The keyword may be placed anywhere in the file entry. Any character on the keyboard, except the arrows, may be used in the file entry, and 2 to 238 characters (spaces included) will be accepted per entry.

The tab key (right arrow) can be used to tab eight spaces. A period at the end of the file entry serves as delimitor.

Storage of a file entry is reasonably fast. For example after a 238-character entry has been typed in, the computer is ready for a new entry about seven seconds after the Enter key has been pressed. An improper entry will result in "I don't understand your entry. Please try again."

Searching

Searching the file is done by entering one or more words (or numbers) followed by "?". If a valid keyword is included in the search request, the file for that particular keyword is searched for a match. A "global search" can also be requested. In this case keywords can be omitted, and all keyword files will be searched.

To shorten the search time, the program has 37 "unnecessary words" which can be changed at the user's option. Any of these words present in the search request will be removed from the request line prior to search, as will the keywords. The selection of all of these words (keywords and unnecessary words) is important and may affect the search time.

Fredrik O. Haarbye, 5510 Broadmoor Plaza, Indianapolis, IN 46208.

Figure 1.

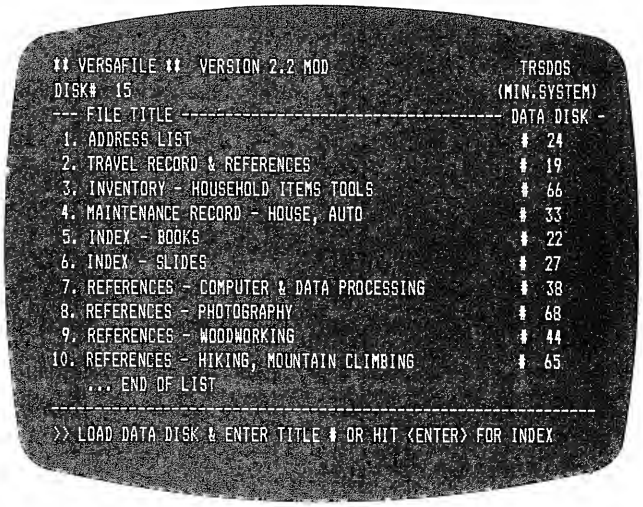
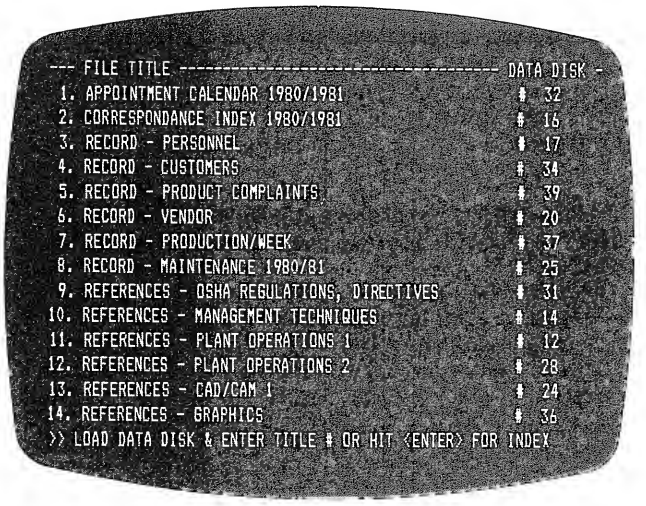


Figure 2.



The computer will search for a minimum of five file items before outputting any information. Therefore, if there is only one match (anywhere) in the file, the whole file will be searched before an output is made. This is a bit of a nuisance but can be corrected with one of the program modifications listed at the end of this article.

A search through a 1000-item data file (average 50 characters per data item) may take about three minutes, while a search through 60 of these items may take 14 seconds. One nice feature of the search function is that it does not give up easily.

If a search request is made with a keyword and no match is found in that keyword file, the other files will also be searched. Output to the printer is optional.

Another (minor) nuisance: an empty line plus the request sentence is printed for every five file items listed. This can also be corrected with a program modification.

There is no facility to edit a file record directly, but this may not be a real handicap since records can be removed from the file with single or multiple kill commands. If the information supplied in the kill command is inadequate, a different file may be killed. However, the record that has actually been killed is listed for verification.

Another thing that makes this program outstanding is its documentation. Author William D. Schroeder should be complimented on this as well as the program quality. Twenty-two remark lines scattered throughout the program nicely explain the program statements.

In addition to an extra copy of the *Versafile* program, the disk contains all of the standard TRSDOS Library Commands and three utilities: FORMAT, BACKUP, and BASICR (re-numbering Basic). It is not likely that these routines (except possibly BACKUP) will be used when *Versafile* is running. The manual explains the use of FORMAT and BACKUP.

To sum it up, this program packs a lot of capability into a small amount of memory space. Although it has no alphabetic or numeric sort ability, this may be an advantage since there are no complicated instructions needed to use it. The program has performed reliably for me during six months of extensive use.

Program Modifications

I have made the following seven modifications to *Versafile*. They provide more disk space, customize the program, and make it easier to use.

1. Kill all unnecessary files on the disk to provide an additional 16 granules of disk space. It is not likely that any of the following files supplied with the program will be used when running *Versafile*. If there is a need, slip in an un-modified disk.

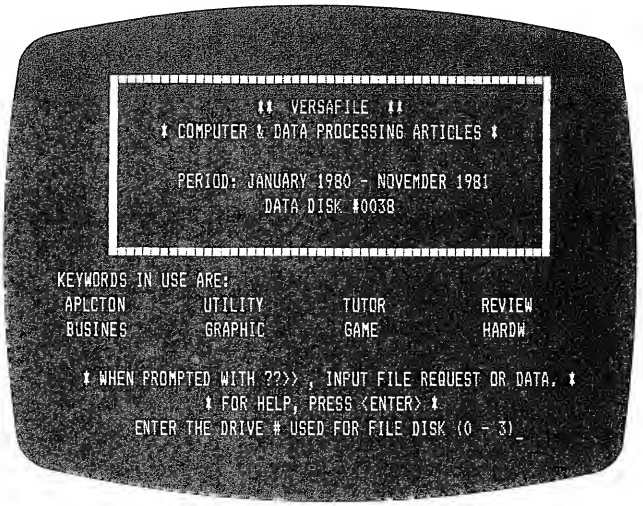


Figure 3.

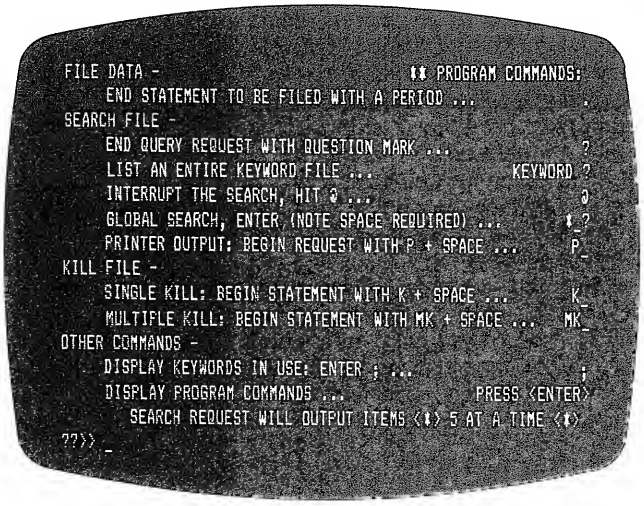


Figure 4.

In the DOS READY mode enter:

KILL FORMAT/CMD.FORMAT (3 grans)
KILL BACKUP/CMD.BACKUP (3 grans)
KILL BASICR/MCD.BASIC (5 grans)
(Do not kill BASIC/CMD)
KILL VERSA/BU (5 grans)

2. Removing all remark lines adds about 1200 bytes of RAM space and one granule of disk space.

Delete the remark parts of lines 70 and 125, and lines 299, 369, 999, 1199, 1369, 1399, 1497, 1609, 1654, 1669, 1699, 1999, 2099, 2999, 3089, 3999, 4999, 5999, 9999, and 18999.

If you have a utility which will remove spaces in the program lines (such as NEWDOS/80 Ver. 2.0), you can gain another 400 bytes.

3. This modification replaces the "Title Page" with one that has information pertinent to the file created by the program. In addition to the title of this file, it also shows the keywords in use for the file. (See Figure 3.)

Delete all lines to and including line 70 and add the lines 10 through 90 and 23000 and 23010 in Listing 1.

4. It is useful to be able to show the keywords any time the program is running. Adding line 320 of Listing 1 and lines 23000 and 23010 of modification 3 will do it.

Also, having to hunt through the manual to find an explanation of program commands is a bit of a chore. To bring all of the commands to the screen any time, change line 300 and add lines 325, 335 and 24000-24100 of Listing 1. (See Figure 4.)

5. Sometimes it may be desirable to change the search to fewer or more than five items before an output is made on the CRT. In response to the ??>> prompt, when a number 1-9 is entered, this becomes the number of items searched before an output is made. Change lines 125 and 1405, add lines 315, 330 and 335 as shown in Listing 1.

6. When several references are listed at one time, it is handy to have them numbered. The redundant printing of an empty line and the request sentence when listing to a printer can also be changed. Add line 1654 and change lines 1655, 4000 and 4010 as in Listing 1.

7. Some odds and ends: When several listings are made for one file request, line 1656 wastes space and makes the output a bit confusing. The same is true for line 1660. I would also suggest changing lines 110 and 120 as in Listing 1. □

Listing 1.

```
10 CLEAR100
15 TITLE$="*  COMPUTER & DATA PROCESSING ARTICLES  *
20 S1$="PERIOD: JANUARY 1980 - NOVEMBER 1981
25 S2$="DATA DISK #0038
30 CLS:PRINT@151,"** VERSAFILE **
35 PRINTTAB((64-LEN(TI$))/2)TI$
40 PRINT:PRINTTAB((64-LEN(S1$))/2)S1$
45 PRINTTAB((64-LEN(S2$))/2)S2$
50 PRINT@70,STRING$(51,131);:FORY=4TO23:SET(113,Y):NEXT
55 PRINT@518,STRING$(51,131);:FORY=5TO24:SET(12,28-Y):NEXT
60 PRINT@576,;:GOSUB23000
65 PRINTTAB(3)"* WHEN PROMPTED WITH ??>> , INPUT FILE REQUEST OR DATA. *
70 PRINTTAB(17)"* FOR HELP, PRESS <ENTER> *
75 PRINTTAB(8)"ENTER DRIVE # USED FOR FILE DISK (0 - 3) ";:PRINTCHR$(95);
80 D$=INKEY$:IFD$=""THEN80
85 IFD$<"0"ORD$>"3"THEND$="":GOTO80
90 CLS:PRINT"DATA DISK IS IN DRIVE #"D$:POKE16410,ASC(D$)
110 CLEAR 12000
120 DEFSTR A,B,C,D,E,F,K : DEFINT L,X,Y,Z:DIME(160),Y(160),A(25),K(25)
125 POKE27000,5:DS=":"+CHR$(PEEK(16410))
300 B="";Z=1:PRINT"??>> ";:LINEINPUT B:A=" "+B
315 IFB)"0"ANDB<"9.5"THENPOKE27000,ASC(B)-48:MP=1
320 IFB=";"THENCLS:GOSUB23000:MP=1:GOTO335
325 IFB=""GOTO24000
330 IFMP=1THENPRINTCHR$(27)CHR$(27);:PRINTCHR$(255):FORX=1TO25:NEXT:PRINTCHR$(27)TAB(4)"NEW";
335 IFMP=1THENPRINTTAB(8)"SEARCH REQUEST WILL OUTPUT ITEMS <#>"PEEK(27000)"AT A TIME <#>":MP=0:GOTO300
1405 Q=PEEK(27000):IFK(2)=""ANDZ>QTHENRET=1:Z=Q:GOSUB1650:RET=0:Z=1:E(Z)=" "
1654 IFP=1ANDZ>5THENTV=Z
1655 IF P=1 THEN TP=TV-Z:FOR L=1TOZ:IFY(L)=Y1THEN TP=TP+1:LPRINTUSING"### ";TP;:LPRINT E(L):NEXT: ELSE NEXT
1656 IF RET=1 AND P=0 THEN PRINTTAB(60)"-"ELSE IF RET=1 THEN RETURN
1659 IFG$="9"THENPRINTCHR$(27)TAB(40)"- SEARCH TERMINATED -":PRINT:RUN110ELSEIFRET=1THENRETURN
1660 PRINTSTRING$(63,45):IFP=1THENLPRINTSTRING$(79,45):RUN110ELSERUN110
4000 IF INSTR(MID$(E(L),2,59)," ")=0THENTV=TV+1:PRINTUSING"### ";TV;:PRINTMID$(E(L),2,LEN(E(L))-2);:
     IFLEN(E(L))=62THENRETURNELSEPRINT:RETURN  ELSE LT=LEN(E(L))  :LS=1:LE=60:
     IF LT<250 THEN E(L)=LEFT$(E(L),LEN(E(L))-1)+".":LT=LT+4:T     :1:PRINTUSING"### ";TV;
4010 IFLE=LSTHENPRINTTAB(4)RIGHT$(E(L),LT-LS):RETURNELSEIFMID$(E(L),LE,1)>" "THENLE=LE-1:GOTO 4010
4011 PRINTTAB(4)MID$(E(L),LS+1,LE-LS-1):LS=LE:LE=LS+60:IFLS>LT THEN  RETURN ELSE IF LE=>LT
     THEN EL=MID$(E(L),LS+1,LT-LS):IFEL=""THENRETURNELSEPRINTTAB(4)EL:RETURN:ELSEGOTO4010
10000 DATA " APLCTON "," UTILITY "," TUTOR "," REVIEW "," BUSINES "," GRAPHIC "," GAME "," HARDW "
23000 PRINT"KEYWORDS IN USE ARE:
23010 RESTORE:FORX=1TO4:READKX$,KY$:PRINTKX$,KY$,:NEXT:RESTORE:PRINT:RETURN
24000 CLS:PRINT"FILE DATA -"TAB(40)"PROGRAM COMMANDS:
24010 PRINTTAB(5)"END STATEMENT TO BE FILED WITH A PERIOD ..."TAB(60)".
24020 PRINT"SEARCH FILE -":PRINTTAB(5)"END QUERY REQUEST WITH QUESTION MARK ..."TAB(60)"?
24030 PRINTTAB(5)"LIST AN ENTIRE KEYWORD FILE ..."TAB(52)"KEYWORD ?
24040 PRINTTAB(5)"INTERRUPT THE SEARCH, HIT @ ..."TAB(60)"@
24050 PRINTTAB(5)"GLOBAL SEARCH, ENTER (NOTE SPACE REQUIRED) ..."TAB(59)"*"CHR$(95)
24060 PRINTTAB(5)"PRINTER OUTPUT: BEGIN REQUEST WITH P + SPACE ..."TAB(59)"P"CHR$(95)
24070 PRINT"KILL FILE -":PRINTTAB(5)"SINGLE KILL: BEGIN STATEMENT WITH K + SPACE ..."TAB(59)"K"CHR$(95)
24080 PRINTTAB(5)"MULTIPLE KILL: BEGIN STATEMENT WITH MK + SPACE ..."TAB(58)"MK"CHR$(95)
24090 PRINT"OTHER COMMANDS -":PRINTTAB(5)"DISPLAY KEYWORDS IN USE: ENTER ; ..."TAB(60)";
24100 PRINTTAB(5)"DISPLAY PROGRAM COMMANDS ..."TAB(50)"PRESS <ENTER>":MP=1:PRINTCHR$(27):GOTO335
```

Special Delivery

Customized Lists and Letters

Herb Friedman

SOFTWARE PROFILE

Name: Special Delivery

Type: Mailing and form letter system

System: 32K TRS-80 Model I, II, or III, Disk Drive

Format: Disk

Language: Assembly

Summary: Excellent

Price: $125

Manufacturer:
Software Concepts
169 Preston Valley Shopping Center
Dallas, TX 75230

Special Delivery is a complete mailing system designed specifically for single or multiple disk drive TRS-80 computers. Unlike other mailing list programs that simply create one or more files that can be listed or searched on demand, *Special Delivery* provides for multiple coded "mail files," the printing of merged letters, conversion of standard Radio Shack software mailing lists into the *Special Delivery* format, and training. The training consists of three sample files for practice in preparing and using a list of names and addresses (the mailing list), preparation of merged letters, and preparation of mailing labels.

Before we go any further we'd better take time out to explain the merge process, because it is the main reason that *Special Delivery* is more than just an ordinary mailing list program.

If you think back to the "junk mail" you received a few years ago offering once-in-a-lifetime buys—real estate in the desert miles from water, "free" trips to vacation swamplands, and the like—you will recall that the mailer tried to personalize the letter as if you and your family were personal friends—the only ones in town to receive the offer.

Unfortunately, computer software wasn't all that good a few years ago, so the program simply dropped your name into a prewritten "computerized" format, and the letter might have looked something like the sample in Figure 1.

Obviously, with all that open space around the name and hometown you had to be a dummy to believe you were getting a personal letter.

The spaces are caused by the manner in which the letter is prepared. Because the software simply drops the name and address into a prepared text, sufficient space must be left to accommodate the longest names and addresses. Whatever space isn't used by characters appears as empty space in the final copy.

But now, with improved software, things are more precise. Word processors can merge the mailing list into the computerized letter and eliminate extraneous spaces with a "pull-up," which simply tightens up the text to the insert. Today, the same letter would appear as in Figure 2.

The letter now really looks and reads like a personal letter. If the printing is done on a letter quality printer, no one could tell for certain whether it is, in fact, a single, personal letter, or one of several million letters which are identical except for the name and address. Keep the merge in mind as we discuss *Special Delivery*.

Special Delivery consists of three distinct programs and three sample files. The programs are: Mailform, Mailrite, and Convert (which converts Radio Shack lists to the *Special Delivery* format).

Mailform

Mailform is a stand-alone machine language program for creation and editing of name and address files (mailing lists). It runs in an overwrite mode with full cursor control that permits selective editing of data entries. If you make a spelling error when entering data you can correct only the error; there is no need to redo the entire data block.

It supports both the Electric Pencil and Radio Shack upper-/lower-case hardware, and supports the Electric Pencil control key, and the Radio Shack control, which consists of simultaneous operation of the

Figure 1.

```
We are pleased to inform the          Joe Doakes
family of        Anywhere,        AK,        that
you have won a free trip to the moon.
```

Figure 2.

```
We are pleased to inform the Joe Doakes family of
Anywhere, AK, that you have won a trip to the moon.
```

Herb Friedman, Tridac Electronics Corporation, 588 Hewlett St., Franklin Square, NY 11010.

Shift and down-arrow keys.

The functions of Mailform are activated by the control keys. A function menu with a short description of each function is displayed at all times at the right side of the CRT screen, as shown in Figure 3.

Figure 3 is an almost identical print of the CRT screen display made with the NEWDOS JKL function. The difference between Figure 3 and the actual CRT display is the underscore next to the data entry area on the left side of the screen.

Each space that can be filled with a character representing data is indicated on the CRT by an underscore. The printer shows this as a left-arrow because Radio Shack has their printers do some weird things with underscores and up-arrows (the standard exponentiation symbol).

The left side of the screen is a record, the individual entry for each person. A character can be entered only where there is an underscore; for example, all states must be identified by a standard Post Office two-letter abbreviation. You cannot use more than two letters because the program will not accept input beyond the end of the line—the last underscore on the right.

Each record has eight Data Fields, from NAME at the top through DATA 2 at the bottom. The list can be searched for information in any of the eight fields, and information in selected fields can be printed out on a list, on labels, or in a letter.

Normally, the DATA fields contain information about the individual, but they can also be used for foreign addresses. For example, assume you want to mail a letter to Canada. Even the accepted abbreviation of "CAN" won't fit in the STATE field. Also, Canada has postal codes that resemble long distance telephone numbers. You handle this situation by simply using the DATA fields. You can put the special postal code in the DATA 1 field and the country in the DATA 2 field.

Directly under DATA 2 is the RECORD # line, which contains space for user entry of the record number for a particular mailing list, and has displays that read LAST and MAX.

LAST and MAX are controlled by the program and cannot be modified by the user. The three-digit number to the right of LAST is the Record number of the last entry. The three-digit number to the right of MAX is the maximum number of records that can be entered for a specific file (mailing list), which is determined by the amount of memory in the computer.

The program looks at the top of memory—leaving 100 bytes free for user use if desired—and calculates how many records can be included in each specific file.

Under the RECORD # line is the FILESPEC line. Here, the user enters the "code" name assigned to a specific mailing list

Figure 3.

```
        MAILFORM -V 1.3- (C)1979 BY SPECIAL DELIVERY-WATSON
NAME     ←←←←←←←←←←←←←←←←←←←←←←←←←      F - PAGE FORWARD
COMPANY  ←←←←←←←←←←←←←←←←←←←←←←←←←      B - PAGE BACK
ADDRESS  ←←←←←←←←←←←←←←←←←←←←←←←←←      G - GET RECORD
CITY     ←←←←←←←←←←←←←←←←←←              P - PUT RECORD
STATE    ←←                             D - DELETE RECORD
ZIP      ←←←←←                          S - DEFINE SEARCH
DATA 1   ←←←←←←←←←←←←←←                 C - CONT. SEARCH
DATA 2   ←←←←←←←←←←←←←←                 T - SORT RECORDS
                                        R - READ FILE
RECORD # ←←← LAST 000  MAX 188          W - WRITE FILE
FILESPEC ←←←←←←←←←←←←←←←←←←←←←←←←       X - EXTRACT FILE
                                                E - EXIT
```

Figure 4.

```
        MAILFORM -V 1.3- (C)1979 BY SPECIAL DELIVERY-WATSON
NAME     Doakes, Joe←←←←←←←←←←←←←←      F - PAGE FORWARD
COMPANY  Wonder Lawn Care←←←←←←←←←      B - PAGE BACK
ADDRESS  1000 Pleasant Street←←←←←      G - GET RECORD
CITY     Anywhere←←←←←←←←←←              P - PUT RECORD
STATE    AK                             D - DELETE RECORD
ZIP      55555                          S - DEFINE SEARCH
DATA 1   three children                 C - CONT. SEARCH
DATA 2   rocketry←←←←←←                 T - SORT RECORDS
                                        R - READ FILE
RECORD # 029  LAST 028  MAX 188         W - WRITE FILE
FILESPEC Space/B←←←←←←←←←←←←←←←←←       X - EXTRACT FILE
                                                E - EXIT
```

Figure 5.

```
        MAILFORM -V 1.3- (C)1979 BY SPECIAL DELIVERY-WATSON
NAME     Doakes, Joe←←←←←←←←←←←←←←      F - PAGE FORWARD
COMPANY  Wonder Lawn Care←←←←←←←←←      B - PAGE BACK
ADDRESS  1000 Pleasant Street←←←←←      G - GET RECORD
CITY     Anywhere←←←←←←←←←←              P - PUT RECORD
STATE    AK                             D - DELETE RECORD
ZIP      55555                          S - DEFINE SEARCH
DATA 1   three children                 C - CONT. SEARCH
DATA 2   rocketry←←←←←←                 T - SORT RECORDS
                                        R - READ FILE
RECORD # 029  LAST 109  MAX 188         W - WRITE FILE
FILESPEC ←←←←←←←←←←←←←←←←←←←←←←←←       X - EXTRACT FILE
                                                E - EXIT
         ACTIVE FILE = Space/B
```

when creating a file, and enters the file name when calling in a file from memory.

Figure 4 shows a typical entry for a record. There is the name of a person in last/first order, the company he works for, his address, how many children he has, and the family hobby (rocketry).

The RECORD # line shows that this is entry 29 in a list that already contains 28 records out of a possible maximum of 188 (the number of records for 32K of memory).

The FILESPEC line shows that this record is assigned to a file (mailing list) named (coded) "Space/B." (Space/A and Space/C, etc., would be other mailing lists of people with similar interests, because a file can contain—in this instance—only 188 records.)

Figure 5 shows how the screen appears if the Space/B file is loaded from memory. Note the line at the bottom that reads: ACTIVE FILE = Space/B. This indicates you have the Space/B mailing list in memory.

Notice that we have "paged forward" from record number one to record number 29, which is the entry we made in Figure 4. Also notice the figure to the right of LAST; it shows 109, indicating we have a total of 109 records in the Space/B file. There is still a great deal of room left in the file.

Depending on what we would like to do, we can list or print the entire list from record number one to the end, or search for specific information by the record number, or field. For example, we could search and list those people in a specific town, street or postal zone, or by DATA information.

If files are too large to fit into the available memory, they can be segmented, then edited and saved to disk under a different FILESPEC. Conversely, files can be chained together for continuous processing as one list.

Mailrite

The Mailrite program is designed to print letters, envelopes and labels from a mailing list created by the Mailform program. It is supplied in two versions: Mailrite/CMD and Mailrite/CEN.

The difference between the two is the type of printer with which it is to be used. Mailrite/CMD is for printers that support an underscore. Mailrite/CEN is for Centronics-type printers, which automatically linefeed when a carriage return is received, and do not support underscore.

For letter preparation, the original Mailrite text is prepared using the Electric Pencil. Special flags imbedded in the text tell Mailrite what data from the mailing list is to be substituted for the flag and merged into the text. The flags represent complete name, last name, address, state, DATA, and so forth. An example of a merged letter is shown in Figures 6 and 7. Figure 6 is a print of the original text with flags, as prepared using Electric Pencil software.

Starting at the top of the letter we have two somewhat confusing lines. These are instructions for the operator and are placed at the top of the letter where they are ignored by Mailrite. (Remember, when you use Mailrite, the top two lines don't print. If you start your letter at the top you will lose the first two lines. This also applies to envelopes and labels. The top two lines are provided so specific instructions can be included with each letter.)

The print of the letter will start with the normal DATE location, which in our form letter has the line YOUR LUCKY MONTH, DAY, AND YEAR. Then come the flags, which are indicated to the program by the less than (<) symbol. N is for the complete name, C the address, and so on. Notice how the flags are imbedded within the text of the letter.

One of the millions of recipients of this form letter will be our friend, Joe Doakes, who provided the data for record number 29 shown in Figure 5.

Figure 7 shows how the letter appears when Mailrite merges the mailing list into the text. There is absolutely no one who can say with certainty that this is a form letter from its appearance. Each bit of information about the Doakes family is correctly positioned and there are no tell-tale extraneous spaces to indicate that the letter is computerized.

After Figure 7 there is nothing more that can be said of *Special Delivery*—that letter says it all. It takes some practice to become familiar with all of *Special Delivery's* features—two to five hours for hassle-free preparation of files and imbedded flags—but the results are more than worth the time spent learning all the ins and outs of this software package. □

Figure 6.

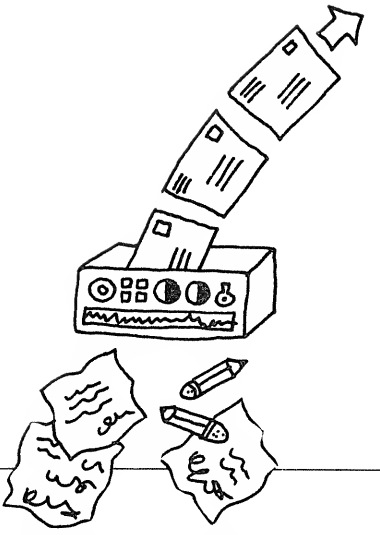
```
SET TO 60 WORDS PER LINE
Form letter for moon trip. Send to families with children
YOUR LUCKY MONTH, DAY, AND YEAR
Mr. <N
<C
<A
<T, <S <Z

Dear Mr. <L:

CONGRATULATIONS to the <L family of <T, <S.  You and your <1,
all active <2 hobbyists, have won a free trip to the moon on
the first NASA passenger space shuttle.  We share the
excitement of the <L family and want to make the trip of you
<2 hobbyists as enjoyable as possible.

If you will forward $1000 as a "good faith" deposit (non-
refundable), we will insure that the <L family, of <T, <S,
will be on the first flight.

Best regards,

Bill Skybound, President
```

Figure 7.

```
YOUR LUCKY MONTH, DAY, AND YEAR

Mr. Joe Doakes
Wonder Lawn Care
1000 Pleasant Street
Anywhere, AK 55555

Dear Mr. Doakes:

CONGRATULATIONS to the Doakes family of Anywhere, AK.
You and your three children, all active rocketry hobbyists,
have won a free trip to the moon on the first NASA passenger
space shuttle.  We share the excitement of the Doakes
family and want to make the trip of you rocketry hobbyists
as enjoyable as possible.

If you will forward $1000 as a "good faith" deposit
(non- refundable), we will insure that the Doakes family,
of Anywhere, AK, will be on the first flight.

Best regards,

Bill Skybound, President
```

A TRS-80 Mailing List Program

Mailroom Plus



Rod Hallen

I have come across many problems that were designed to maintain mailing lists, but none as easy to use as Mailroom Plus from The Peripheral People. In addition, Mailroom Plus, which was originally written for the National Rifle Association, has some features that greatly enhance its utility.

MRPLUS is written in Basic and requires at least 32K of RAM and one disk drive. All lists are maintained on the disk, but file manipulation takes place in memory which is a much faster method of operation. Similar programs use random access disk files but this takes time. The difference is particularly noticeable when sorting lists.

Application

MRPLUS is set up to accept Name, Address, Telephone number, Info and Category. Info can be anything that you'd like to add to the file and Category can be a membership number, Ham call sign, equipment bought or sold, or anything else you would like to use for identification.

On entry to MRPLUS the following Menu is displayed:

1. Create new list
2. Add to list
3. Edit list
4. Search & display
5. Search & print
6. Sort

10. Safe on disk
11. Input from disk

The documentation is quite well done and a sample name list is provided to give you some practice in using MRPLUS. The manual takes you briefly through the various options that are available and you quickly become used to the method of operation. Prompt messages in all of the right places help to keep things straight.

The manual then describes each of the following options in great detail.

1. Create new list — asks for a name for this list and then prompts with Name, Street, City, etc. and waits for an entry for each. Names are entered and stored in reverse order, i.e., Smith John but they are displayed and printed in the normal way (John Smith). You can null any entry with the Enter key. After each record is entered it is displayed and you are asked if it is correct. You can then make corrections or go on to the next record. Entering END for the name will terminate the list.

The fields (name, street, city, etc.) are not limited to size as long as the length of the entire record does not exceed 255 characters. This means that long entries will not have to be unnecessarily abbreviated and should be adequate for most situations.

2. Add to list — This selection allows you to add more records to an existing list. They are added to the end of the file and can then be integrated using the Sort feature.

3. Edit list — If it is necessary to go back into a list to review it or make corrections, this selection is made. You step through a record with the enter key until the line requiring correction is reached. That line is then retyped.

4. Search & Display — This is one of the outstanding features of MRPLUS. It will search and display any record which contains a stated string. If you give "John" as the search string, every record that contains the word John will be displayed one at a time under your control. If the search string was given as CA 92, then all California entries with ZIPs starting with 92 would be displayed.

The power of this feature really shines when you have entered pertinent information into the Category line. If you are a computer store owner and want to keep track of the equipment your customers have purchased, you might enter the following for one of your customers:

Category? .12345.L2.32K.DISK2.

This would be serial number, Level II, 32K RAM and two disk drives. Using Search & Display you could come up with a list of all of your customers who have disks, Level II or any other information you enter here.

Maintaining a club list would work in the same way. If Category contained membership class, number, interests, etc., you could search for any desired group within the club. Businessmen could keep track of backorders, customer interest, or any other desired category.

5. Search & Print — works the same as 4 except that the output goes to the printer instead of the video screen. In addition, you may output the entire record or just the information required to print address labels. These can be selected as one or four labels across. If you want to print the entire file instead of just selected parts, a separate program called "AUTO-PRINT" will do that for you.

6. Sort — Choose Name, ZIP, or Category and this will sort a file in alpha-numerical order. Since the sort takes place in memory, it is quite fast. Much faster than a disk sort using random access disk files. After the sort is finished, you are given the option of eliminating duplicates from the list. Record numbers of duplicate records are displayed and you can erase the appropriate one. Once the file has been sorted you can use option 5 to print, or option 10 to save on disk.

10. Save on disk — A file in memory can be saved on disk at any time and each file has a name. If you bring in a file from the disk, make some additions or changes, and then save to

Rod Hallen, PO Box 73, Tombstone, AZ 85638.

the disk with the same file name, the new file will replace the old one. If the old file was called "NAMES" and you name the new file "NAMES1," then both files will be on the disk. The capacity of the disk is the only limitation to the numbers of files you can save.

11. Input from disk — Brings a file in from the disk for adding, changing, or printing.

In connection with memory and disk capacity, the following information is given: you can get a 150 to 300 name file into memory depending upon how long each record is. In order to protect you from losing a file due to an OUT OF STRING SPACE error, MRPLUS starts to display the amount of memory left after you have exceeded 150 records in a file. If a file is getting too big, you can break it down into one or more parts and save each part as a separate file on the disk. You can also merge two smaller files into one larger one.

The manual recommends keeping MRPLUS on one disk and the name and address files on a separate disk. It also recommends that you record each file twice for backup protection. In connection with this, information is given on doublesiding your disks. This will double your recording capacity. I am using the Wangco 82 disk drive which will read either side of a disk without modifying the disk. (I have heard that Wangco is going out of the disk drive business and that the Model 82 can be purchased at a very reasonable price.)

You can also increase the capacity of your data disks by erasing Basic, BACKUP, FORMAT and any other unnecessary programs. This can be done using the Radio Shack Master Password which is provided.

I guess I've made it clear that I'm sold on Mailroom Plus. It is available from The Peripheral People at P.O. Box 524, Mercer Island, WA 98040 for $30 postage paid. It is supplied on cassette for transfer to your diskette or they will record it on a customer provided disk.

Mailroom Plus makes creating, maintaining and printing mailing lists a very simple chore whether it is a personal list or a group of business or membership lists. I have been using another mailing list program for some time but I am now converting all of my files to Mailroom Plus. □

Grammatik From Aspen Software

I Sing the Editor Electric

Stephen Kimmel

Name: Grammatik
Type: Prose editor
System: 32K Model I, II or III TRS-80, Disk drive
Format: Disk
Language: Machine language
Summary: Useful addition to word processing software
Price: Model I-$49, Model II-$99, Model III-$59
Manufacturer:
Aspen Software Co.
P.O. Box 339
Tijeras, NM 87059

Since I started reviewing software back near the dawn of time, I have seen very few programs that I thought everyone should have. I have seen many programs that were just plain bad and many programs that struggled to be adequate. I have seen several programs that were good and a few programs that I thought were extremely good. I don't get excited about many programs. The last one was *Scripsit*.

I am excited about *Grammatik*.

A short time ago (March '82) I did an article reviewing spelling checkers. I said that the proofreaders were of minimal value. After all they merely tell you which words to look up in the dictionary. Alas, spelling is only the beginning of the story. Even if all my words were spelled correctly I could still appear an idiot with my limp word choice, bad grammar, and terrible typing skills. So, while spelling is important, what I really need is something that will take care of all those other things.

What writer wouldn't give anything for an editor he could turn off? Who doesn't want a program that will turn out brilliant text? Or at least words the editor will buy? An electronic editor who would catch all the mistakes that make you look like an idiot would be perfect. Sigh. Such a program isn't available yet but the pieces are

Stephen Kimmel, 4756 S. Irvington Place, Tulsa, OK 74135.

beginning to appear.

First we had the spelling checkers. Now we have *Grammatik* to check style, usage, punctuation and a little bit of grammar.

In the Beginning

Bruce Wampler, the author of *Proofreader*, sits down one day and says to himself, "Self, why can't I do the same thing for grammar and style?"

"The answer is obvious," self says to Bruce, "spelling is a piece of cake compared to checking grammar."

"Yeah, but Bell Labs is doing it with the Writer's Workbench."

"Bell Labs has a Megalith 5000 computer and 800 computer scientists working for them."

Dr. Wampler, however, is undeterred, and after long hours of labor brings forth *Grammatik*, a program to check style and grammar. And, just for laughs, he decides to let it check for sexist terms as well.

Searching for Offenders

I can hear you now. *Grammatik* can check my grammar and style? How? *Grammatik* contains a "dictionary" of 500 commonly misused phrases. Figure 1 lists several phrases, error types and suggestions used in *Grammatik*.

The program scans through your ASCII or normal word processor document in search of these offenders. Whenever it finds one it stops and makes a suggestion.

Every time you use the word "hopefully," it will tell you that you have used a commonly misused word ("hopefully" means "in a manner that is full of hope." It doesn't mean, "it is to be hoped.") and that you should avoid it.

Every time it finds "I myself" the program will tell you that it is redundant and that "I" is sufficient. Unfortunately, the program won't check for endings that are misused. The endings "wise," "ize" and "ly" are horribly overused and can almost always be eliminated to the benefit of all. I think the next version will scan for things like this.

Probably the best way to describe what *Grammatik* does is to list the error types that the program uses. See Figure 2.

Grammatik also checks the number of sentences and words and the average length of the sentences and words. It displays the percentage of short and long sentences. Using that information, it is a simple task to calculate your "fog index." And it counts the occurrences of forms of the verb "to be" and the common prepositions.

All these are indirect measures of the readability of the document. Long sentences and long words are signs that the document is hard to understand. Too many uses of the verb "to be" indicate over-use of the passive voice, which saps the strength from a document. A high ratio of these and the prepositions indicate that the work can probably be improved by rewriting.

So far, we have seen several impressive implications for *Grammatik*. Note that *Grammatik* doesn't know anything about the meaning of words. It can't check for subject-verb agreement, dangling participles, or split infinitives.

Of course, there are cases in which *Grammatik* will call something an error that is actually correct. This seems to be particularly true of the sexist analysis of fiction. There are cases where "his" is the correct word. Robert picked up his ball. The document must still be checked by a regular human.

Turning the Program On

The advertisement had me drooling with desire. I had to have it. So, I ordered one on Friday. Astoundingly, it arrived on Monday. I went directly to my trusty TRS-80 and began to analyze a short story of mine, "Snow Before the Summer Sun."

It is difficult to imagine the program being any easier to use. Typing GMK starts the program. It is menu driven and most of the commands take a single keystroke. D reads in the dictionary. I=SNOW told it which file to check. F= > told it to ignore the Scripsit format characters. I told it to start by typing "//."

The story began to appear on the screen. When the program found what it considered a problem, it stopped to call my attention to it and to give its suggestion. When I hit Enter, it went on. When it was finished with the story, it gave me its final analysis table as shown in Figure 3. It only took a few minutes.

I leaned back, sending my swivel chair to its farthest back position and closed my eyes. My fingertips massaged my forehead as though that would drive away the slowly creeping headache. Carefully, I glanced

Figure 1. A Sample of Grammatik Phrases.

"Incorrect Phrase"	Error	Suggestion
a number of	Wordy	several, many, some
ain't	Informal	revise
alas	Archaic	revise
already	Misused	already=by now
alot	Spelling	a lot
and/or	Awkward	revise
hopefully	Misused	avoid
I myself	Redundant	I
must of	Improper	must have
seldom ever	Redundant	seldom
stick to your guns	Trite	persevere
through the use of	Wordy	by, with
very	Vague adverb	avoid or revise
Xerox	Trademark	photocopy

Figure 2.

A—archaic usage
B—Unbalanced ( ) or “”
C—Capitalization error (such as “i” or “STePhen”)
D—Doubled word or punctuation (and and ,,)
E—User defined error
G—Gender specific term (“laborer” instead of “workman”)
I—Informal usage (“ain't”)
J—Jargon or technical term
K—Awkward Usage (“and/or”)
M—Commonly misused word (“eminent,” “imminent,” and “immanent”)
O—Overworked or trite (“stick to your guns”)
P—Punctuation error
R—Redundant phrase (“seldom ever” should be “seldom”)
S—Spelling error (“alright” should be two words)
T—Trademark (Xerox is not a verb)
U—Improper Usage (“must of” is wrong)
V—Vague adverb (“very” adds very little)
W—Wordy phrase (“a number of” can be replaced with “several”)

at the screen. The program was good. Very good. Yet there was this gnawing ache that bothered me and wouldn't go away.

Perhaps my initial expectations about *Grammatik* had brought me to this: late night, tired eyes, wishing that I was a smoker so that I could forcefully snuff out a well chewed butt. I picked up the magazine and let it fall open to the page. The advertisement said that the program would check capitalization, punctuation, doubled words and several commonly misused words. The program did all that and more. If anything, the advertisement was a classic example of undersell. I was incredulous: here was a program that didn't claim to bring heaven on earth and actually delivered more than it promised.

I lifted the soda can and found that it had been empty for hours. A quick toss in the general direction of the waste paper basket showed why the Lakers still weren't interested in me. I settled for covering my mouth with my hand.

I had expected too much from the program or perhaps too little of myself. Although I like to consider myself a professional class writer I am, in my heart, still the English student who got a "C" because the teacher was merciful. I expected the program to tell me a hundred ways to improve my story. It didn't. It found six errors and I had committed all but one of those intentionally. I expected the program to turn me into Harlan Ellison. It had told me that I was a pretty good writer already.

Sitting down once again, I stared at the glowing white letters. I had decided that the program would be useful to a professional writer if only because it checks for doubled words. Considering the price, a mere $49 for the TRS-80 Model I, I could even recommend it on that basis. Don't sell your kids to get one though.

Who Needs It?

Who needs this program then? Where does it fit in the cosmology of computer software? It is such a nifty program that it has to be perfect somewhere. The connection dawned on me finally. This program is like having an English teacher available to you all the time. Running it on a regular basis would improve your writing skills.

Several people have reported that they receive fewer problem messages as they work with it longer. *Grammatik* would be a good choice for students and anyone who feels his writing skills could be improved. Considering the state of American literacy, that includes most of us.

I have a higher enthusiasm threshold than that. Like the spelling checkers, *Grammatik* can also be expanded to include phrases of particular significance to you. It can become your personal editor and English coach. I love to start sentences with "and." That's not a particularly good idea. So I loaded the *Grammatik* phrase dictionary into *Scripsit* and I added the phrase ".And" with the note to be careful not to overuse it. Now I get a reminder everytime I do it.

There are other words that I use too often. My copy of *Grammatik* checks for them, too, and gently tells me to watch out. I am constantly looking up "its" and "it's." Now the program stops to remind me which one is which. "Affect" and "effect" are the same way.

Grammatik has room for 300 additional phrases. That is more than enough. After three hours I added only 50 phrases. It is this utility that turned that initial evening's frustration into enthusiasm.

I like *Grammatik* a great deal. At $49, it is a worthy and useful addition to your word processing software. Now, if someone will write a program to detect dangling participles and split infinitives...Or better yet a program that will turn me into Harlan Ellison. □

Figure 3. The Grammatik analysis for "Snow Before the Summer Sun."

```
Summary for SNOW / Problems detected: 5


# sent: 330 ; words: 3000

avg sent length: 9.1 ; avg word len: 4.1

# questions: 2 ; # imperatives: 4

short sent(<14 wds):270 ; long sent(>30 wds):3

longest 36 wds at #76 ; shortest 1 wds at #184

to be's: 20 ; prepositions: 188

User category totals:

NONE
```

Six Spelling Checkers For the TRS-80

Hte Proofreader Porgrams

Stephen Kimmel

No, the title isn't a mistake; it's a joke. Also it's an illustration of a very common error that this group of programs is designed to catch; the simple typographical error. For those of us who are only marginally talented typists, it seems our fingers sometimes don't do what our brains tell them to do. We know that "the" isn't spelled "hte." Our fingers just got a little confused. We're probably the same folks who could never rub our tummies and pat our heads at the same time.

To add insult to injury, our eyes are in on the conspiracy to make us look like fools. If I had put "porgrams" in the text instead of the title, a sizeable percentage of us would never have found it. Add to all of this the fact that, at the end of a long day, even the words you know are right look wrong.

Is there hope for us otherwise intelligent individuals? Are we forever doomed to the mocking laughter of our cohorts when they read our letters? Was all that time spent sleeping in English Composition wasted?

Look! Up in the sky! It's a bird! It's a plane! No, not one, not two, but five spelling programs for the TRS-80. (I don't know if there are analogous programs for the Apple or the Atari other than Spellguard which requires CP/M. Perhaps the presumption is that anyone smart enough to buy an Apple is smart enough to spell.)

Before we get too excited by all of this, allow me to throw a little cold water on the subject. It should be noted that no program available on any machine checks your grammar. (With the possible exception, that is, of the recently announced Grammatik from Soft-Tools. More on this late-breaking development at six.) None of them will be able to tell if you should have said "sale" or "sail." Or even "them" or "then," "win" or "wine," "a cross" or "across."

Stephen Kimmel, 4756 S. Irvington Place, Tulsa, OK 74135.

Consider the difficulty of writing a proofreading program. You can't just say "computer, pick out all the words that aren't right." You have to instruct the computer how to tell if a word is wrong. Consider the title. Everyone would recognize "hte" as a misspelling since "hte" isn't even a legal combination of letters in English. "Porgrams" however, could be a word—perhaps a trade name for a new line of super cheap software. In the context of this magazine, however, and in the common experience of everyone reading it, it almost certainly is an error.

The poor dumb computer doesn't have your experience. So how does it decide whether a word is right or wrong? It checks all the words in a document against a dictionary of several thousand words stored in memory. If the word isn't on the list then it asks (At this level the program becomes so simple that I'm tempted to offer it as a programming exercise.) thus gaining the benefit of your experience. Or at least your ability to look in a bigger dictionary. That's why the size of the dictionary is featured so prominently in the ads for this type of program. The more words it knows, the fewer times it has to ask you.

How many words is enough? Webster's New Collegiate Dictionary contains over 150,000 entries. Few folks, however, know anywhere near that many words. Many secretaries have little books of about 20,000 words. My third novel contained only 4,000 unique words. This article has less than 1,000 unique words. Hence, a program should probably contain, at least, 10,000 words to be adequate, and be able to expand to fit your particular word choice.

Some of the programs seek to expand their effective dictionaries by recognizing suffixes and prefixes. I have mixed feelings about this, since it can get you into trouble. Suppose your program ignores the last letter "s" unless it is "ss." For lots of words this is all right. However "alumnus" would become "alumnu" and incorrectly called wrong while "alumnas" would become "alumna" and called, again incorrectly, right. Words such as "has," "was," "this" and "metropolis" would also fall into the cracks. The results could be disastrous if you were to rely too heavily on the computer.

I suspect that this sort of game could be played for any common prefix or suffix. Name a suffix or prefix and find a word in which that letter combination isn't a prefix or suffix. Would "rely" disappear completely?

The other approach would be to list the legal suffixes and prefixes in the dictionary. That seems awkward and doesn't gain you that much. However, most suffixes and prefixes do follow the rules and this technique does reduce dramatically the number of times a program must stop to check variations on the same word. It won't keep you from having to check the words yourself.

Which brings us to the great Achilles' heel of this sort of program. No group of computer programs illustrates the fallacy of relying on computers more graphically than these. It comes back to the old saw, "garbage in, garbage out." Although I prefer a more current rendition of the same maxim: "A computer can make mistakes in seconds that would take a dozen men days to make."

You see, the computer assumes you know what you are talking about, whether you do or not. If you tell it to spell the word "seperate" "or accross," it will believe you. Your proofreader program will then repeat your mistake with amazing speed. When the program truly fits your vocubulary, it will contain all your own errors and won't be proofreading at all. Further, as you may have deduced, what these programs do is tell you which words to look up in a dictionary. You are paying to have a program tell you which words it thinks you should check. Nothing more. Nothing less.

Thus, this entire class of programs is of

rather minimal utility. On the other hand, lots of people like me have problems with simple typos. That's why no fewer than seven of these programs have appeared essentially simultaneously.

I had available to me for this review five of the nine programs of which I am aware. A brief discussion of the missing programs seems in order; prefaced by the comment that I am unqualified to comment extensively, since I haven't actually seen any of the others. Most of this information is derived from manufacturers' advertisements and literature.

The entry of Radio Shack into the proofreading sweepstakes was to be available at the end of November, 1981. Initial reports are that it has a 30,000 word dictionary on the Model I/III and a 60,000 word dictionary on Model II, and will cost $149. It appears that the Model II version may be worth the effort but that's too much money for the Model I/III. Based on Radio Shack's record, you'll be able to buy it sometime in March, 1982.

Spellguard

A review of Spellguard appeared in the July issue of *Creative*. Advertisements for it claim that it will proofread 10,000 words in one minute. I am somewhat suspicious of that claim. My 48K TRS-80 won't hold 10,000 words (about 60,000 bytes) in a single load. That could be 10,000 very short, essentially all correct words on a multidrive system with a 4Mhz clock CPU. Maybe.

A general comment on speed is in order here. None of the claimed speeds include you looking up any words in an outside dictionary. They tell you how long it will take the program to compare the text to their dictionary. Be aware that it takes longer to proofread a document than they imply, unless the document didn't need proofreading anyway.

Superspell

Also available for CP/M users is a program called Superspell that comes as part of the Select word processing program. An "S" keystroke from within your word processor invokes the proofreader and its 10,000 word dictionary. This program is not cheap at about $395, but then it is a complete word processing package.

Now for the main subject of the review. These are arranged in the order that I received them.

Proofreader

The Soft-Tools Proofreader program is a three-disk package that includes two dictionary disks and the main program disk. It was written by Bruce Wampler in what appears to be Fortran. The approach is as unique as the choice of language. You invoke this program entering "PROOFRDR" from DOS. Proofreader then asks for a file to proofread. It takes your text and creates a list of the unique words in your document. It will check for the word "and " only once.

The program can handle a document of essentially infinite length, as long as there are only 1100 unique words. The list is then sorted and compared to the next disk. The unknown words, which presumably include the misspelled words, are then displayed and/or printed and/or saved to a disk file. To that extent, the operation of the program is very simple and almost bulletproof. The task remains for you to decide if the remaining words are misspelled or simply not included in the 38,000 word dictionary. You have to find them in the text—not a big deal with the automatic search feature of most word processors—and make the changes.

Proofreader makes a single effort at recognizing suffixes. It assumes that a final "s" preceded by a consonant is plural unless that consonant is also an "s." There are very few words for which this isn't true. Typically, these are verb combinations and you won't get into trouble there.

At $89 Proofreader is the least expensive of the group and at the same time contains one of the most extensive dictionaries. The dictionary can be expanded to an additional disk of plain text words. As the documentation states, though, the more extensive the added dictionary, the slower the program will run. For the user who is willing to do a little extra work and who is as tight with his dollars as I, Proofreader is the program of choice.

Soft-Tools, which recently changed its name to Aspen Software Company, has recently announced Proofedit, a program that will make Proofreader interactive with Scripsit. Proofedit will also give the user full ability to add and delete words from the dictionary. Although I haven't seen this combination, it should make the work quite a bit easier for the user. The price for the combined package is just $119.

Soft-Tools has also announced an entirely different sort of program in Grammatik. This program will check punctuation, repeat words and do at least a minimal check on your grammar and style. I'll be doing a more extensive review on Grammatik in another article.

Hexpell 2

Hexpell 2 by Hexagon Systems requires two disk drives, and they aren't kidding about that. It can, however, be supported on a single double-density disk and an appropriate system. Hexagon reports that sometimes there is excessive disk activity.

It is much easier to run than Proofreader—once you get it running. Because of their scrupulous honesty, their program is one of the hardest to get running. Radio Shack—bless their pointed little heads—told them that they couldn't include even a minimal TRSDOS so they sent out the program with no system at all.

Before you can use this program you have to copy their programs and data files onto a TRSDOS (Or other DOS. Hexspell appears to work with most systems. NEWDOS 80 appears to give it the most trouble) diskette. No fun at all. Surely all those other disk programmers aren't writing their own systems! Kudos to Hexagon Systems for their integrity. A Bronx cheer to Radio Shack for their shortsightedness.

Once you get Hexspell running, it is a pretty spiffy program. Here you begin with "BRUN SP" which starts the Microsoft compiler run module. Again you enter the name of the file you want to proof. The program checks through your document one word at a time, displaying the document as it goes. The chosen speed is about 200 words per minute.

At this pace you should have little problem reading along, and the Evelyn Wood crowd may find it drags a little. This gives Hexspell one of the slowest times of the group.

The display is vaguely similar to Scripsit's. If it finds a word that isn't in its dictionary (up to 28,000 words), it stops and asks. You are shown several words before the word in question and the rest of the sentence. Thus, you see the word in context. You can leave it alone, replace it with a corrected word or add it to the dictionary. A corrected version of the document is automatically saved to the disk.

Actually the program can learn many more than 28,000 words. The file is structured so that when word 28,001 is learned, the least used word is thrown away. The program changes to match your word selection exactly. Hexspell's 28,000 words then are more than adequate. If you have two disk drives and are still tight with your dollars, then Hexspell may be your program.

One brief sour note. If you are running with a lower case modification, then be sure to have a lower case driver working before you run Hexspell. The program doesn't have a driver and has our old friend, the TRS-80 keyboard reverse, working. Any words you replace will be all caps. It might also give you problems cleaning up the dictionary.

Chextext

Chextext from Apparat is a solidly designed program for $79.95. It has a dictionary of 10,000 words which can be expanded up to 50,000—if you have a dual,

Program Name	Proofreader	Hexspell 2	Microproof	Spellguard
Available From	Aspen Software MHE Box 14 Tijeras, NM 87059	Hexagon Systems Box 397 STN A Vancouver, BC Canada V6C 2N2	Cornucopia Software P.O. Box 5028 Walnut Creek, CA 94596	Pelican Programs 49 Pelican Ct. Syosset, NY 11791
Cost	$89	$99 US	$165 (as reviewed)	$295
Required System	32K - 1 Disk	48K - 2 Disk	32K - 1 Disk	CP/M
Supplied Dictionary Size	38,000 words	25,000 words	50,000 words	20,000 words
Time to Correct this Article	17:35 minutes	32:40 minutes	13:45 minutes	*
Words Questioned	80	212	73	*
Comments	Non-correcting. The user must make changes manually. Least expensive.	Creates file with corrected document.	Automatically corrects original file. Works from within Scripsit.	Reviewed July 1981 *Creative Computing.*

Table of Programs.

80-track disk drive. Presumably it works with NEWDOS 80 to support that kind of equipment. For a $3.00 handling fee, Apparat will send you a 20,000 word dictionary on a diskette that will handle twice the capacity. Chextext needs more diskette for its dictionary, because they haven't encrypted the dictionary. This makes Chextext's dictionary maintenance one of the simplest and most exhaustive around.

You have two choices in using Chextext. You can operate it as a separate program with "CHEXTEXT" or you may use a Scripsit modification routine that is supplied and invoke Chextext from within Scripsit with a "P,CHX" command. The program then begins the usual Scripsit print check. This can be a hassle if you weren't expecting it to find formatting errors. But you have to correct them sometime and it might as well be now.

Then you get into Chextext itself and the program begins to proofread your document to eliminate multiple checks. That finished, it checks the words against the dictionary and saves any unmatched words to an internal "suspect" word list. After all the words are checked you are given the choice of ignoring the word, adding it to the dictionary, marking the word in the document (the last letter is changed to a # sign wherever it occurs) or forgetting the whole thing. If you mark misspelled words it is still incumbent upon you to go through the document using the search and replace option to correct the words.

Operated as a separate program, it is very similar to Proofreader. Everything is menu-driven. It appears that you have to work through the program separately to gain the full advantage of the dictionary maintenance and the other features of the program.

As expected, this is a solid bit of programming with nothing to be said against it. I found only one bug in the program, and it was merely an annoyance rather than a genuine problem. The dictionary is adequate, but there are larger dictionaries available. There are programs available that don't require two disk drives and programs available that work faster. Dictionary maintenance is the strong suit here.

Microproof

The last program to arrive may very well change my mind about the importance of price. At $165, it is the most expensive of the group, but it has the largest dictionary, works the fastest and is the easiest to use. I'm speaking of Microproof by Cornucopia Software. It comes in less expensive configurations. One version that costs $70 works much like Proofreader. For another $60 you can get it to make the corrections as Hexspell does. The final $35 buys you the ability to invoke the proofreader from within Scripsit or Electric Pencil.

The claims for Microproof are, like those of Spellguard, so strong that I was originally tempted to reject them out of hand. The 50,000 word dictionary is the largest claimed by 30%. It is claimed to be infinitely expandable. (Infinite is a whole lot and would probably require an infinite number of disks. I don't have that many.)

It also claims to proofread and correct 10 pages of text (single-spaced pages have about 600 words each) in less than a minute. All this and more on a 32K single 5" disk system? Well, closer examination shows the claim to be for double-spaced pages using the CP/M version of the program. The author, Phil Manfield, admits that the TRS-80 version is much slower.

But it is still fast. And it combines the approaches of the other systems. Integrated with Scripsit or Electric Pencil, the program can be invoked with a single command from within the word processor program.

The first thing the program does is save the text to diskette. It then reads the file back in creating a unique word list as Proofreader does. The program compares the words to the dictionary. The program calls for the appropriate diskette changes so it can work with one disk drive. A list of unknown words which is usually very small is then displayed. Finally, the program will go back through your file, searching for the occurrences of the changed word and makes the changes. If there were words you wanted to see in context they will be shown at this time. I was dazzled by how quickly it worked. Like the others though, it slows down considerably while waiting for me to look through the dictionary to check my spelling.

A dictionary of 50,000 words? All of the programs except Chextext use various systems to encrypt their words and reduce the amount of memory required. Microproof goes further in this reduction than any other. I have no idea if the dictionary

The Word	Chextext	WordSearch	Superspell	MIZ Spell
Oasis Systems 2765 Reynard Way San Diego, CA 92103	Apparat, Inc. 4401 S. Tamarac Blvd. Denver, CO 80237	KEYbits Inc. P.O. Box 592293 Miami, FL 33159	Select Information 919 Sir Francis Drake Kentfield, CA 94904	Programs Unlimited Dept. 881M Box 265 Jericho, NY 11753
$75	$79.95	$195	$395, Apple; $595, Z-80	$49.95
CP/M	48K - 2 Disk	CP/M	CP/M	Model I 48K disk
45,000 words	10,000 words	8,000 words	20,000 words	18,000 words
*	21:20 minutes	*	*	*
*	267 words	*	*	*
*	Solid, reliable program. Dictionary in ASCII.	*	A full featured word processing program.	*

*Not available for testing.

actually contains that many words, but there are quite a few words in this article that only Microproof caught. Fifty thousand words may be a bit excessive, but in this type of program it is best to have the largest number of words you can afford.

Microproof does the best job of recognizing the suffixes and prefixes. I think it does this by encoding what type of suffix goes with what type of word. You are given the option of describing a new word as a noun, a verb, an adjective or an adverb. I can think of no other reason for doing this.

So far, I have been unable to find a Microproof claim that didn't appear to be true.

The Test

Proofreader is the cheapest. Microproof the most expensive. Proofreader is the most difficult to use. Microproof is the simplest. This doesn't really begin to give the complete picture of what is happening here, however.

You want something that has an acceptable speed, contains all the words you have doubts about, and most important, will make your life easier. That, after all, is what the computer is all about.

So I ran a test on the programs. How fast do they work? How many times do they stop to ask you what you meant? I chose something available to everyone—this article. I'll leave it to you to judge how representative my word choice is.

I have thrown in a few odd ones just to check the vocabularies. Few people routinely work "kudos" into their daily conversation. This article contains, pending editoral intervention, approximately 3600 words and fills about 16 double-spaced pages or eight single-spaced pages. Excluding the Table of Programs, it contains 942 unique words, counting the intentional misspellings. As such, it is a little longer than most business letters and comparable to most reports.

As I noted before, the time involved in proofreading a document has to include going to the dictionary to check everything out. None of the advertised times include any allowance for this. Yet, if you are like me, you'll end up checking some words that look all right to you. To be fair, I allowed time to check ten words in the dictionary. All of the programs questioned more than ten words so this seemed reasonable to me.

The times required vary considerably, and are shown in the attached tables along with the number of words checked. The results? Microproof was overwhelmingly the fastest, taking less than fourteen minutes to proof and correct the document. Proofreader was quite respectable at seventeen minutes. This was largely due to human shortcuts. Hexspell took quite a while because of the feature which displays words at reading speed.

Recommendations

For the average user—meaning someone who doesn't earn his living writing on his computer—a proofreading program is of minimal value. So if you are going to get one it is best to get the least expensive you can find. The best program in the low price class is Proofreader. It had acceptable speed and the ability to match itself to your vocabulary.

For those who can justify the added expense to go first class, then there simply is no finer program available than Microproof for the TRS-80, Apple or CP/M system. □

Statistics With the TRS-80

Randy Heuer

Although the sales of microcomputers indicate that the general public is becoming convinced that a small computer can be of some benefit or use, much of the commercially produced software has been concerned with less than serious applications. As a result, many people not familiar with programming end up with a sophisticated and rather expensive video game.

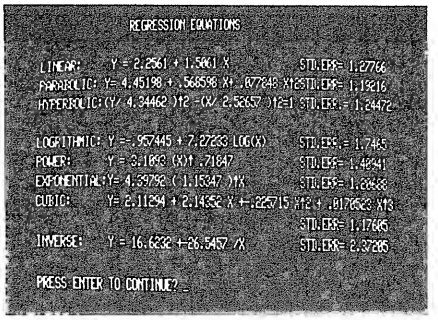
Fortunately, things are beginning to change. With games now saturating the market, software manufacturers are turning to more serious applications. Some of these applications will, of course, take advantage of the computer's ability to perform numerical calculations quickly and accurately.

One of these applications is complex statistical calculations. In the past, people who needed the results of these calculations were forced to spend countless hours on a handheld calculator or many dollars to use a large computer. For these users, a microcomputer, with an appropriate software package, can be an excellent solution to the problem.

Such a package has been introduced by Creative Computing Software — Advanced Statistics for the 16K, Level II TRS-80 (CS-3303). The package comes with a cassette tape containing nine programs and a comprehensive 5" x 8" documentation booklet packaged in a vinyl binder. The retail price is $24.95.

This package features eight advanced statistical procedures which, in the past, were only available on large computer systems with complex statistical packages. Now anyone can have access to such procedures anytime for less than $900.

Randy Heuer is a frequent contributor to *Creative Computing* magazine.



The Programs

The heart of the package is the Data File Management routine. This program allows the user to create and modify cassette-based data files. By employing a special feature which loads only fifty data records into memory at any one time, data files can contain an unlimited number of values. This feature allows one to analyze large amounts of data on a relatively small computer. The data files constructed by this program can be used interchangeably with seven of the statistical programs. Thus a variety of tests can be conducted on the same data.

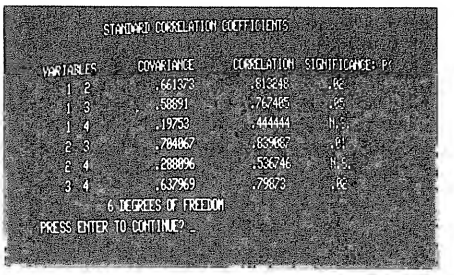
The first statistical program calculates six descriptive statistics of a single variable. These statistics include the mean, median, standard deviation, range, skewness and kurtosis. The program also displays the number of occurrences, percentile and Z-score for each value.

The Two Variable Statistics program is used to calculate the statistical relationships between two variables. In addition to the basic descriptive statistics for each variable, a t-test for the difference in the means, the linear correlation coefficient and its significance level along with a linear regression is performed.

The Crosstabulation program is used to determine whether two methods of classification of multinomial data are independent. For example, one might have the following data concerning the size of car involved in an accident and whether or not there were any fatalities:

	TYPE OF CAR			
Type of Accident	**Small**	**Medium**	**Large**	**Total**
Fatal	67	26	16	109
Non-fatal	128	63	46	237
Totals	195	89	62	346

The Crosstabulation program allows one to determine if there is any



statistically significant relationship between the size of car and the severity of the injury sustained. The chi-square test conducted by the program will determine if there is any evidence to conclude that the size of the car has any relationship to the severity of the accident. In this case, such a hypothesis is not supported by the data.

While this example uses a rather simple 2 x 3 table, the program can handle tables as large as 10 x 10. In addition to the chi-square value, the program also calculates the level of significance and gamma statistics.

The Regression-Trend Analysis program computes the least squares regression coefficients from paired data. The end result of this procedure

is a set of predictive equations. The following models are calculated: linear, parabolic, hyperbolic, logarithmic, power, exponential and cubic. For each model, the program calculates the standard error of estimate. The user may then select any of the models and enter values of the independent variable. The program will provide estimates for the dependent variable.

Multiple Linear Regression is also used for developing predictive equations. In multiple regression, several independent variables are used to forecast the dependent variable. This program allows the user to perform multiple linear regression using up to ten independent variables. It computes both the unstandardized and normalized regression models and also calculates the correlation matrix, multiple correlation coefficient and standard error of estimate.

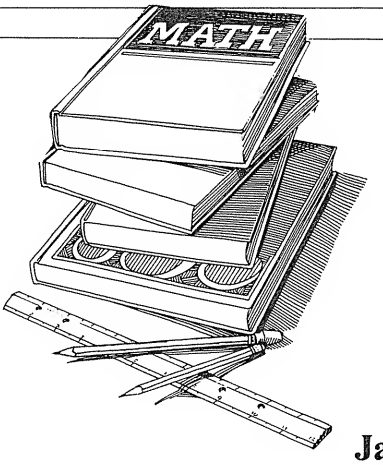
The Correlation Analysis program computes the product-moment correlation matrices, multiple correlation coefficients and partial correlation coefficients along with their associated significance levels. An unlimited number of cases for two to five variables may be entered.

Analysis of Variance is a statistical procedure for testing whether the means of three or more groups are equal. Both the standard, one-way ANOVA and the two-way method for classifying data may be employed. Up to 100 individual groups may be included in each test. The program calculates and displays the standard analysis of variance table along with the mean and standard deviation for each variable.

The final program is the Advanced Multiple Regression procedure. This test is similar to the Multiple Regression procedure but provides several additional statistics not available on the other procedure. These statistics include the analysis of variance, a t-test for the population correlation coefficient, t-tests for the individual regression coefficients and values for E(Y) and Y(new).

In Conclusion

The performance of statistical tests is an excellent application for microcomputers. We at Creative Computing Software feel that this package is an outstanding group of statistical programs for the TRS-80 and that it will provide the statistical power that most users require. We believe that this package is a harbinger of many useful packages that will make the microcomputer more than just a sophisticated toy. □



Factoring Whole Numbers

Math For Older Students

James S. Coan

The promotional material for *Factoring Whole Numbers* states: "These are highly interactive programs that your students will enjoy. Each lasts about 20-30 minutes depending on student responses. Pacing to suit the individual 'talking' to the computer is built into the programs so that the student succeeds as learning occurs. The Series is appropriate as new material for students in the upper elementary and junior high schools, and for review or remedial work at the secondary or junior college level."

I have the Apple Version 2.0 programs for 16-sector 3.3 DOS. This comes on three diskettes, each of which contains four well designed factoring exercises.

James S. Coan, RD #1, Box 149, New Hope, PA 18938.

Documentation

The software comes in a nice three ring notebook. The diskettes are in plastic jackets at the back. You get a title page, a single CPU software license, and five pages devoted to the teaching material itself. Non-profit educational institutions may purchase permission to make copies for additional CPUs. None of this material is required for the student. It is intended for the teacher.

However, there is no discussion of how to get the teaching material running on the computer, nor is there any mention of how to terminate a session.

The Programs

The user soon learns that the display screen is set up for upper and lower case letters using the Apple hi-res graphics screen. The menu offers to run any of the programs on the current disk, and makes instructions optional. Each program has a nice leading hi-res display screen. Responses are ended by pressing Return. For word responses it is sufficient to enter the first letter only.

Figure 1 shows the contents of the three disks. Each A program introduces the topic and provides some practice. Each B

SOFTWARE PROFILE

Name: Factoring Whole Numbers

System: TRS-80 16K, 6 tapes
TRS-80 Model I 32K,
3 diskettes,
TRS-80 Model III 32K,
3 diskettes,
Apple 3.2 32K,
3 diskettes,
Apple 3.3 Vers. 2.0 48K,
3 diskettes
Pet 16K, 6 tapes
Pet 16K, 3 diskettes

Summary: Twelve interactive lessons, games, and exercises

Price: $90

Manufacturer:
Quality Educational Designs
P.O. Box 12486
Portland, OR 97212
(503) 287-8137

program is designed to extend the concept and encourage the student to explore further. Most of the B programs finish up with a game or contest involving two or more students.

To quote the author: "Some of the B programs go beyond the standard junior high curriculum. These are designed to preview, in an informal fashion, ideas which will be important in later mathematical work." The material seems to do this very well.

Each lesson/exercise/game is designed to accommodate a wide range of skill levels within the topic being presented. Many of the instructions are optional. The display speed is user-controlled. Some of the games may involve up to six students.

The package is definitely user-friendly. All responses which one might expect are well fielded by the programs. Wrong answers are handled very well.

The analysis of responses is quite complete. It notices illegal decimal numbers, numbers too high, numbers too low, and illegal negative numbers, and requests, "Number please" when all else fails. It even handles responses such as 1.4E1 when the correct response is 14. This is well done.

There is never a sarcastic word in this package. Right answers are treated with enthusiasm, while wrong answers are handled in a supportive way. The whole idea is to promote successful progress through the programs.

Often a program will offer to explain a concept. This provides the first time user a chance to get the maximum instruction, and allows the repeater to pass over unwanted detail.

Some of the programs ask questions designed to assess just how appropriate this lesson is for the student at the time. If the user's responses are reasonable, the lesson proceeds. If his responses are clearly out of line, the program suggests another lesson to be studied and returns to the menu.

All questions recognize Q as a request to select another program. Ctrl-C or Reset at any time also returns the user to the menu.

Each diskette may be run independently by either turning the Apple on with it in the disk drive or by typing in #S (slot). One can easily move from one diskette to the next by placing the next one in the disk drive and issuing the Q request. There is no direct method offered for ending a session. The user is expected to turn the Apple off.

The optional instructions are well paced. The screen is presented a few lines at a time. Each screen is frozen in place with "Press return to continue" displayed on the bottom line.

I have selected a few sections to review here in detail.

Diskette One

Introduction: The introduction explains how to use the computer. The student is offered the option of all upper case or upper/lower case display. He can even control the time delay between sentences displayed on the screen. (It is assumed that the user has been able to put the disk in the drive and turn the machine on.)

1A Factor Pairs: Factoring a whole number into a factor pair is presented as finding the possible lengths and widths of a rectangle with a given area. It is recommended that you obtain 100 square floor tiles. Clearly you should get the little 1 inch by 1 inch ceramic ones. Alternatively you could cut cardboard squares. The student uses these squares to form rectangles and come up with areas and, later on, perimeters.

Diskette Two

3B The Sieve of Eratosthenes: This is an excellent treatment of the topic. The screen by actually crossing out all multiples of the most recently discovered prime. The student is asked for the first number to cross out and the computer does the rest. It is fun to watch.

In addition we learn when Eratosthenes lived and a little about Euclid as well. We learn about Twin Primes, Palindrome Primes, and even Goldbach's conjecture. (All even numbers can be written as the sum of two primes, one (1) may be used for this.) All in all, a math teacher's dream.

4A Exponents: This one very nicely teaches why exponents are useful by demonstration. If you properly factor 128, the program next asks if you want to try 16384. If you say "yes," the author is definitely up to the challenge. On the other hand, if the student repeatedly fails to get right answers, the program apologizes and returns to the menu. Clearly not threatening or intimidating at any level.

There is something in these programs for a wide range of students. Any teacher who has a computer which will support these programs and who teaches Factoring Whole Numbers will want to use them. □

Figure 1.

Diskette One
Introduction
1A Factor Pairs
1B The Rectangle Game
2A Pairs and Squares
2B Guess and Test

Diskette Two
Introduction
3A Primes and Composites
3B The Sieve of Eratosthenes
4A Exponents
4B How Many Factors

Diskette Three
Introduction
5A Highest Common Factor
5B The Euclid Game
6A Least Common Multiples
6B Factoring Finale

Arith-Magic

Problem Solving Programs For Older Students

Dan Isaacson

Whenever I go into a bookstore I always look for a new book by an author I have read before and liked. I am seldom disappointed. I'm finding that idea carries over into quality software, too. I liked Quality Educational Designs' earlier product, *Math Problems*, and, not surprisingly, I like their current effort, *Arith-Magic*.

There are three problem-solving programs in this set: Diffy, Tripuz, and Magic Squares. All give practice in mental arithmetic with addition, subtraction, multiplication, or division. All are motivational and different from most usual drills.

SOFTWARE PROFILE

Name: Arith-Magic

Type: Math

System: 32K TRS-80 with Disk Drive or 48K Apple, Applesoft, Disk Drive

Format: Disk

Language: Basic

Summary: Different and well done

Price: $35

Manufacturer:
Quality Educational Designs
P.O. Box 12486
Portland, OR 97212

Diffy

In Diffy, the student selects four natural numbers which are printed on the screen as the corners of a square. Then he finds the difference between each successive pair of numbers. The four differences become the corners of the next square. The student continues finding differences and creating new squares until the number in each corner is zero. The objective is to find four original numbers which will require as many moves as possible to reduce to zero.

A sample game asks for four numbers. The numbers are displayed like this:

```
   ?
3  27
13 31
```

What is the difference between the two numbers on each side of the "?"? The student is given two chances to respond correctly and then the correct answer is printed. The next frame would look like this:

```
   24
3  27
   ?
13 31
```

Diffy keeps track of the largest number of moves by any player so far, along with the player's name, which is an excellent motivational reward.

Tripuz

Tripuz asks the student to choose addition or multiplication for practice. The computer generates three numbers:

```
3  18  6
 24  48
   8
```

and asks for a value of the ? where the middle numbers (i.e. 24 and 18) are found as the product of the two numbers on either side. The final solution to this problem is:

At the secondary level, algebra students are encouraged to look for a general algebraic solution for both the addition game and the multiplication game.

Although addition and multiplication are the modes available, students must use the inverse concepts of subtraction and division to solve the puzzles.

Magic Squares

Magic Squares leads the student to an intuitive understanding of averages and arithmetic sequences. In 20 years of teaching I have never been motivated by magic squares, but now, after using the Magic Squares program, I have painlessly learned the concepts involved in designing a magic square; a big hand for QED.

Magic Squares provide an example of the power of computer simulation in problem solving. The computer eliminates computational drudgery and allows the user to concentrate on concepts and hypotheses.

Commentary

I can't end a review without commenting on programming style in general. Software producers, like students, learn by doing—each new program is better than the last from a style point of view, but it is the reviewer's job to keep producers

Dan Isaacson, School of Business, California State University, Fresno, CA 93740.

on their toes. I make the following observations hoping that future courseware for schools will show increased awareness of the psychological considerations in courseware design.

QED has correctly allowed users to press one key to mean a whole word (such as Y for Yes), but the program also accepts YED as YES. I suggest rejecting more than one character as a response, thus short-circuiting this problem.

Long, animated title pages (sometimes with sound) are great—the first time through. But users get exasperated with them if the program is used often. (Title pages seem to take longer and longer each time one uses them.) I've had comments on some programs like: "After the fifth time, I turned off the sound because it drove me nuts!" Title pages in these three games are slightly too long. It would be nice if a user could skip the title page after the first use.

Because everyone reads at a different speed, I would recommend that no message leave the screen until the user allows it to leave. In almost all cases, that's how QED has programmed. Several times, though, a praise message is not under user control and leaves the screen unexpectedly, which jars the user.

It is desirable that users be able to page backward as well as forward in a tutorial. We all have read a paragraph and then realized that we didn't remember a thing in it. A book allows us to go back and re-read it. Many educational programmers have not recognized that necessity yet. I felt a desire, several times in Magic Squares, to return to a previous frame—which I couldn't do. Fortunately, later on, the program cycled through those frames I wanted, but I was temporarily frustrated as I couldn't get the help I needed exactly when I needed it.

As a final note, I found no way to exit from the program menu without turning off the computer and re-booting. This is an understandable consequence of the author's attempt to protect his copyright and prevent piracy, but it makes programs less user-friendly. I hope this problem is shortly overcome.

Now if programmers have heeded my concerns, prospective purchasers may ignore this reviewer's wishful suggestions and consider buying a copy of Arith-Magic. I, for one, look forward to finding more, new releases from QED at my friendly, neighborhood computer store. □

muSimp/muMath-80

A Symbolic Mathematics System

David D. Shochat

SOFTWARE PROFILE

Name: muSimp/muMath-80
Type: Symbolic mathematics programs
System: CP/M, Apple, TRS-80 48K
Format: Disk
Language: Machine
Summary: "Lisp-like" math language
Price: $250
Manufacturer:
Microsoft
10700 Northup Way
Bellevue, WA 98004

The following is a revision of a review of muSIMP/muMATH-79 which originally appeared in SIGPC Notes vol. 4, no. 1/2.

David D. Shochat, Dept. of Mathematics, Santa Monica College, 1910 Dico Blvd., Santa Monica, CA 90405.

About 20 years ago, I had the opportunity to see a real computer, an IBM 709. When I asked whether it could do calculus, I was told that a digital computer could do numerical differentiation and integration, but that symbolic manipulation of the function definition was inherently outside the domain of the computer.

The muMath software package, developed by Albert D. Rich and David Stoutemyer of the Soft Warehouse and available through Microsoft for 8080/Z-80 disk systems (and now Apple II as well), can do a surprising amount of calculus—symbolically, the way you learn to do it in a calculus class. For example, *muMath* can integrate

```
(1 + X^2)^(1/2)
```

in about seven seconds (on my 2 MHz Z-80 CP/M system). *muMath* can also do exact rational arithmetic, with (up to) 254-byte integers in any number base to 36, a great deal of algebra, some symbolic trig and even matrix algebra: *muMath* will find the inverse of

```
{[1, 2],        or of        {[a, b],
 [3, 4]}                      [c, d]} .
```

Getting all of this to work on a microcomputer with only 64K (or less) is impressive, to say the least. The secret is modular construction. You load only part of *muMath* at a time. It is unlikely, for example, that you will need to do matrix inversion and integration at the same time. The precious memory resources must be used as efficiently as possible, a fact which reflected in the ritual of building and saving the system.

To get started, execute MUSIMP.COM. The muSimp interpreter is loaded in, a sign on message is printed, and the muSIMP "?" prompt appears.

muSimp is the Lisp "surface language" in which *muMath* is written. You may respond to the prompt in several ways.

1. You may make an assignment:

```
S:  13;        (cr)
```

muSimp's response here would be:

```
@: 13
?
```

2. You may use an existing function:

```
D: S * 4 ;        (cr)
@: 52
?
```

3. You may define a new function. (Every line of user input is terminated with a carriage return):

```
FUNCTION COMBS (N, K),
  WHEN K = 0, 1 EXIT,
  N * COMBS (N-1, K-1) / K
ENDFUN $
? COMBS (D, 5);
@: 2598960
?
```

The definition of COMBS, which computes the number of combinations of N things K at a time, shows that recursion is allowed.

```
LISTEXAMPLE: '((1 ONE) (2 TWO)
  (3 THREE)) &
@: ((1 ONE) (2 TWO) (3 THREE))
? FIRST (LISTEXAMPLE) &
@: (1 ONE)
? REST (LISTEXAMPLE) &
@: ((2 TWO) (3 THREE))
? REST (FIRST (LISTEXAMPLE)) &
@: (ONE)
? REST (REST (FIRST (LISTEXAMPLE))) &
@: FALSE
```

(the name FALSE is equivalent to the empty list)

```
? FUNCTION MAPFIRST (LIS),
  WHEN EMPTY (LIS), FALSE EXIT,
  ADJOIN (FIRST (FIRST (LIS)),
    MAPFIRST (REST (LIS)))
  ENDFUN $
? MAPFIRST (LISTEXAMPLE) &
@: (1 2 3)
```

The use of the "&" as a termination symbol instead of ";" indicates to the DRIVER function, which contains the main interaction loop, that the result of the computation should be printed as a list, rather than as a mathematical expression in standard mathematical notation. A "$" means the result should not be printed at all.

```
EXPRESSION: X + Y $
? EXPRESSION ;
@: X + Y
? EXPRESSION &
@: (+ X Y)
```

Internally, only the latter (list) form really exists. But this is almost totally transparent to the user. The DRIVER function mentioned above gets its input through a function called PARSE, which normally expects all of *its* input to be in non-list notation. It is this PARSE function that really defines the difference between muSimp and normal Lisp.

For example, if PARSE sees FUNCTION in the input stream, it translates everything from there to the next occurrence of ENDFUN into a list, which is essentially the Lisp equivalent of that function definition.

In Lisp, everything is a list, including function definitions. In muSimp, PARSE translates both standard mathematical notation, and MuSimp syntax itself into list/Lisp form. So the difference between muSimp and normal Lisp exists only "at the surface."

When PARSE sees a single quotation mark, two things happen. First, it reads what follows in normal list notation. Second, it adds the QUOTE function name onto the expression, which causes the expression to be "taken literally," rather than being evaluated. Thus in the definition of LISTEXAMPLE above, even if we had previously made an assignment

```
ONE: UNO $ ,
```

LISTEXAMPLE would still contain only ONE, rather UNO.

Once a function definition has been made, it becomes a part of the system, which can then be called upon by other function definitions. And that's how you bring *muMath* into the picture, because new function definitions (and other kinds of valid muSimp input), can be read in from disk as well as from the keyboard. For example, if you type:

```
RDS (ARITH, MUS);
```

with the source file ARITH.MUS on the disk in drive A, all the muSimp source code constituting ARITH.MUS will be read in and made a part of the system just as COMBS and MAPFIRST were. Now you can do:

```
1/2 + 1/3 ;
@: 5 / 6
```

muMath is a collection of 15 muSimp source files, all but one of which depend on certain other source files being "in" at the same time. To do definite integrals, for example, you must load in five of these source files.

Any time you want you can save the state of the system on disk including (in internal form) all the muSimp code read in so far, just by typing

```
SAVE (<name>) $
```

(with a disk with enough free space in drive A), where <name> can be any available primary file name for your DOS. muSimp will create a special memory image file with primary name <name> and secondary name SYS. Then you can come back any time and just by typing (while in muSimp):

```
LOAD (<name>) $ ,
```

you will be right back where you were just before the SAVE. Or, another way to do it is to type:

MUSIMP <name>

as a DOS command, in which case muSimp will LOAD <name>.SYS for you before entering the DRIVER loop.

The package comes with ten other source files which are 90% comments. These are the lesson files, which are so well-written they make the process of learning to use the system a great deal of fun. The lesson files are read in like any muSimp source file, but they are echoed at the console and consist mostly of comments, delimited by matching percent signs. Every once in a while, the lesson gives you an exercise to do, ends the comment with a "%", and then makes the assignment:

RDS: FALSE $

That stops the input from the disk and gives you control of the system to do your exercise. When you're done, you type:

RDS: TRUE $

and the lesson takes over again. The lesson may also stop being a comment long enough to do an example, providing actual muSimp source to the system. This example may, in turn, depend upon function definitions which you added to the system while doing a previous exercise.

It all works beautifully as long as you don't try to stop in the middle of a lesson and save what you have done so far. It is possible, but tricky: you must refer to the printed copy of the lesson to see what "finishing up" operations it would have done, do them through the keyboard, and finally SAVE a memory image as usual. When you come back, you will still have to run through the lesson file from the beginning, but now you can skip the exercises you have already done.

Five of the lessons teach the use of those features introduced to the system by the ARITH and ALGEBRA files of *muMath*. After completing them, you can use the other *muMath* files without much difficulty, with the aid of the printed documentation.

The remaining five lessons teach the muSimp language — to a point. The trouble is, they are so beautifully written, you will feel somewhat lost when, after the fifth lesson, you must start learning from the printed documentation, which is much more terse than the lessons. The lessons really "take you by the hand"; the printed material works in a way which is actually similar to the way the system is structured internally: defining itself in terms of itself.

It was a frightening experience at first, trying to learn from the printed discussion of muSimp, but I finally got used to it. After a year and a half, I think I can appreciate the concise style.

The documentation has been improved considerably, since the original muSimp-79 version, but there is still room for improvement. The function READ is not documented at all (probably an oversight).

Considerable information is included about the various property lists used by PARSE, but the definition of PARSE itself is missing—I had to dig out the internal (Lisp) form to see how it works. And it is only by seeing how PARSE works, that some of the documentation which *is* included becomes intelligible.

muMath can handle a wide variety of symbolic math problems, so whenever it fails to come up with an expected solution or simplification, it leaves you wondering why. The only way to find out *why* is to delve into the source files themselves and uncover the methods used by *muMath*. I've also found this to be one of the best ways to learn about important techniques which are not discussed in the lessons or in the printed documentation (such as storing function definitions on property lists).

Another interesting approach is simply to experiment with *muMath*. In the examples which follow, the symbol " → " will mean "simplifies to."

8^(1/2) → 2^(3/2),

but

343^(1/2) doesn't simplify.

This is because ARITH works with a very short list of primes: (2 3 5). If you reassign:

PRIMES: '(2 3 5 7) $

then,

343^(1/2) → 7^(3/2).

Some complex arithmetic is possible (# I is i):

(1 + #I)^2 → 2*#I,

but

(2*#I)^(1/2)

appears to be too much to ask, even with both trig files loaded.

With the LOG file loaded,

LOG (2^X, 2) → X,

(the second argument is the base), and

LOG (8, 2) → 3

(an improvement in the latest version).

muMath forces the user to do a certain amount of crucial decision-making by setting what are called control variables. For example, if the control variable NUMNUM has the value 2,

2 * (X + 1) → 2 + 2*X,

and if NUMNUM is -2,

2 + 2*X → 2 * (1 + X).

If NUMNUM is 30,

(X + 2) * (X + 3) → 6 + 5*X + X^2,

but, and this is perhaps the biggest weakness of *muMath;* it cannot factor even a simple quadratic such as the one above. And yet, with the equation-solving files loaded,

SOLVE (X^2 + 5*X + 6 = = 0, X) →

{X= =-2,

X= =-3} ,

even though the former task is easily reducible to the latter. The manufacturer says (in one of the lessons) that they are working on it for future releases.

You would expect:

X/(X*Y) → 1/Y,

but one day I couldn't seem to make it happen. I finally realized that the problem lay with the control variable EXBAS which needs must be a positive multiple of 2 in order for the cancellation to work. Now EXPBAS being a positive multiple of 2 is supposed to allow such things as:

(X*Y)^2 → X^2 * Y^2.

In order to make sense out of all this you must realize that internally, a fraction A/B is equivalent to A * B^-1 (actually the list: (* A (^B -1))), so the X*Y in the denominator of the cancellation example is really a factor of the form: (X*Y)^-1, which explains why the state of EXPBAS is so crucial, in a problem which seems to have nothing to do with exponents.

The control variable DENNUM must be a negative multiple of 15 in order to get:

1/X + 1/(X + 1) →

(1 + 2*X) / (X * (1 + X)),

but then DENDEN must be a positive multiple of 3 if you want the denominator multiplied out, and so it goes.

Guiding *muMath* through a tricky problem of adding rational expressions sometimes seems to take as much skill as doing the problem yourself. Of course, *muMath* is especially impressive with a really tedious problem such as (1 + X)^20 (PWREXPD must be 2).

I have found the trig and calculus capabilities of *muMath* to be surprisingly good: *muMath* will differentiate just about any function you give it, since it knows all the standard rules. Integration is, of course, the real test.

```
INT (1/(X^2 + 5*X + 6), X)→

LN ((-4 - 2*X)/(6 + 2*X)),
```

which can be further simplified only by a subsequent re-evaluation of the answer, with NUMNUM and DENDEN set to -2. A careful study of the two integration source files reveals that the above problem is done by completing the square. Since *muMath* can't factor quadratics, there is no way it could do the problem by partial fractions. But it is clear, even without examining the source files, that *muMath* really doesn't know anything about partial fractions, since it can't integrate

```
1 / (X^3 + 1),
```

even if you do the factoring for it. *muMath* appears to be able to do integration by parts, successfully integrating such things as

```
X^2 * #E^X
```

(#E is e), but if you look at the source code, what you find is that *muMath* knows many of the special reduction formulas, which one normally derives using parts. Surprisingly, *muMath* can't integrate

```
X * (1 + X)^(1/2),
```

which is easy using parts.

The latest version of *muMath* (MuSimp/muMath-80) includes three source files which were not in the previous version: TAYLOR.DIF, LIM.DIF, and SIGMA.ALG.

The file TAYLOR.DIF, which consists of a single function definition, generates Taylor polynomials. For example,

```
TAYLOR (TAN (X), X, 0, 5) →

X + X^3/3 + 2*X^5/15 .
```

Expansions about points other than 0 are "multiplied out" whether you like it or not. This turns out to be inherent in the algorithm used, but one can easily define:

```
FUNCTION TAYLOR1 (EX, VAR, POINT,
  ORDER) ,
  EVSUB (TAYLOR (EVSUB (EX, VAR,
  VAR+POINT), VAR, 0, ORDER), VAR,
  VAR-POINT)
 ENDFUN $
```

Now,

```
TAYLOR1 (LN (X), X, 1, 5) →

-1 + X - (-1+X)^2/2 + (-1+X)^3/3

-(-1+X)^4/4 + (-1+X)^5/5.
```

LIM.DIF, as its name suggests, calculates limits, using the differentiation machinery of *muMath*. For example:

```
LIM ((X-SIN (X)) / X^3, X, 0) →

1 / 6 .

LIM ((1 + 2*X)^(3/X), X, 0) →

#E^6 .

LIM (LN (X) / X, X, PINF) →

PZERO .
```

In the first two examples, the limit is understood to be from the right (otherwise an additional argument to LIM must be used). The last example shows the use of PINF, to denote positive infinity, and PZERO, to indicate that the limit of 0 is approached through positive values. If we ask *muMath* to do a more general example, e.g.

```
LIM ((1 + A*X)^(B/X), X, 0);
```

a curious thing happens: *muMath* starts asking for additional information.

```
@:

??? A ???

ENTER SIGN (0 + -)?
```

A single character response of "+" is accepted and the computation immediately resumes, finally producing the expected answer,

```
+E^(A*B) .
```

As you would guess, the file SIGMA.ALG does summations (and products too). For example:

```
SIGMA (K^2, K, 1, 100) →  338350

SIGMA (1/K^2, K, 1, 20)→

17299975731542641 / 10838475198270720
```

In some cases, it can even handle a sum with a variable number of terms:

```
SIGMA (K^2, K, 1, N)→

(1+N-3*(1+N)^2+2*(1+N)^3) / 6

SIGMA (1/2^K, K, 0, N-1) →

(-2+2^(1+N)) / 2^N .
```

Incidentally, if you want the result in another form, you must *re*-evaluate the answer with appropriate flag settings. For example, the above result was obtained even with DENNUM = 3. But then,

```
EVAL (@) →  2-2^(1-N)
```

(The atom @ is always bound to the result of the previous computation.) If LIM.DIF is loaded along with SIGMA.ALG, the prospect of infinite series arises. As far as I can tell, this is limited to geometric and telescoping series (we can't expect miracles). Also, its best *not* to have LIM.DIF loaded unless you need it, as it interferes in an odd way with the finite sum activity of SIGMA, e.g. asking (twice!) for the sign of -LN(2) during the evaluation of a finite geometric series with common ratio 1/2.

When I first started in with *muMath,* it was the symbolic math which fascinated me most. And it is indeed an amazing accomplishment, especially considering the limited memory space. Also, I understand that there are problems in physics and engineering which can be rendered tractible only by using a combination of numerical and symbolic methods.

I really don't know how useful *muMath* would be to a person who doesn't already know the relevant mathematics. I would love to see what would happen with *muMath* in the hands of my beginning algebra students, but I haven't yet had the opportunity. I really believe a person could learn some mathematics just by learning to use *muMath,* but this is only speculation.

What excites me most now is the muSimp language itself. Programming in muSimp involves using, in a very concrete way, the same techniques of inductive definition, building complex structures from simple ones, and then analyzing and operating on those structures inductively, that are fundamental to theoretical work in set theory and logic. It also turns out to be much more useful as a general purpose language than I originally suspected, particularly now that you can easily link muSimp to your own machine language routines.

muSimp is also an excellent bridge between "traditional" programming languages and Lisp. Based on my own experience, I think it would be very easy and natural for anyone accustomed to Pascal or PL/I to start right in with muSimp, but then when he subsequently (I would say inevitably) discovers Lisp, it will seem natural too. I feel, however, that any further discussion of the muSimp language should be conducted in the context of Lisp itself and Lisp surface languages generally.

The *muMath* package is certainly an impressive piece of work. Although I am particularly interested in its potential as a teaching tool, it will undoubtedly find uses in other areas as well. But at any rate, the time I have spent with *muMath* and muSimp has been challenging and a great deal of fun. □

TRS-80 MicroPilot

Writing Courseware on the TRS-80

R. Reed Hardy and Eliot S. Elfner

After using Radio Shack's *MicroPilot* (formerly known as Pilot Plus) to write about 5000 lines of tutorial software, we feel qualified to praise and criticize this version of Pilot.

Pilot has become the generic name for a programming language that is especially well suited for authoring and executing CAI educational programming. There are now several versions of Pilot in fairly common use, each of which has its own strengths and weaknesses.

This review is not intended to be a complete comparison of *MicroPilot* with other versions of Pilot. It is simply a description of the authors' experience with *MicroPilot*.

Let me begin by praising Radio Shack for their initiative in developing *MicroPilot* and thanking them for allowing us to use a pre-publication version so that we could begin authoring programs at least six months earlier than would otherwise have been possible.

Dr. R. Reed Hardy, Associate Professor of Psychology, St. Norbert College, De Pere, WI 54115.
Dr. Eliot S. Elfner, Associate Professor of Business Administration, St. Norbert College, De Pere, WI 54115.

Strengths

The strengths of *MicroPilot* lie in its efficiency for educational/interactive programming. With *MicroPilot* you can ask a question, get an answer, evaluate the answer and appropriate feedback with three or four simple commands. You can

SOFTWARE PROFILE

Name: TRS-80 MicroPilot
Type: Programming Language
System: TRS-80 Model I, III, 32K
Format: Disk
Language: Machine
Summary: Easy and direct route to custom C.A.I. software
Price: $79.95
Manufacturer:
Radio Shack
1800 One Tandy Center
Ft. Worth, TX 76102

keep track of a student's progress so that you can automatically start him where he left off when he logs onto the system. All of these characteristics are common to most forms of Pilot.

The features that make *MicroPilot* shine are its edit/run capability, its graphics, its sound generation capability, and its record keeping commands.

There is also a graphics screen generation utility (Graphics/QLT) that allows the user to "draw" a screen and store it on disk for use in programs. These screens are developed during both programming and execution, much more quickly than screens developed using graphics commands in most cases. (Note: this utility will not run on machines without lower case capability.) This screen generation utility can also be used to allow the student to "draw" a screen which can then be saved for later evaluation or simply dumped.

Another *MicroPilot* strength is the editor that is included with the software. The editor is essentially the same as that used for Level II Basic which makes it very convenient for those who are accustomed to using Radio Shack Basic.

During program authoring, *MicroPilot* allows the author to run a program starting with any specified line number, execute any valid *MicroPilot* command directly, break into a program and then reenter the program where it was interrupted by typing "continue." Lines and characters can be edited with ease.

There are other strengths that those who aren't familiar with other versions of Pilot may not appreciate. For example, there are commands to clear the screen and home the cursor (N:), delay program execution for any specified number of seconds (D:N), pause until a key is pressed (W:), and print text on the printer instead of the screen (P:).

These are functions that can be accomplished with other versions of Pilot by using ASCII codes or other roundabout methods, but *MicroPilot* makes their use a breeze. Another advantage relative to other versions of Pilot is that the author can run a program without loading special execution software. This makes editing and general debugging much faster.

Given all of the above praises, you may be beginning to think that *MicroPilot* is nearly perfect. Well, don't run out and buy it without finishing this article. Along with its strenths, *MicroPilot* has several significant weaknesses.

Weakness

Probably the single most damaging weakness in *MicroPilot* is the fact that the student user has very easy access to the program. Any student with elementary programming ability can list a program, change it, and save it with little difficulty.

These operations can be done only while the user is in "command" mode, and when a *MicroPilot* program is running, the student would not normally be able to do any of the above dastardly deeds. However, there are several ways a student can find himself in command mode. The following is a list of some of these:

1) Since the Model I version of *MicroPilot* does not come with an Initial Program Load utility (I.P.L. utilities automatically begin the execution of a specified program when the computer is turned on), the student must either be supervised during initial program load or he must load and run his own program from the *command mode*.

2) Since *MicroPilot* has no ON ERROR GO TO command or its equivalent, *any error* (e.g., the student enters an O when the program expects 0, or makes an error in giving his log-on file code) will cause the program to crash and come down in *command mode*.

3) In addition to the above problems, there is no easy way to disable the Break key which halts execution and puts the user in *command mode*.

Before *MicroPilot* can be considered a serious educational tool, this easy access to command mode must be corrected. In other versions of Pilot, this has been done by separating the program execution software from the authoring software. This effectively prevents the listing or modification of programs by the student. If a program crashes under such circumstances, the student user can usually do only one of two things; 1) seek help from an assistant, or 2) type run and start again. This is as it should be.

As things stand with *MicroPilot*, a reasonably competent programmer could list test items and, by interpreting the program, get the correct answers before taking the test. An even more insidious possibility is that the student could directly modify his disk file indicating perfect performance on all unit quizzes without ever going through the course. This characteristic of *MicroPilot* makes it unsuited to some CAI applications.

While the easy access to command mode is the major problem with this version of TRS-80 *MicroPilot*, there are a few other software characteristics that make for programming difficulties:

The programmer cannot POKE a value into a memory location in MicroPilot, nor can he call and execute a machine language program.

There is no way to check the DOS Directory from *MicroPilot*, or to specify the drive number when loading and saving programs and/or student records.

The documentation of the record keeping command is so unclear and incomplete that the programmer literally has to find out for himself by trial and error how the commands work and what they do. (Note: *Always* be sure you close a file immediately after reading or writing, or you could clobber the directory on your disk and lose access to all the programs on that disk.)

The general documentation that is sold with *MicroPilot* assumes no prior programming experience on the part of the user. Thus, it is very awkward and inefficient for the user who already has a programming background.

A disadvantage of *MicroPilot* relative to Basic is that it is limited to 32K, disk configurations of the TRS-80 Model I, or Model III. Thus, those with 16K, level II machines cannot use courseware developed in *MicroPilot*. However, courseware programmed in Level II Basic could be used in such configurations.

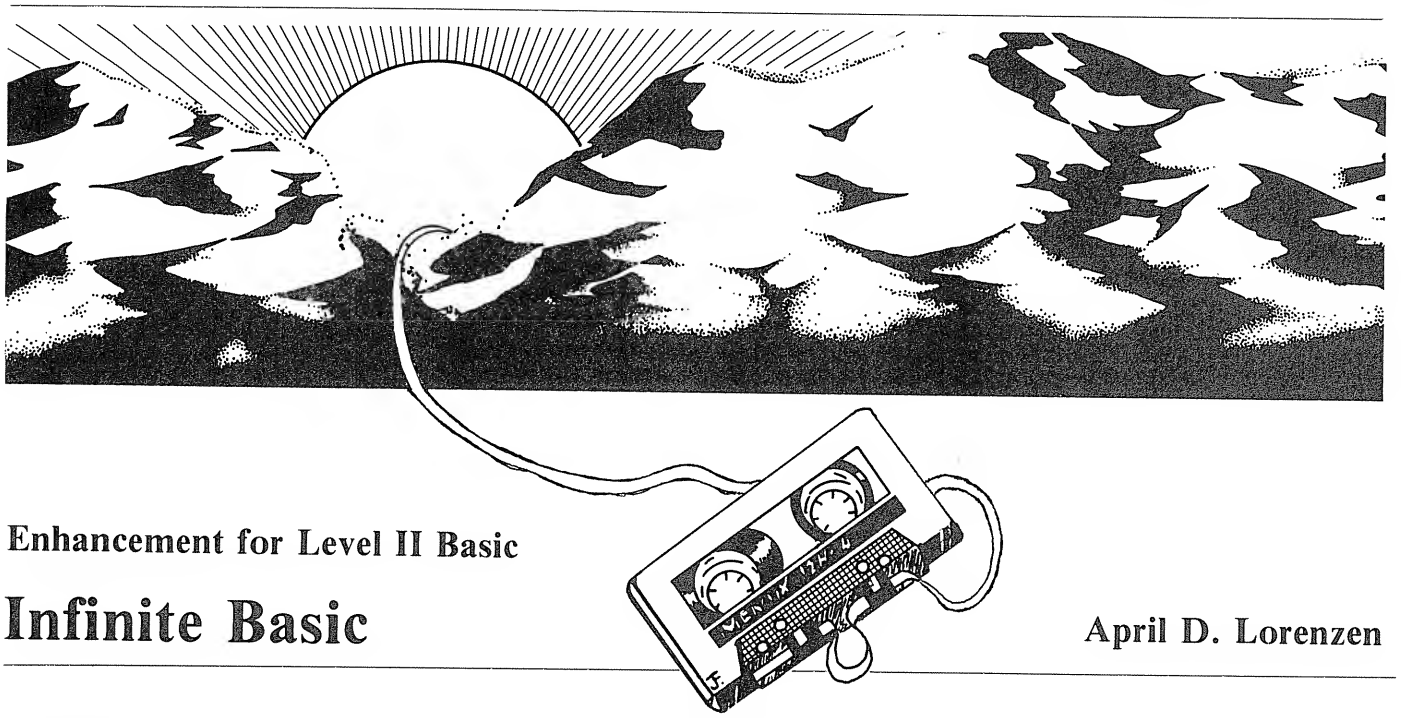
MicroPilot vs. Basic

Some comments are appropriate on the relative merits of *MicroPilot* and Basic as CAI courseware authoring languages. Basic is a full-fledged, flexible, programming language, capable of a wide variety of programming applications, including the development of CAI courseware. However, mastering Basic requires a great deal of programming skill, and expertise.

MicroPilot, on the other hand, can be implemented in simple applications quite easily. Almost anyone wishing simply to present some material, and check for understanding, could pick up the *MicroPilot* syntax quickly. However, taking advantage of the complete range of *MicroPilot* commands, and linking contingent feedback and progress skillfully, as required in good CAI courseware,

demands the same level of ability and skill required of a Basic programmer. It seems that intricate CAI courseware would be equally difficult to program with either *MicroPilot* or Basic.

So if you are already a hot-shot Basic programmer you can probably get along without *MicroPilot*, but if you are expecting to limit your programming to educational and especially tutorial programming, you will probably find Radio Shack TRS-80 *MicroPilot* the easiest and most direct route to writing excellent software. □



Enhancement for Level II Basic

Infinite Basic

April D. Lorenzen

Frustrated by the limitations of Level II Basic? *Infinite Basic* from **Racet Computes** may be your saving grace! Unlike most other enhancements to Level II, *IB* takes up very little memory because you create a specialized load module containing only the function routines necessary for a specific program. The module can be located anywhere in memory and saved on the same tape as your program. The machine language routines are activated by executing a ?USR(1) statement and deactivated by executing the *Infinite Basic* command &NOIB.

Matrix functions include reading or writing arrays to tape without leaders between elements, redimensioning and deleting arrarys with no loss of data, reading data statements into arrays, and copying or transposing one array into another as well as scalar, element, and matrix addition, subtraction, multiplication, division, solutions to simultaneous linear equations, Basic subroutine calling, and subroutine return value facilities are all part of the package and individually selectable.

Extensive string manipulation routines include justify, rotate, shift, and truncate left or right, search a string for a substring, delete or insert a substring, and truncate left or right, search a string for a substring, delete or insert a substring, and the verify routine which compares two strings and returns the location of the first discrepancy. You can also create an absolute string pointer, convert any string or numerical value to a hexadecimal string, perform a character string or multivariable sort, PLUK or PLUG a two byte word into memory, propagate a byte consecutively in memory, and compress or decompress bytes to four, five, six, or seven bit formats.

Infinite Basic allows you to draw and erase horizontal and vertical lines at high speeds and control screen scrolling up, down, left, and right with more handy routines.

Many functions could be used in text editing such as the text pack function which deletes spaces from the ends of strings and leaves words separated by not more than one space. A character other than a blank space can be specified and will be deleted instead. Centering and justifying text, generating random strings, and converting to upper or lower case at will is also possible.

The routines mentioned above and several more are fully explained with program examples in eighty-four pages of excellent documentation. I received my copy of *Infinite Basic* less than two weeks from the day I mailed my order. It comes on a single cassette with the tape version on one side and the disk version on the other. I had no trouble loading it on the first try. The only criticism I have is that the terminology used in the manual may not be familiar to the average TRS-80 programmer. A glossary of terms would be a definite advantage. Perhaps David Lien will write a book entitled *Learning Infinite Basic* for all us novices!

Infinite Basic is easy to use, well documented, has greater capability and executes faster than Basic subroutines in less memory, and appears to have unlimited applications. An add-on module (*Infinite Business*) is available and more modules will be introduced in the future. Both are available for 16K to 48K cassette and disk based TRS-80's at the very reasonable price of $59.95 (*Infinite Basic*) and $29.95 (*Infinite Business*) from Racet Computes, 702 Palmdale, Orange, CA 92665. □

April D. Lorenzen, Route 2, Box 44, Canton, Kansas 67428.

Pascal for the TRS-80

David E. Powers

Some observations on implementing Pascal on any micro, with emphasis on the version from FMG for the TRS-80.

Three hundred people squeezed together for Peter Kugel's overview of Pascal, among the best attended seminars at the New York National Small Computer Show in August. Participants included aficionados of Cobol, Basic, Fortran, assembler and a few other languages, yet all shared interest in Pascal. Professor Kugel convinced the crowd that Pascal has emerged as a fundamental language, one serious programmers must learn. The Federal government has approved the development of ADA, a Pascal-based language, for use at all Federal computer installations. Just to stay abreast, professionals need a working knowledge of Pascal. Also, as more instructors employ it in their classrooms, more programmers will use it in their work.

Why Pascal?

What about computer hobbyists and other small system users? Among us brews a puzzling controversy over the relative merits of Basic and Pascal, yet Pascal is faster, more logical and substantially more flexible. Friends of Basic observe that it is entrenched. They argue that it is easy, but so is Pascal, and there is nothing Basic can do that Pascal cannot. The reverse is untrue. Although considerably more sophisticated than Basic, Pascal's elegant use is easier to learn, and it provides greater power for advanced programmers. Users grow with Pascal far more than they can with Basic.

David E. Powers, 10 Wilben Ct., New Hyde Park, NY 11040.

```
 1      PROGRAM FACTROOT;
 2      CONST EPSILON=1E-5;
 3      VAR LOWLIMIT, HIGHLIMIT, LOOPCOUNT: 1..20;
 4        HOLD: REAL;
 5      FUNCTION FACTORIAL (VALUE: INTEGER): REAL;
 6      VAR LOOPCOUNT: 1..20;
 7        PRODUCT: REAL;
 8      BEGIN
 9        PRODUCT:=1;
10        FOR LOOPCOUNT:= VALUE DOWNTO 1 DO
11          PRODUCT:=PRODUCT*LOOPCOUNT;
12        FACTORIAL:=PRODUCT
13      END; (*FACTORIAL*)

14      FUNCTION NEWTON (START: REAL): REAL;
15      VAR SQROOT: REAL;
16      BEGIN
17        SQROOT:=1;
18        REPEAT
19          SQROOT:=(SQROOT+START/SQROOT)/2
20        UNTIL ABS(START/SQR(SQROOT)-1)<=EPSILON;
21        NEWTON:=SQROOT
22      END; (*NEWTON*)
23      BEGIN
24        WRITELN('FACTORIAL AND ROOT COMPUTATION.');
25        WRITELN; WRITE('ENTER LOW LIMIT (0<LOW<21)   ==> ');
26        READLN (LOWLIMIT);
27        WRITE ('ENTER HIGH LIMIT (1<HIGH<21) ==> ');
28        READLN(HIGHLIMIT);
29        FOR LOOPCOUNT:=LOWLIMIT TO HIGHLIMIT DO
30          BEGIN
31            HOLD:=FACTORIAL(LOOPCOUNT);
32            WRITELN(LOOPCOUNT:10, HOLD:10, NEWTON(HOLD):10)
33          END;
34      END. (*FACTROOT*)
```

Listing 1
FACTROOT, a Pascal program to find factorials and their square roots.

Niklaus Wirth developed Pascal to teach computer novices structured programming, sometimes described as coding without "GOTO." True, structured programs rarely need "GOTO," but the characterization is an oversimplification. One writes a structured program in "top-down" fashion, solving overarching problems first, postponing attention to detail, then breaking solutions into progressively smaller problems, solved individually in "stepwise refinement." Basic is oriented to lines rather than to logical units. Programs move through algorithms vertically. Pascal is block-structured; programmers build and manipulate whole ideas at once.

The University of California at San Diego has implemented a version of Pascal suitable for microcomputers, yet considerably more powerful than a "tiny" Pascal that was not designed to meet Nicklaus Wirth's specifications, published in the **Pascal User Manual and Report.** I was delighted to learn that the FMG Corporation of Ft. Worth offered a release of UCSD Pascal for the TRS-80. It conforms well to Wirth's standards.

Speed is the most immediately impressive feature of Pascal on the TRS-80. A Basic interpreter must translate each Basic statement into

machine language instructions every time a program runs, but a Pascal system compiles statements into object code only for the first execution, then saves the low level code (called P-code) for future runs. Pascal yields faster execution, often by a factor of four or more.

Listing 1, a Pascal program (inspired by one in Peter Grogono's **Programming in Pascal**), calculates factorials in a user-selected range from 1 to 20 and subsequently derives square roots of the factorials using Newton's iterative method (if R is an approximation of the square root of a positive number, N, then (R+(N/R))/2 is a closer approximation), within an error of 0.00001. Listing 2 performs the same task in Basic. I did not write either routine for efficiency, but deliberately multiplied number crunching to test speed. Also, I tried to prepare Basic code that would emulate the readability of Pascal, yet remain within the syntactic bounds of TRS-80 disk Basic. With a range of 1 to 20, the Basic program runs for 35 seconds. The Pascal version does the job in less than 12 seconds. Basic completes a range of 16 to 20 in 16 seconds; Pascal, in four. Imagine the consequences for long programs.

Note the Pascal program's clarity. The language is self-documenting, particularly since it allows long distinct variable names. Indentations demonstrate graphically the logic of program units. The heart of the Pascal version of "factroot" is the last third, the largest logical structure, beginning at line 23. (Pascal programs do not customarily have line numbers; the numbers in Listing 1 are for reader convenience.) It prints instructions (lines 24, 25 and 27), and reads information from the keyboard (lines 26 and 28). Next, the program enters a loop (line 29-33) which calculates factorials in the selected range (line 31) and displays the results along with the square roots of the results (line 32). The program calls the functions "Factorial" and "Newton" and passes to them the parameters "Loopcount" and "Hold."

Function "Factorial" (line 5) yields a real result. "Loopcount" is a local variable (line 6); it is not the same "Loopcount" as in the main program. Also, the program allows "Loopcount" a value range from 1 to 20; any value outside that range will cause abnormal termination. The function declares a variable, "Product," to hold a real number. The loop (lines 10-11) runs until the factorial is calculated, then returns the value of "Product" to the calling point in the function name, "Factorial" (line 12).

"Newton" (line 14), the square root calculation sequence, accepts the value passed in the real parameter, "Start." The variable "Sqroot" is declared as real (line 15) and a repeated statement calculates the result iteratively (lines 18-19), terminating when the test in line 20 is satisfied.

The beginning of the program contains its name and a declaration of a constant, "Epsilon," the error allowed in the "Newton" function. The program allots a subrange of integers from 1 to 20 to variables "Lowlimit," "Highlimit" and "Loopcount." "Hold" is declared as a real variable.

Listings 3 and 4 (of a trivial program called "Reverse") compare relative string manipulation speed. Either version will accept a typed-in string, reverse its order, display it and repeat the process 100 times. Try Listing 4 on a TRS-80, using as an input string the English alphabet typed three times in succession. Basic will require 133 seconds to complete the task, but Pascal finishes the job in 27 seconds.

"Factroot" and "Reverse" are very simple programs, intended only to illustrate structure, readability and speed. Any Basic programmer can produce programs like "Factroot" and "Reverse" after minimal introduction to Pascal. With a little effort an experienced user of Basic should write meaningful Pascal software very readily.

Language Characteristics

Among Pascal's strengths are flexible manipulating and structuring of data. UCSD Pascal predeclares five variables types: integer (whole numbers from -32768 to +32767), real (which may have fractional parts), character, string and Boolean. Character variables compare to one-letter string literals in Basic, and strings in UCSD Pascal are similar to strings in Basic. Boolean variables are logical variables; their values may be only true or false.

Pascal also permits creation of new variable types, often as subranges of other types. For example, the declaration in line 3 of Listing 1 places "Lowlimit," "Highlimit" and "Loopcount" within the subrange of integers from 1 to 20. Any attempt to assign a non-conforming value will cause a runtime error.

A declaration like

```
TYPE DAY = (SUNDAY, MONDAY,
  TUESDAY, WEDNESDAY, THURS-
  DAY, FRIDAY, SATURDAY);
```

creates a new variable type called "Day." A programmer may then declare variables of the new type, even as subranges in variable statements like

```
VAR WORKDAY: MONDAY . . . FRIDAY
```

which defines a subrange for "Workday." If the program, otherwise unaware of workers' sensitivities, should assign Saturday or Sunday to the variable, "Workday," the computer would detect the error at once.

It is also possible to declare variable types as sets or arrays or as records, single variables into which a programmer may group distinct variable types. For example, the following declarations establish a record type "Customer" which contains the customer's account number, his name, his account balance, whether he is a residential customer, and what appliances he has purchased:

```
TYPE APPLIANCE = (STOVE, DISH-
  WASHER, REFRIGERATOR,
  AIRCONDIT, COMPACTOR);
TYPE CUSTOMER =
  RECORD
    ACCOUNT: INTEGER;
    NAME: STRING;
    BALANCE: REAL;
    RESIDENTIAL: BOOLEAN;
    PURCHASES: SET OF APPLIANCE
  END;
```

Programmers may access individual fields by name or manipulate whole records in interesting ways, placing them into files, arrays, sets or other records, creating ever more complex and flexible data types.

Most functions and procedures native to Basic are in Pascal in some manner, although usually in improved fashion. There is even a limited GOTO. Omissions of popular features (such as default type declarations, automatic conversions, an exponentiation operator) were deliberate, to serve the need of good programming style. Many operations, functions and procedures in Pascal are new to Basic programmers. Table 1 describes some of the UCSD Pascal features. Those marked with an asterisk are UCSD extensions, not standard in all implementations.

String handling features comprise one important set of UCSD extensions. They are more powerful than the routines in TRS-80 disk Basic. Users may address or alter individual string elements in single operations because the string is a character array. For example, the fifth element of "ABCDEFG" may be referenced as ST[5], assuming that the variable name "ST" were assigned to the string. The procedure INSERT allows groups of characters to be inserted at any string

position, without breaking and reconcatenating the string, as in Microsoft Basic. For instance, the Pascal statements

```
STRING 1: = 'HELLO. HOW ARE YOU
  TODAY?'
INSERT (', FRIEND', STRING1, 6)
```

will change STRING1 to read, "HELLO, FRIEND. HOW ARE YOU TODAY?" Other powerful string functions enhance interactive programs.

The UCSD version corrects a glaring deficiency of standard Pascal. Initially, Pascal did not support random access of files — a serious difficulty for disk users. UCSD Pascal includes the procedure SEEK which grants direct access to any numbered record in a file. For sequential access one may either read one record at a time or use the functions BLOCKREAD and BLOCKWRITE which transfer large segments of data to or from structured input/output devices (such as disks). UCSD enhancements to file manipulation and other I/O routines make FMG Pascal I/O operations versatile and broad, but easy for beginners who do not yet require all of the sophisticated functions but wish to grow into them.

The SEGMENT PROCEDURE, another UCSD extension, provides versatile memory allocation. Because the system software occupies a large block of memory, limited space is available for object code generated from source text. By using SEGMENT PROCEDURES the programmer may elect automatic reduction of a program into units that remain in secondary memory (e.g., disk) until they are needed. When the program requires them, the system loads them into RAM where they remain until overwritten by another segment procedure. Injudicious use of this feature will substantially decrease program speed.

System Operations and Support

Pascal for TRS-80 loads directly into RAM from a system disk, writing over the memory area otherwise used by TRSDOS. Consequently, programs may not use TRSDOS routines or vectors. Many of the Level II ROM routines are unusable, as they employ jumps to vectors outside the ROM.

Once loaded, UCSD Pascal presents a friendly welcoming message and requests a command. A user may select from several options: call the filer program, which itself presents a number of file manipulation and examination options; execute a program; run a program on which the user has been working; or invoke the editor, compiler, linker or assembler.

Programmers use the editor to produce any source code. It is an adequate utility with versatile editing instructions. Commands to insert, delete or change text, copy from one section of source to another, indent automatically, and justify right or left margins (but not both at once) make the editor useful and flexible. Additional advanced commands enhance it further. The absence of tab control in the TRS-80 version makes for slow columnization, especially of assembly language programs.

The compiler program translates source code from the editor into executable object code, called P-code. Source code may even contain references to other Pascal source programs, in which case the compiler will search for the necessary files and compile them into the object code, affording users a powerful facility to build libraries of external procedures and functions. Since variables within a program may be private to that program, added modules may use identical private label names without harm to the final compiled result.

During compilation the system displays its progress on the video screen. Upon discovery of a syntax error, the compiler will stop, describe the error, give its location and prompt the user to select one of three options: continued compilation (to find more errors); termination of compilation and return to the system control level; or transfer to the editor to repair the offending instruction.

Assuming a correct compilation, the user may run the program or invoke the linker which will add to the object code additional object code routines which the programmer has declared as external. Some of the Pascal I/O routines are not in the main system, but are stored in a system library. The linker finds the required routines and copies them into the otherwise incomplete code. Also, the linker permits a user to copy his own relocatable assembly language or Pascal utility programs into Pascal software as functions or procedures. In other words, the linker provides a facility whereby a Pascal program and a relocatable assembled programs may, likewise, be linked together.

The UCSD macro assembler is far better than Radio Shack's first Editor/Assembler, but not as powerful as Microsoft's Macro-80. Still, it provides a system which assembles relocatable machine code to be linked into Pascal programs, to run alone, or to be linked

```
10 DEFINT H,L
20 EP=1E-5
30 P1$="##########": P2$="      ##.#####[[[["
40 CLS: PRINT "FACTORIAL AND ROOT COMPUTATION "
50 PRINT: LINEINPUT "ENTER LOW LIMIT (0<LOW<21)   ==> "; LOWLIMIT$
60 LOWLIMIT=VAL(LOWLIMIT$)
70 IF LOWLIMIT<1 OR LOWLIMIT>20 THEN PRINT "ERROR " : END
80 PRINT: LINEINPUT "ENTER HIGH LIMIT (1<HIGH<21) ==> "; HIGHLIMIT$
90 HIGHLIMIT=VAL(HIGHLIMIT$)
100 IF HIGHLIMIT<1 OR HIGHLIMIT>20 THEN PRINT "ERROR " : END
110 FOR LOOPCOUNT=LOWLIMIT TO HIGHLIMIT
120 GOSUB 900
130 GOSUB 1000
140 PRINT USING P1$; LOWLIMIT;: PRINT USING P2$; RESULT; ROOT
150 NEXT
160 END
900 RESULT=1
910 FOR I%=LOOPCOUNT TO 1 STEP-1
920 RESULT=RESULT*I%
930 NEXT
940 RETURN
1000 ROOT=1
1010 ROOT=(ROOT+RESULT/ROOT)/2
1020 IF ABS(RESULT/(ROOT[2)-1)>EP THEN 1010
1030 RETURN
```

Listing 2 The Basic version of FACTROOT.

```
PROGRAM REVERSE;
VAR LEN, LOOP1, LOOP2: INTEGER;
  TARGET, NEWTARGET: STRING;
  CH: CHAR;
BEGIN
  WRITELN('ENTER YOUR TARGET STRING.');
  READLN(TARGET);
  NEWTARGET:=TARGET;
  LEN:=LENGTH(TARGET);
  FOR LOOP1:=1 TO 100 DO
    BEGIN
      FOR LOOP2:=LEN DOWNTO 1 DO
        BEGIN
        CH:=TARGET[LOOP2];
        NEWTARGET[LEN-LOOP2+1]:=CH;
        END; (*LOOP2*)
      TARGET:=NEWTARGET;
      WRITELN(TARGET)
    END (*LOOP1*)
END. (*REVERSE*)
```

Listing 3

REVERSE, a Pascal program to manipulate an input string.

might want to display on the screen at different times with different numbers or labels.

My main complaint about Level III graphics is the same as my complaint about Level II graphics, which is the screen just doesn't give you fine enough resolution. Try generating a circle and you'll see why higher resolution is more desirable. Only the crudest figures can be drawn with Level III.

Renumbering and "Single-Stroke" Capabilities

One of the concepts behind Level III BASIC is that it has enough features so that there is something for just about everyone. It may or may not really accomplish this goal. The next most important feature after graphics is renumbering program lines. If you've done much programming at all, you'll like this one. The Level III instruction for renumbering program lines is the NAME command. It is a flexible command, one that demonstrates the level of care for the user that has gone into most of Level III.

Using the NAME command, you can renumber program lines beginning at any location in the program. The chosen location can have any designated line number as long as it is greater than the previous line number. And, the following line numbers can be incremented by any number, as long as you don't go beyond 65,529.

One more thing; the NAME command automatically updates all the line number references following GOTO, GOSUB, THEN, ELSE, etc.

Some people feel that single-stroke instructions are the most important features of Level III BASIC. Actually, "single-stroke" is somewhat misleading and in deference to this, the Level III manual refers to them as "Shift-Key Entries." By pressing the SHIFT key and one of the letter keys (A-Z) you can enter an instruction, command, or any string up to 15 characters into the computer. In other words, instead of typing LIST and pressing the ENTER key each time you want to list a program, you can simply enter SHIFT L. To run a program, you simply enter SHIFT R.

Level III maintains a list of 26 Shift-Key Entries, but the really important thing is that they are user changeable. By using the new LSET command, you can change the Shift Keys to anything you want, including commonly used INPUT responses. And, just in case you ever figure out how to modify your TRS-80 for upper/lower case, you can turn the Shift-Keys off by typing LSET RESET and pressing ENTER. The command for turning them back on is LSET SET.

Strings and Other Things

Level III BASIC expands the use of INPUT statements in two important ways. First, by using the new statement, LINE INPUT, your program will accept inputs that include commas, quotation marks, dashes and all other punctuation marks. Also, LINE INPUT eliminates the automatic question mark (?) prompt. If you want a question mark, you simply include it in the prompt string.

The other way in which Level III expands the use of INPUT statements is with the addition of #LEN. By adding this to INPUT (INPUT#LEN) you can put a time limit on responses to INPUT prompts. This will be a very helpful for education and game programming. For example, you could write a math drill program that gives students ten seconds to work out each problem before going on to the next problem.

String manipulation power is increased in Level III BASIC with the addition of the INSTR function and the expanded use of MID$. INSTR, which is also found in Disk BASIC, allows you to search a string for a specified substring. For example, you could search a list of telephone numbers for numbers that have the 212 area code. MID$ can be used on the left-side of the equation in Level III as well as the right-side of the equation. This feature, too, is found in Disk BASIC.

Level III BASIC even corrects some of the problems with Level II BASIC. It eliminates keyboard bounce and provides more reliable cassette tape loading. Two new commands, LOAD and SAVE replace the Level II CLOAD and CSAVE commands. They permit cassette tape interfacing without the usual "volume sensitivity" of the tape recorder.

As with TRS-80 Disk BASIC, Level III BASIC spells out the error messages instead of displaying two letter abbreviations. Instead of getting the error code NF, you get the error message, NEXT WITHOUT FOR. This greatly simplifies debugging because you don't waste time referring to the error code chart.

With a TRS-80 Expansion Interface, you can make use of the Level III string function, TIME$. It can be used like any other string function to add a built-in digital clock and calendar to your computer.

Upon loading, TIME$ is set to all zeroes (0), and it begins working automatically. You can easily set it to the correct time and date, then use it to time various program activities, or to perform calculations where time and date are essential.

With the optional RS-232 port, then you can use the new statement, PRINT#-3. This allows you to output information through the RS-232 port to control line printers, LED's, stepping motors, lights, or whatever.

For those sophisticated enough to be using machine language subroutines in their programs, Level III simplifies exit to user subroutines with automatic conversion of Hex and Octal Constants. It also allows 10 user defined routines, as compared to one in Level II BASIC.

Summary

In addition to the features of Level III BASIC, many people will appreciate the way in which it has been documented. The instruction booklet is based on extensive documentation written by Microsoft's Andrea Lewis. It was written for people who have some programming experience, but the user doesn't have to be an expert to understand it. Examples and sample programs are abundant.

One apparent result of the care that has gone into Level III is that on the whole it is a more professional package than Radio Shack's Level II BASIC. It's hopeful that this effort pays off and that software to follow will equal or exceed GRT's standards.

Now for some of the negative observations promised earlier. To begin with, Level III wouldn't be necessary if it weren't for some apparent shortcomings of Level II BASIC, which was also written by Microsoft. The problems with loading and saving programs on cassette tape (CLOAD and CSAVE) are particularly irritating. Why didn't Microsoft and Radio Shack see this problem and correct it? It seems that too many companies are always in too big a hurry to get product out. Level II BASIC probably should not have been released in the first place with these

problems.

While Level III BASIC can be stored on disk as a file, it can't be used with Disk BASIC. It's hard to imagine that many Disk users will want to give up the use of file commands. Actually, Level III probably shouldn't be marketed to Disk BASIC users at all. Instead, a disk version of Level III with the neat graphics statements and other unique features should be planned for the near future.

One more concern is the 5.2K RAM memory that Level III requires. Granted, this isn't very much when you consider all the new features, but a TRS-80 without an Expansion Interface can only have a maximum of 16K RAM. Many TRS-80 programs written in Level II BASIC, including those marketed by GRT, take more than 12K RAM. You won't be able to take advantage of LEVEL III's new LOAD command for these programs.

As I said, you have to weigh the advantages of Level III against the cost. The decision is yours. □

Example of Level III BASIC Graphics Program*

The following program is a Level III graphics program that uses the three new graphic statements, LINE, GET @ and PUT @. It generates the image of a rocket ship and makes it move up the screen. It was written in graphics mode.

Program	Explanation
10 'ROCKET IN GRAPHICS MODE	Remark
20 CLS	Clear Screen
30 LINE (3,1) - (3,2), SET	
40 LINE (2,3)- (4,4), SET,BF	Draws Rocket
50 LINE (1,4) - (1,5), SET	
60 LINE (5,4) - (5,5), SET	
70 DIM A% (2)	Dimension Array
80 GET @ (1,1) - (5,6), G,A%	GETting Array
90 CLS	Clear Screen
100 FOR Y = 42 TO 1 STEP -1	So rocket will start at bottom of screen and move up
110 PUT @ (1,Y) - (5,Y + 5), SET, A%	PUTting array on bottom of screen
120 NEXT Y	Creates movement
130 CLS	Clears screen
140 GOTO 100	Let's do it again, and again, and again...

*This example Is from the Level III Instruction Booklet.

Forth For the TRS-80

Anthony T. Scarpelli

If you are interested in one of the most powerful computer languages ever invented, then you will be interested in this review of Forth.

Forth was created in 1969 by Charles H. Moore at the National Radio Astronomy Observatory in Charlottesville, VA. It has grown into a language that not only controls the radio telescopes of that observatory, but is used at other observatories around the country. It is also one of the better application languages around because of the ease and speed with which programs can be written in it.

When I became interested in Forth, I didn't want to spend a great deal of money just to try it out, so I invested in the public domain assembly language source code of the language which is offered by the Forth Interest Group for less than $20. That version is meant to be implemented on an 8080 microprocessor based machine, and is not easily converted to a Z80 based microcomputer.

Anthony T. Scarpelli, 98 Foxcroft Dr., Scarborough, ME 04074.

By far the easiest way to try Forth is to purchase a version designed to run on a Z80 computer.

The version I now have is MMSForth, which costs $129.95 for the disk-based version 2.0 from Miller Microcomputer Services.

Getting Started

The first thing you notice when you open the package is the loose-leaf, three-ring binder containing 135 pages of information on Forth: how to get it running, how to program with it, and some examples of simple programs. When you send in your license agreement, you receive another 67 pages which contain a memory map, the Forth glossary, 8080 assembler tables, and several other tables and lists.

It takes several hours to read through the manual, but to get started you need read only the preface and the first appendix.

The first appendix tells how to boot the system disk. What appears after the boot is the copyright information, your serial number, a copyright message, and the address of MMS. A few seconds later the directory listing the options you have available appears. You need not choose any of them, though, the boot loads the Forth language and is ready to go.

This directory is called the Utilities menu, and allows you to choose from the following:

FORMAT allows you to format a diskette.

BACKUP allows you to back up the system or any Forth diskette.

COPIES allows you to copy a range of blocks (a block is 1024 bytes of information).

SEARCH allows you to search for occurrences of words.

into other machine language software. More significantly, it is the only assembler that will work with the FMG UCSD Pascal package. Typically, assembled machine language programs or subroutines run much faster than comparable Pascal programs.

Some Fine Points and Some Problems

We have been discussing Pascal as a compiled language. Strictly speaking, UCSD Pascal for the TRS-80 is also an interpreted language. The Pascal compiler translates source code into a kind of object code called P-code which is not machine executable as is an assembly of Z-80 mnemonics. An assembly is directly runnable on a microprocessor. P-code is a special low level translation of Pascal source code, and is substantially the same either for a TRS-80 or a PDP-11. Its mnemonics look like assembly language and its object code looks like hexadecimal object code, but it needs further interpretation.

Since the Pascal P-code cannot run on a Z-80 microprocessor, the Pascal pseudo-machine prepares a final translation each time the compiled program is run. The pseudo-machine is a program in Pascal that emulates hardware when it reads individual P-code instructions. It then translates the P-code into Z-80 instructions and passes those to the microprocessor for execution.

The pseudo-machine implementation of UCSD Pascal assures standardization and system portability. To prepare new releases, all the implementor needs to vary is the pseudo-machine interpreter so that it will generate output in the microprocessor's native language. The library, compiler, linker and various other system components stay largely unchanged.

On the other hand, UCSD pseudo-machine implementation creates a serious problem for the user. A Z-80 treats P-code as meaningless instructions. Consequently, the TRS-80 user is unable to convert compiled Pascal programs into TRSDOS command files to be invoked from the command mode of the disk operating system. Although a minor problem for programmers dedicated to Pascal alone, those who work in several languages may find it inconvenient to be unable to access Pascal routines in those languages.

Moreover, since all Pascal files are placed on disk according to rules established in the pseudo-machine, the user may access data files and even true machine code files produced with the UCSD assembler only within the Pascal system. Conversely, the Pascal system is not presently able to access TRSDOS files.

UCSD Pascal presents some additional difficulties. The system lacks facility for double precision arithmetic. Don French, who implemented the FMG Pascal release, suggests that a revision is coming soon that will use binary coded decimal routines to permit multiple precision and long integers. FMG deliberately omitted turtlegraphics from the system. Turtlegraphics is a line drawing facility, for which the TRS-80 graphics are much too coarse. Substitute graphics instructions would be a constructive addition to future revisions of FMG UCSD Pascal.

A committee prepared the users' manual, published for UCSD by its Institute for Information Systems. Organized poorly, it mercilessly sends the reader leapfrogging through text to refine comprehension of any given point. We can forgive the style, but too much expense was spared on typography and layout. Although the authors did not intend the manual as a tutorial, it does not serve well as a reference, even to its own operating system.

The FMG release of UCSD Pascal for TRS-80 requires an environment with 48K of user RAM and two disk drives. It is an excellent language package, a fine means to learn structured programming and a versatile system for the hobbyist, business user or home computer enthusiast. Its minor deficiencies are overwhelmed by its strengths and, consequently should become a microcomputer standard. □

The FMG/UCSD Pascal system is available from FMG Corporation, PO Box 16020 - B9, Fort Worth, TX 76133 for $150 (or $100 without Macro Assembler, linker and library).

Operation	Purpose
DIV	Integer division operator.
MOD	Integer remaindering operator.
ODD	Determines if an integer is odd.
PRED	Returns predecessor of an argument.
SUCC	Returns successor of an argument.
SIZEOF*	Returns number of bytes allocated to a variable.
CASE	Multiple branch "if" statement.
REPEAT . . UNTIL	Conditional iterative operation.
WHILE . . DO	Conditional iterative operation.
CONCAT*	Concatenates strings.
COPY*	Accesses indexed section of a string.
DELETE*	Removes characters from a string.
GOTOXY*	Addresses cursor to screen coordinate position.
IORESULT*	Advises program of error in I/O routine.
BLOCKREAD*	Reads blocks of data from a file.
BLOCKWRITE*	Writes blocks of data to a file.
SEEK*	Allows random access to a file.
SEGMENT PROCEDURE*	Retains procedure on disk to be overlaid when called.
EXTERNAL*	Declares procedure to be linked from a library. *denotes UCSD extension

A Selection of UCSD Pascal Operations

```
10 CLEAR 500
20 DEFSTR T
30 PRINT "ENTER YOUR TARGET STRING."
40 LINEINPUT TARGT
50 FOR I1%=1 TO 100
60 T2ARGT=""
70 FOR I2%=LEN(TARGT) TO 1 STEP-1
80 T2ARGT=T2ARGT+MID$(TARGT, I2%, 1)
90 NEXT I2%
100 TARGT=T2ARGT
110 PRINT TARGT
120 NEXT I1%
```

Listing 4
The Basic version of REVERSE.

Bibliography

Kenneth L. Bowles. **Microcomputer Problem Solving Using Pascal.** New York: Springer-Verlag. 1977. — Although limited in scope, this text is geared specifically to UCSD Pascal and to small computers.

Peter Grogono. **Programming in Pascal.** Reading, MA: Addison-Wesley Publishing Co., Inc. 1979. — Excellent tutorial, covering all features of standard Pascal. Superb bibliography.

Kathleen Jensen and Niklaus Wirth. **Pascal User Manual and Report, Second Edition.** New York: Springer-Verlag. 1979. — The language standard. A must for the serious Pascal user.

UCSD Pascal System II.0 User's Manual. La Jolla, CA: Institute for Information Systems. 1979 — The system manual. Whatever its deficiencies, it is required.

Niklaus Wirth. **Systematic Programming: An Introduction.** Englewood Cliffs, NJ: Prentice-Hall, Inc. 1973. — Discusses structured programming for ALGOL 60 and Pascal.

Expand the Power of Your TRS-80

TRS-80 Level III Basic

David Bunnell

My Level II TRS-80 Computer with 16K RAM memory has more power than any other Level II TRS-80 Computer with 16K RAM in the world! But since this will be short-lived, I want to tell you about it now while it lasts.

The secret to my powerful TRS-80 is that its Level II BASIC has been "enhanced" by a new software package called Level III BASIC. Written by Microsoft, the same company that wrote Level II BASIC, Altair BASIC, PET BASIC, Applesoft BASIC and several others, Level III BASIC comes on cassette tape and takes 5.2K of RAM memory. It was previously sold under the G2 label from GRT Corporation but since they've gone out of business it is now being sold by Microsoft Consumer Products for $49.95 (10800 N.E. 8th, Suite 819, Bellevue, WA 98004). One of their dealers is Lifeboat Associates, 2248 Broadway, NY, NY 10024.

I'm a free-lance writer who was fortunate enough to write the instruction booklet for Level III. I virtually "lived with" Level III BASIC for six weeks and I'm anxious to let other people know what I found out about this new software.

But before I go on, let me assure you that while I am enthusiastic about Level III, I am not writing this as a favor for anyone. In fact, I have some critical observations to make that Microsoft would probably rather not be included.

System Requirements

Level III BASIC was written primarily for TRS-80 Computers with Level II BASIC and 16K RAM memory. However, it is also being marketed to people who have TRS-80 systems with Expansion Interfaces and disk drives. One feature, the TIME$ string, was written to take advantage of the Real Time Clock, which is a part of the Expansion Interface. Another feature, the PRINT#-3 command, will let you output information through the RS-232 port if you have an Expansion Interface with this option.

You can use Level III if you have a TRS-80 with disk drives, but it will not work with Disk BASIC. You can simply store it on disk as a file and then load it on top of Level II BASIC. So be advised, the advantages of Level III BASIC for disk systems is marginal. Any program written in Level III BASIC can only be stored on cassette tape. You cannot store Level III programs on your disk.

Why would anyone with a TRS-80 disk system buy Level III BASIC? The answer is that in addition to adding many Disk BASIC features to Level II BASIC, Level III BASIC also adds some features that are currently unique to both Level II BASIC and Disk BASIC. The most important of these are some powerful new graphic statements that make it much easier to write TRS-80 programs with charts, graphs and even crude animation.

Other features unique to Level III include: renumbering program lines, single-stroke instructions, new SAVE and LOAD commands and the option of setting a time limit for responses to INPUT prompts.

The remaining features of Level III BASIC are all TRS-80 Disk BASIC features. They include: spelled-out error messages, 10 user-defined routines, LINE INPUT statement, more flexible MID$, INSTR string, user-defined functions and hex and octal conversion.

Most of these features will be discussed in more detail, but first, the price of Level III, which is $49.95, should be mentioned. We all realize that there are always new things to buy for our computers, whether they be software packages, programs, hardware options, or peripherals. Therefore, the question should be: Does Level III add enough new capability to an already powerful Level II BASIC to make it worth the investment?

If you like graphics in your programs and want to write programs with graphics, then the answer to the above question is probably "yes." Actually, you can't do anything graphically with Level III BASIC that can't be done with Level II, however Level III makes it much easier and more efficient (you use up less memory).

Graphics Capability

The Level III graphic statement, LINE, lets you draw either a line or a rectangle on the screen between any two designated points. You don't have to methodically plot out every point as you do with Level II.

The LINE statement works in two modes, graphics mode and character mode. In the first of these, the screen is divided into a grid that is 128 across by 48 down. In character mode, the resolution is not so fine (64 by 16), but motion across the screen is much faster. Character mode also gives you the option of drawing on the screen with alphanumeric characters or any of Level II's graphic characters (remember the CHR$ string?).

In addition to the LINE statement, Level III also adds a GET @ and PUT @ statement. These statements let you save a graphic array (called GETting an array) to be put back on the screen at any designated position (called PUTting an array). These statements greatly simplify the procedures for writing programs with animation. They can also be used to save the format of charts or graphs which you

David Bunnell, 4629 Geary, San Francisco, CA 94118.

TRANSLATE allows you to translate from the older version 1.9 to the present version.

ALLCAPS allows you to change lowercase letters in some of the blocks to uppercase if your system doesn't support uppercase.

CUSTOMIZE allows you to configure the system for your own TRS-80.

EXTENSIONS displays the system options available.

PROGRAMS displays the programs that are available on the other diskette.

The extensions include: DBL-PREC (double-precision numbers), ARRAYS (one and two dimensions), STRINGS (similar to Level II strings), RANDOM (a random number generator), GRAPHICS (the TRS-80 graphics), SCREEN-PRINT (prints the screen to your printer including graphics characters), CASSETTE (tape routines), CLOCK (time and date routines), and TOOLKIT (various other handy routines).

The power of Forth comes from its ability to create new words, and to create words that define other words. When you use a word, Forth executes it as long as it is in the dictionary, and as long as any parameters the word may need precede it.

This version of Forth includes about 200 words with which you can create more words. One of the ways to create a word is with a colon definition. For example, if I write:

: TEST word word word etc. ;

I have defined a word called TEST using several previously defined words. The colon precedes the new word, and the semicolon ends the definition. This process continues until an entire routine or even a program can be called by just one word.

All of the utilities, extensions, and programs in MMSForth are words that have been previously defined, words that are made up of the core words, and core words that eventually become machine language routines.

Features

Let us now consider some of the features that make this version of Forth a very good buy and a very handy development system.

MMSForth will run on a 16K machine with only one disk drive. With 32K or more, however, there is more room for words and programs. Backing up a diskette can be done on just one drive, so a minimum system is all that is needed to use the language. It will also support multiple drive systems, and a version for the Model III is available.

One of the nice things about MMS-Forth is that many of the features of Level II Basic have been incorporated as extensions. Strings and graphics use similar words, and are used much like their Basic counterparts. The difference is that the execution speed of Forth is nearly as fast as assembly language, so graphics and string manipulation are much faster.

Another feature is that the screen-print routine will print the TRS-80 graphics if your printer is capable of handling them. The printer driver can be the regular ROM routine or a custom driver that can be changed to fit your system. An extended driver is available with page-formatting features.

SOFTWARE PROFILE

Name: MMSForth

Type: Language and programming environment

System: TRS-80 Model I, III 16K, IBM PC 32K

Format: Disk

Language: Machine

Summary: Good value

Price: TRS-80 version $129.95, IBM version $249.95

Manufacturer:

Miller Microcomputer Services
61 Lake Shore Rd.
Natick, MA 01760

Programs

Some of the programs included on the second disk show Forth at its best. The SORT program demonstrates differences in speeds between different types of sorts. To show, for example, how slow an insertion sort is, this program loads the screen with a group of random characters, then sorts them. It does the same for the selection and Shell sorts, the quicksort, and a quicksort with assembly partition. It is fun to watch, and a good way to show off your system.

The game of Life is an old favorite. Programmed in Basic, the game takes a long time to go through a generation, but this Forth version is nearly as fast as an assembly language program.

One of the handy things you can do with Forth is to put assembly language mnemonics right into the definitions. Or you can create "code" words of just assembly language mnemonics. This version of Forth contains an 8080 assembler. When the loops of Life are converted to assembly language routines, the speed of a generation is extremely fast—much less than a second.

To start you off, a few patterns are available to initialize the Life generations. With the doodle routine, you can draw your own patterns.

Another game on the disk, BREAK-FORTH, demonstrates how a high level language can be used to write a fast action, real-time version of a popular game.

CHECKBOOK is an example of a business program written in Forth. The source code for the program is provided and explained word by word.

Using some of the standard Forth editor commands, and some of its own, NOTE-PAD allows you to write one page of text.

The Editor

The standard screen editor provided with the system is one of the best I have used. By using the CLEAR key, and the SHIFT and CLEAR keys together, you can delete, insert, and move characters, lines and whole screens of information. The arrow keys move the cursor around the screen with ease.

Forth is written in blocks of 1024 characters which comprise one screen of information (64 characters by 16 lines). Writing Forth programs involves the editing of these screens. Once a block is edited, it is saved in one of the two block buffers so another block can be edited. When a third block is called up, the first block is automatically saved on disk. When you finish writing the program, the word FLUSH saves the remaining blocks to disk.

Documentation

I am a firm believer in good documentation which provides as much information about a program as possible. MMS-Forth does a pretty good job of it, but doesn't go all the way. If you were to buy a Basic interpreter, you can be quite sure you wouldn't get the source code for it. With MMSForth you receive the source code for the entire system disk except for the first 13 blocks. You must list it yourself, but it is there. You also get the source code for the program disk.

The first 13 blocks contain the core words and assembly language routines as well as the disk I/O. The blocks which are provided allow you to see high level Forth programming, and when you become good at Forth, you can modify the code to improve upon or customize it for your own purposes. A total of 128 blocks of source code is a great deal to understand and change, but at least it is there, and I commend MMS for providing it.

If you have heard anything about standardization, it is probably that everyone

would like it, but little is being done about it. There are currently four languages that have been standardized, but Forth is not yet one of them. However, thanks to a great deal of effort on the part of people who want to see Forth standardized, the Forth-79 Standard has come into existence.

MMSForth 2.0 contains the words published in this standard. This means that source code created on the TRS-80 can be transported to any other Forth machine and vice versa. That is the theory, anyway. I commend MMS for going along with the standard, and hope that in the future we TRS-80 owners will be able to talk to an Apple or IBM owner as easily as to another TRS-80 owner.

I have read everything available on Forth, and I can say that, except for one book, learning Forth from the available information is like learning Chinese from a dictionary. I thank MMS for providing a great deal of information on Forth, but Forth is an entire language, and you cannot learn it simply by reading the documentation normally provided with the system. You must go to other sources, and you must sit at the keyboard and work with the language.

Forth was not an easy language for me to learn. All of the manuals I have read started out with the easy things, but had a tendency to stop. The more difficult concepts, the use of assembly language, the extensibility of Forth, the best way to program in Forth, and good programming techniques, just were not there. I had to struggle with words that were defined, words that were not clearly explained, and words that were not explained at all.

MMS provides you with enough to start out, but you must purchase and study some of the other books and manuals available before you will become good at Forth. Luckily, one of the best books, *Starting Forth* by Leo Brodie (Prentice-Hall, 1981), is also sold by MMS. I recommend purchasing this book along with this program.

Although I have been very pleased with MMSForth, I have two real complaints about it. The first is that looking up unfamiliar information is very difficult. There is a table of contents, but it is too broad to be of much help.

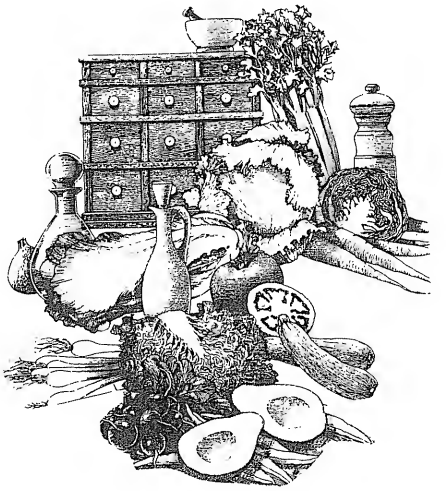
My second complaint is that occasionally some aspect of what I wanted to know was not explained in sufficient detail for a beginner. An index would be very handy, and I understand that one is being compiled. So there is hope. Whenever I ran into a real problem, a quick call to Miller Microcomputing got me an answer to it.

Summary

My overall recommendation is to buy this program if you want to try Forth. It is definitely a good value. □

Diet Programs for the TRS-80

Better Nutrition Bite by Byte



Stephen Kimmel

If you want the computer to have an impact on the maximum number of people, then you have to pick a task that in some way affects the lives of many of them. In this frame of mind, I kicked off the sheets and blankets, poked Georgia in the ribs and asked her what the most common human activity was.

"Huh? What time is it?" she moaned, a model of erudition.

"Three o'clock. What's the most common human activity?"

"Sleep," my wife the dietitian and president-elect of the Oklahoma Dietetic Association said. She was unconscious again almost before she finished the word.

Stephen Kimmel, 4756 S. Irvington Pl., Tulsa, OK 74135.

"Oh," I responded. I lay there thinking of some way for the computer to affect the way people sleep. As with breathing, I couldn't imagine a way for the computer to become involved in this undeniably common activity.

Eating is the most common human activity that involves any thought processes. Unlike breathing or sleeping, we have to decide what we are going to eat.

And for millions of Americans it isn't simply a matter of finding out what's in the refrigerator. Nearly 40% of all Americans are on some kind of diet. This includes about 5% on diabetic diets and an equal number on salt restricted diets for high blood pressure. Nearly a third of us are trying to lose weight. Look at your bookstore if you need more evidence. Look at the best seller lists. See how many diet books you find.

Aha! A perfect application for the home computer! Enrich the lives of millions by enabling them to take better control of their diets and hence their lives. Or is it?

Nutrition Is an Imprecise Science

Nutrition is a classic example of a field where precision doesn't matter. Consider a specific example; vitamin C in an orange. The nutrient tables will tell you to the nearest milligram what is contained in an average orange. One medium California Navel orange contains 85 milligrams of vitamin C. A Valencia orange has 59. Oranges from Florida contain an average

of 68 milligrams. All of which is meaningless because oranges can range from a tiny 2" in diameter to a colossal 4". That means that two oranges of the same variety could vary as much in size as 800% with a corresponding variation in vitamin C content.

Actual content will vary according to the weather, the soil, and a thousand other factors. There are ways of telling exactly how much vitamin C there is in a specific orange. Unfortunately, the orange is inedible when the process is complete.

Fortunately, the fact that accuracy is impossible is unimportant. The body is a remarkably efficient chemical factory. If you eat too much of something, then the body will save it; as it does when it stores vitamin D in fat tissue; convert it to something else, protein to glucose and fat (Almost all Americans eat too much protein); or throw it away, as it does with excess iron. If, perchance, you don't eat enough of something, the body will take it out of storage, make it out of something else if it can, or simply do without it for a while.

It is difficult, at best, to determine what a human needs to consume. It depends on his metabolic rates, his particular body chemistry and his weight. Anyone who believes that accuracy in this field is possible is invited to tell me to the nearest gram what he weighs—and converting your weight from a bathroom scale doesn't count since it is only accurate to the nearest pound. Maybe. The point is that it is absolutely impossible to say exactly what any individual needs to consume.

It is impossible to know precisely what you eat, and equally impossible to know precisely what you need to eat. Your response to this fact of life is to eat a variety of foods every day. The surest way to become malnourished is to eat the same thing every day for a prolonged period of time. The surest way to maintain good nutrition is to eat a wide variety of foods, including breads, meats, vegetables, fats, and fruits.

Enter the Computer

This may make it sound as if there is no need for computer programs. Not quite true. There is growing awareness of the importance of nutrition; even doctors are beginning to take nutritional considerations seriously. Few human activities have as much potential to improve the quality of our lives as eating well.

Nutrition calculations, however, while simple, are tedious and time consuming. Not surprisingly there is a great deal of activity in this field. It is comparable to anything except accounting and games, with at least fifty programs in use, under development and/or for sale in America. Now look through the ads in this magazine. Where are all the diet programs?

Almost every hospital and every school with a nutrition curriculum has a diet program running on its computer system—typically the Megalith 5000 series machine—and almost all were developed in-house. So there are programs out there, and there are people writing diet programs.

Why, then, are there so few for small computers? A fair question and one that I asked myself frequently in preparing this article. The answer lies in the fact that most of the programs were written by people who suffered from Big Computer Psychosis.

Big Computer Psychosis, as we all know, is a horrible malady that afflicts people who have a moderate working knowledge of some so-called "main frame computer." They can be instantly identified by their use of the phrase, "That'll never fit on a microcomputer." Their logic runs like this: "I had a hell of a time getting this on my $100,000 computer. It just barely fits. I don't know much about microcomputers, but I know they're smaller than my computer. Therefore, it will never fit on a microcomputer." Wonderful logic. Also very wrong.

Texas Tech University, like most large schools, has a Department of Food and Nutrition within its college of Home Economics. (That is certainly an artifact of the days when medical people thought nutrition had little to do with an individual's health and dietitians were fat old ladies.)

Texas Tech's diet program is called, cleverly enough, DIET. It is available as a deck of cards (740 for the program, 647 for the data base) for duplication costs.

DIET is written in Cobol and executed on an IBM 370/145. It was designed for nutrition students to use in performing detailed analyses on one or more days of meals. The student enters the quantity and the food eaten and the program computes the quantity of each of 28 nutrients consumed during the time period. The Texas Tech program, which is one of the best, contains 626 foods and 28 nutrients. That's quite a bit of information.

It is easy to see why a diet program would be difficult to fit on a computer of any size. Basically, a diet program is very little more than a database manipulator. The essential mathematics are very simple. You multiply the nutritional content of a given food by the amount eaten, add the results to the other foods eaten during a given time period and compare the total to the recommended daily allowance (RDA). The program isn't much. It is the data base that consumes all the memory.

As a rule of thumb you need at least 100 foods to even begin to do a reasonable job of simulating an American diet. The question is how many nutrients and qualities you want to maintain for each. Suppose you have 20 nutrients for each of 300 foods and you store them as floating point variables. Further assume that each food has a twelve-letter name. Without even blinking you've used up nearly 28000 bytes. Then if you use data statements to put the information into memory, you've used up that number again. Now you've consumed 56000 bytes, more than most personal computers have available. And you still haven't determined the individual's RDA.

Your choice, then, is fewer foods or fewer nutrients or better programming. Since most of the creators of large programs have large computers, they choose the brute force approach. Small wonder they don't think a home computer can do the job.

Contrary to the opinion common among programmers working on large machines, the small computer is fully capable of doing the job. Some of these programs, I found, do the job better than the monsters. In this article I will review a few of the programs available. Some are magnificent efforts. Some are simply funny.

Diet Information Program

The first of our programs is in Dwyer and Critchfield's "Basic and the Personal Computer." Diet Information Program was originally designed as a practical example of a data base program. It includes 28 foods and 18 nutrients—too few to be of any practical use. Pick 28 foods, any 28 foods, and eat nothing else. If you stick to it for over a week, write and tell me what you ate.

Indeed, the authors really have illustrative purposes in mind and the program wasn't meant to be used. It performs the most fundamental task of a diet program: you input a group of foods and the program tells you how that compares with your requirements. It you were unable to find anything better, you can easily expand the number of foods. The authors list several fine sources of additional data where you can find additional food numbers to enter in your program.

Still, you will run out of memory quickly, since the running program has the data as both statements in the program and as data. Very fundamental. Don't bother. There are programs available that will do the job.

Diet

Diet, by Creative Computing Software as part of Ecology Simulations-2, is one that will do the job. It is one of four

programs designed for classroom use. It is an interesting combination: pollution control, rat control, disease control and fat control. Here we have the minimum 100 foods and only four nutrients for each. However the four nutrients are the most fundamental, protein, fat, carbohydrates, and calories—if you want to fall into the fallacy of thinking of a calorie as a nutrient.

The program implies the "wide variety" concept of nutrition. If you eat a wide enough variety you would pick up the required vitamins and minerals. The Creative Computing program isn't really designed for home use. It is geared toward classroom application and instruction. The student enters what he has eaten for breakfast, lunch, dinner and a snack and the program tells him how close he has come to meeting his daily needs. It does a reasonably good job.

To use Diet in the home environment will require few modifications. It can be used to design a workable diet and will project how much you can expect to lose or gain if you stick to a prescribed diet. It might have been better if it allowed you to adjust your menu on-line and gave you a continuous update on how the foods you had selected so far compared with the prescribed quantities. These are fairly simple modifications and well within the abilities of most programmers.

The program really needs a printed sheet to go with it so the user can record what he eats, or plans to eat. To help you,

BREAKFAST MENU

Serving Unit	2100 Calories Number of Servings
2 slices lunchmeat	1 2/3
1/2 cup cooked cereal	1 2/3
1 cup low fat milk (or buttermilk or yogurt)	1 2/3
1/8 avocado	1 2/3
12 grapes	1 2/3
tea or coffee (no sugar) or bouillon	1

Select: 1-Different Foods or 2-Return to Meal Menu

LUNCH MENU

Serving Unit	2100 Calories Number of Servings
2 eggs	1 2/3
1/2 cup rice	1 2/3
2 tsp. French dressing	1 2/3
1 cup low fat milk (or buttermilk or yogurt)	1 2/3
1 small orange	1 2/3
pickles (not sweetened)	1

Select: 1-Different Foods or 2-Return to Meal Menu

DINNER MENU

Serving Unit	2100 Calories Number of Servings
2 eggs	1 2/3
1/2 cup cooked spaghetti or noodles	1 2/3
1 cup cabbage	1 2/3
10 cherries	1 2/3
pickles (not sweetened)	1

Select: 1-Different Foods or 2-Return to Meal Menu

Figure 2. Sample Menus Produced by Flexidiet.

1 BACON
2 BOUILLON
3 CORNED BEEF
4 ROAST BEEF
5 BOLOGNA
6 CHICKEN
7 FLOUNDER
8 FRANKFURTER
9 HAM
10 HAMBURGER
11 LIVER
12 PORK CHOP
13 SALMON
14 SAUSAGE
15 TUNA
16 TURKEY
17 VEAL
18 BEEF STEW
19 PEANUT BUTTER
20 PIZZA
21 CREAMED SOUP
22 TOMATO SOUP
23 VEGETABLE SOUP
24 SOYBEANS
25 SPAGHETTI
26 BUTTER
27 CHEDDAR CHEESE
28 COTTAGE CHEESE
29 CREAM CHEESE
30 SWISS CHEESE
31 CREAM
32 EGG
33 MARGARINE
34 WHOLE MILK
35 SKIM MILK
36 MALTED MILK
37 YOGURT
38 BAKED BEANS
39 LIMA BEANS
40 GREEN BEANS
41 BEETS
42 BROCCOLI
43 CARROTS
44 COOKED CARROTS
45 CORN
46 LETTUCE
47 MAYONNAISE
48 PEAS
49 POTATOES
50 FRENCH FRIES
51 OIL & VINEGAR
52 SALAD DRESSING
53 SPINACH
54 BISCUIT
55 WHITE BREAD
56 WHEAT BREAD
57 CORNFLAKES
58 GRAHAM CRACKER
59 CRACKER
60 FLOUR
61 PASTA
62 OATMEAL
63 PANCAKE
64 RAISIN BRAN
65 WHITE RICE
66 BROWN RICE
67 WHEAT GERM
68 APPLE
69 BANANA
70 CANTALOUPE
71 CUCUMBER
72 GRAPEFRUIT
73 GRAPES
74 ORANGE
75 ORANGE JUICE
76 PEACHES
77 PINEAPPLE
78 RAISINS
79 STRAWBERRIES
80 TOMATO
81 WATERMELON
82 CAKE
83 CANDY
84 CHOCOLATE
85 SANDWICH COOKIES
86 DOUGHNUT
87 GELATIN
88 HONEY
89 ICE CREAM
90 JAM/JELLY
91 PEANUTS
92 CUSTARD PIE
93 FRUIT PIE
94 POPCORN
95 POTATO CHIP
96 PUDDING
97 SUGAR
98 BEER
99 COFFEE
100 COLA
101 TEA
102 WINE

Figure 1. List of Foods in Creative Computing's Diet.

Figure 1 provides a list of the foods in the Diet database. Make copies of it and use it to record what you eat as you eat it. The ability to recall what you have eaten in the last 24 hours is frequently shaky. Quick. What did you have for lunch yesterday? How many ounces of milk did you drink yesterday? Was it whole, 2% or skim milk? For the diet program to do its job right, it has to know precisely what and how much of it you have eaten. Unfortunately, people don't drink five ounces of milk. They drink a big glass or a little glass. Once again we see the impossibility of accuracy. That's the major problem with our next program.

Compudiet

One of the best programs, and certainly one of the most obscure, is Compudiet by Information Technology Systems. The program is designed to work on a 16K TRS-80. My current version has 127 foods and 13 nutrients in its database. Compudiet 1.4, which has an uncertain release date, is said to contain 200 foods and 20 nutrients, and will be comparable to many of the programs the "Big Computer Psychosis" people have.

It overcomes the database problem with another program which is definitely available. BASiCIO, also from Information Technology Systems, is a machine language program that is built into other programs and performs what amounts to a core image dump and retrieval with the cassette. The data is dumped exactly as it appears in the computer. It saves and loads a huge array of data very quickly and with maximum economy of both memory and tape.

Why haven't you heard of Compudiet? Because Information Technology Systems doesn't have the economic resources of Instant Software.

What makes Compudiet such a great program isn't the fact that it loads faster than anyone else's program or that it has a large database. No, the real strength of Compudiet is that the author recognizes the fact that the computer and nutrition analysis are only a part of the total weight loss system. Compudiet comes with a dieter's manual (!) that is designed to help you identify the whys and wherefores of your eating. It addresses the issue of whether you eat when you are nervous or bored or lonely. Only Compudiet is concerned with the reason for a weight problem, and of the group, Compudiet has the best chance of helping you lose weight.

But, alas, the program may no longer be available. Such is justice.

Datadiet

So far we have seen nothing to really stir the big guys from their conviction that the task is too big for a home computer. Compudiet, they say smilingly, is a nice try but horribly limited. Well, boys and girls, that's about to change.

Out in California is a bunch of folks who will make all these people stand up and take notice. There is a program that would show these people that a small computer can do anything the big ones can do. IPC Datadiet has a program called Datadiet (don't you love the originality in this field?) that contains 1000 foods (expandable to 1500) and 14 nutrients. That's more foods than almost any of the megalith programs. It is designed to operate on a 48K TRS-80 with two 35-track single density disk drives. Fifteen hundred foods with 14 nutrients and a long name? How do you get all that into a small computer at once? Simple. You don't. You bring whatever data you need from the disk and you leave the rest safely out of memory. Thus, you can have a couple thousand foods available to you and still have some room for your program.

Datadiet wasn't designed with the home user in mind. It was written to serve professional dietitians and physicians. No home user needs the kind of power available through the Datadiet system. The system appears to be capable of coping with a hospital dietitian's entire work load. Of course, this program doesn't come cheap. Datadiet sells for $399 to users who don't require special consultant help, which shouldn't be necessary since the instructions are amazingly clear and easy to follow. This is a fully professional program and while the price tag seems a little steep to me it is in line with the sort of accounting software that is suitable for major commercial purposes. Figure 3 is a listing of a small portion of the database in Datadiet.

IPC Datadiet has two other programs currently available which are also designed for the nutrition professional. Recall24 uses a much smaller database than Datadiet with just 54 foods. However, using the exchange system it offers an effective 170 foods. When you use the exchange system you can't get much more than protein, fat and carbohydrates although Recall24 does include cholesterol too.

Recall24 has the best fully interactive menu evaluation I've seen, giving running totals of each nutrient. The program calculates your current diet according to sex, age, and activity level. After entering what you have eaten in the last 24 hours (a 24 hour recall), the program evaluates what you have eaten by food groups. Again, this is a fully professional program (that is suitable for home use) and you pay for it. Recall24 goes for $160.

Fastfood is, by comparison, dirt cheap at $35 and is essentially identical to Recall24, except for the diet evaluation. The foods in the Fastfood database are—I assume everybody has already guessed—50 of the most common fast foods, ranging from Egg McMuffin to Kentucky Fried Chicken. The reason that Fastfood is so much cheaper is that it is much less useful.

IPC Datadiet plans to get into the home market with Home Nutrition Guide, which will cost about $35 and have the Recall24 database. Unfortunately it will lack the evaluation portion of the program—the worst thing they could have cut out.

The following is a partial list of computer programs available that relate to diet. □

Diet
Classroom Instruction
Creative Computing
P.O. Box 789-M
Morristown, NJ 07960 $24.95

Compudiet
Weight Loss
Information Technology Systems
P.O. Box 2667
Sarasota, FL 33578 $19.95

Recall24
Medical Professional
IPC Datadiet
P.O. Box 28156
San Jose, CA 95125 $160

Datadiet
Medical Professional
Same as above $399

Fast Food
Same as above $35

The following programs are in use at major institutions. Interested readers should check with the institution concerning system requirements and availability.

Mini List Computerized Nutritional Analysis
University of California at Berkeley
Department of Nutritional Science
Database contains 235 foods, 50 nutrients, $75
Contact: Janet C. King, Ph.D.

DIANA (Dietary Analysis)
Los Angeles Valley College
Family and Consumer Studies Department
Database contains 635 foods, 10 nutrients, Price on request
Contact: Ida Jaqua

Little Falls Hospital
Clinton, NY 13323, $3000
Contact: Mary E. Kilby, RD

FOINANA, User's Guide to Food $, and LEAN
University of Minnesota
Agricultural Extension Service
St. Paul, MN $25-$100
Contact: Muriel S. Brink, Extension Nutritionist

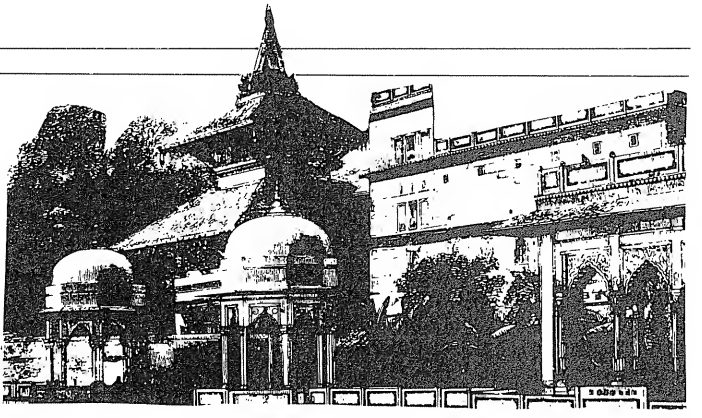
Pillsbury's Computerized Nutrition Exhibit
Pillsbury Company
Minneapolis, MN
Contact: Marlene A. Johnson, Consumer Relations

801	2043B	SHRIMP COOKED (FRENCH FRIED)	1.0	OZ
802	2045C	SHRIMP CANNED MEDIUM 2.5" LONG	10.0	SHRMP
803	2049C	SIRUPS MAPLE	1.0	CP
804	2049D	SIRUPS MAPLE	1.0	TBSP
805	2050B	SIRUPS SORGHUM	1.0	TBSP
806	2051D	SIRUPS TABLE BLEND CHIEFLY CORN LIGHT & DARK	1.0	CP
807	2051E	SIRUPS TABLE BLEND CHIEFLY CORN LIGHT & DARK	1.0	TBSP
808	2052C	SIRUPS CANE AND MAPLE	1.0	CP
809	2052D	SIRUPS CANE AND MAPLE	1.0	TBSP
810	2060	SOUPS ASPARAGUS CREAM OF CAN MADE WITH WATER	1.0	CP
811	2061	SOUPS ASPARAGUS CREAM OF CAN MADE WITH MILK	1.0	CP
812	2063	SOUPS BEAN WITH PORK CAN MADE WITH WATER	1.0	CP
813	2065	SOUPS BEEF BROTH CONSOMME CAN MADE WITH WATER	1.0	CP
814	2067	SOUPS BEEF NOODLE CAN MADE WITH WATER	1.0	CP
815	2069	SOUPS CELERY CREAM OF CAN MADE WITH WATER	1.0	CP
816	2070	SOUPS CELERY CREAM OF CAN MADE WITH MILK	1.0	CP
817	2072	SOUPS CHICKEN CONSOMME CAN MADE WITH WATER	1.0	CP
818	2074	SOUPS CHICKEN CREAM OF CAN MADE WITH WATER	1.0	CP
819	2075	SOUPS CHICKEN CREAM OF CAN MADE WITH MILK	1.0	CP
820	2077	SOUPS CHICKEN GUMBO CAN MADE WITH WATER	1.0	CP
821	2079	SOUPS CHICKEN NOODLE CAN MADE WITH WATER	1.0	CP
822	2081	SOUPS CHICKEN WITH RICE CAN MADE WITH WATER	1.0	CP
823	2083	SOUPS CHICKEN VEGETABLE CAN MADE WITH WATER	1.0	CP
824	2085	SOUPS CLAM CHOWDER (MANHATTAN) CAN WITH WATER	1.0	CP
825	2087	SOUPS MINESTRONE CAN MADE WITH WATER	1.0	CP
826	2089	SOUPS MUSHROOM CREAM OF CAN MADE WITH WATER	1.0	CP
827	2090	SOUPS MUSHROOM CREAM OF CAN MADE WITH MILK	1.0	CP
828	2092	SOUPS ONION CAN MADE WITH WATER	1.0	CP
829	2094	SOUPS PEA GREEN CAN MADE WITH WATER	1.0	CP
830	2095	SOUPS PEA GREEN CAN MADE WITH MILK	1.0	CP
831	2097	SOUPS PEA SPLIT CAN MADE WITH WATER	1.0	CP
832	2099	SOUPS TOMATO CAN MADE WITH WATER	1.0	CP
833	2100	SOUPS TOMATO CAN MADE WITH MILK	1.0	CP
834	2102	SOUPS TURKEY NOODLE CAN MADE WITH WATER	1.0	CP
835	2104	SOUPS VEGETABLE BEEF CAN MADE WITH WATER	1.0	CP
836	2106	SOUPS VEGETABLE W/BEEF BROTH CAN MADE W/WATER	1.0	CP
837	2108	SOUPS VEGETARIAN VEGETABLE CAN MADE W/WATER	1.0	CP
838	2110	SOUPS BEEF NOODLE DRY MIX MADE WITH WATER	1.0	CP
839	2112	SOUPS CHICKEN NOODLE DRY MIX MADE WITH WATER	1.0	CP
840	2114	SOUPS CHICKEN RICE DRY MIX MADE WITH WATER	1.0	CP
841	2116	SOUPS ONION DRY MIX MADE WITH WATER	1.0	CP
842	2118	SOUPS PEA GREEN DRY MIX MADE WITH WATER	1.0	CP
843	1-056	SOUR CREAM CULTURED	1.0	CP
844	2140A	SOYBEANS MATURE SEEDS DRY COOKED	1.0	CP
845	2144A	SOYBEANS SPROUTED SEEDS COOKED DRAINED	1.0	CP
846	2145A	SOYBEAN CURD TOFU 2.5" X 2.75" X 1"	1.0	PIECE
847	2146B	SOYBEAN FLOUR FULL FAT STIRRED	1.0	CP
848	2148A	SOYBEAN FLOUR LOW FAT STIRRED	1.0	CP
849	2149A	SOYBEAN FLOUR DEFATTED STIRRED	1.0	CP
850	2156A	SOY SAUCE	1.0	CP

Figure 3. Datadiet Database.

Better Than Adventure?

The Temple of Apshai



Roxton Baker

"The Temple of Apshai" is the third major game from Automated Simulations. Like its two predecessors, Starfleet Orion and Invasion Orion, it is available for both the TRS-80 and the PET. My direct experience with the Temple is on the TRS-80, but I believe the PET version is very similar.

The Orion games have, as of this writing, remained relatively unknown. They are not heavily promoted, have recieved little attention in magazines and are somewhat expensive. The Temple, being priced even higher, less familiar in concept and less well advertised, may suffer the same result.

This is unfortunate, as these are all games of unusual quality and significance. The three could more fairly be called "supergames" in company only with Scott Adams' excellent Adventure series. All of

Roxton Baker, 56 South Rd., Ellington, CT 06029.

these, the Temple in particular, are of great internal complexity and detail. Though not in itself apparent to the user, this sophistication results in a game action inherently more satisfying than that of the Star Treks, arcade shoots and graphics extravaganzas that form the body of light TRS-80 software.

This review will discuss how the game is built, how it works internally and how well it plays. It is impressive in all of these aspects, though its pacing suffers from some speed limitations discussed later.

It is not surprising that all of the supergames make use of a similar structure. One "master" program creates and manages the fantasy world in which you play. The master finds data for this scenario in one of several available data files. Thus, there are several available worlds. In each world, or adventure, the juxtaposition and attributes of locations, treasures, monsters, etc., can be totally different. However, the objective of the game and the structure (if not the detail) of the commands you issue stay the same. This technique allows the creation of many different games without requiring the authors to rewrite, and the players to relearn, the basic rules. One would hope for the further advantage, yet to be seen, of not requiring the customer to repurchase the master program.

The Temple of Apshai, then, is comprised of a master program ("DUNJONMASTER") and four data files. The vendor assures us that "dunjon" is an archaic form of our "dungeon," and not some "lo-klass kwikee" spelling. In fact, one of its meanings is "labyrinth," which applies here to the four different scenarios (Levels 1-4) that DUNJONMASTER can create from the data files.

Each level is a maze of connected rooms and hallways. Distributed throughout are various treasures, traps and monsters. These four separate labyrinths provide you with the opportunity to build the wealth and character of the adventurer whose identity you've assumed. Building up this character is, in fact, the entire object and purpose of the Dunjonquest system; the four-leveled Temple of Apshai is the first available exercise-ground.

Readers familiar with the popular game of "Dungeons and Dragons," and its derivatives, will recognize the idea immediately. The manual explains that the Dunjonquest system is a mechanization of D&D wherein the computer assumes the most important duties of the "Dungeon Master." These involve creating the dungeon and managing both the attributes of the characters, and the results of their play. Although lacking the full panoply of D&D, and its interplay of a multitude of character types, the computer simulation is complete enough to allow the rough translation of individual characters from the standard game into the Temple.

The concept of the player's "character" is most important to this game. Character is defined in terms of six numerical parameters, the values of which can range from 3 (abysmal) to 18 (super). These parameters are:

Intelligence

Intuition

Ego

Strength

Constitution

Dexterity

Each of these bears in its own way on the outcome of any event or action involving the character.

A seventh parameter, "experience," permanently modifies the effect of the six defining values. Experience points accrue from your character's exploits in the dungeon, and are used as a positive weighting factor. Thus, as you adventure, your character seems to become a little more intelligent, a little stronger, etc. This is the primary way that a character is built.

Also, while in the dungeon, you search for money, magical items and anything else that might help you. You may find a healing potion that will cure your wounds, or a magic sword more deadly than your factory piece. All of these can make you a stronger adventurer but they can be lost. Many treasures have a defined monetary value. At a central location (the Inn) you can sell them in order to buy better (but heavier) arms and more medicine. There you must deal with the Innkeeper.

INNKEEPER is a seperate program that runs on a startup and whenever you leave the dungeon. If it is the beginning of a new game and you do not have a character, the Innkeeper will create one. Your new character will be endowed with a random set of six defining values, and with a small amount of silver. Or, you may have left the dungeon in mid-game, to cash in your booty. In either case, it is at the Inn that you are able to bargain your available money for arms and equipment. A shrewd merchant, the Innkeeper will size up your character and deal accordingly. Don't be surprised at anything he says or does.

Upon leaving the Inn, you have a choice of dungeon levels to enter. New characters, inexperienced and ill-equiped, should venture only into the first level dungeon of the Temple (the Gardens). Good treasures are scarce here, but the monsters of this level are not as dangerous as on the other levels. It is when you enter the first room of the dungeon, and begin to meet monsters, that the action begins.

The TRS-80 screen display within the dungeon is schematic in form: a top view of the room around you and the connecting halls. A marker indicates your position and heading. You may also see the symbol of a treasure. On the right is displayed your current fatigue, wounds and the weight you carry. A number of single-keystroke commands are available for turning, movement and other actions.

The eye-popping instuctional manual ("Book of Lore") details the appearance and contents of each of the fifty-odd rooms(!) on each of the four levels (!!). It tells you of the many possible traps, and of the twenty treasures of each level and their value. It describes the twenty-three exotic monsters, differing in ferocity and toughness, that may attack you in the temple. And it includes everything else you would normally expect of a manual, in spades. This book even surpasses the high standards of the rest of the game— an excellent effort by the authors.

The mechanics of running the Temple programs on a TRS-80 should be mentioned. As of 9/79, the version you receive is on tape. You must load INNKEEPER to define your character, and then DUNJONMASTER to begin play. Both of these are lengthy Basic programs. INN-

KEEPER loads in nine data blocks (comprising one data file) with which DUNJONMASTER creates the dungeon level you requested. This all takes about ten minutes, just for tape loading. My copy was perfect, but tapes deteriorate with use. To make a backup copy you must be able to duplicate the 36 (total) data blocks. This will require utility software such as CLONE from Mumford. These are not complaints; there is a price to pay for running such a complex system of programs from tape.

Disk users must load the programs and data files onto disk. The necessary utility software is listed (but not supplied on tape). Then changes must be carefully made to INNKEEPER and DUNJONMASTER to allow for disk I/O. Disk users can save the state of dungeons that they leave in a new data file. Tape data files are hard to use, so the ability to create them was left out of the already crowded (16K) tape versions.

My TRS-80 Disk Basic would not read the Basic programs on the original tape. I had to first load them in under Level II Basic, and save them off to a blank tape. Then I read these new copies in under Disk Basic, and put them on disk. This procedure took time and created bugs. You are likely to have the same problem. If you want a true disk version of the Temple, contact Automated Simulations. They may make one available if there are enough requests.

Is this supergame a good game? Yes. In fact, it is one of the best games available for the TRS-80. It is interesting to learn, and both fun and challenging to play. The complexity of the action and the scenarios is beyond simple description, but is simply presented.

I might suggest a few improvements to the actual implementation of the Temple on the TRS-80. A higher-speed graphics technique (such as poke graphics) should be used in the drawing and redrawing of rooms. This occurs often and takes about thirty seconds each time, resulting in a significant decrease in overall game speed. The use of a TRS-80 clock speed-up kit (as offered by Mumford or Archbold) helps, but even when cut by one third, the delay is too long. Compare this with Adams' machine-language adventures, where location changes are instantaneous and you move as fast as you type.

Another nice addition to the Temple would be cassette-port sound effects for traps, monster attacks, etc. This would add to the fun, but there may not be room for such programming in 16K.

The documentation for the Temple needs no improvement. The beautiful 56-page manual explains everything, and does it well. This adds great value to the game. Beyond that, the friendly people at Automated Simulations are available by phone for questions and comments. And, they can spell— a skill unheard of in programmers.

That this is such a nice package indicates that the authors are gameplayers first, and computerists second. Today's serious gamers expect their wargames, role-playing games, etc. to be carefully designed, thoroughly researched, adequately tested, well-documented and handsomely produced. Today's computer hobbyist merely hopes for a game not totally dull, with a legible page of instructions. Both pay the same. The disparity in products arises from the twenty-year lead that the gamers have.

The price is high, but the Temple stands up well in a value comparison with the competition. You could easily do much worse. Computer game authors too often confuse programming effort with programming worth. The consumer discovers this error in judgement at his own expense. I would like to see lower prices on all these programs,particularly those supplied without such fine documentation. People just cannot afford all of the good software that they should have.

The Temple of Apshai is available for TRS-80 (reviewed here) and for PET. It is priced at $24.95. The TRS-80 tape version runs in 16K; the disk version in 32K. They are identical. The PET version requires 32K. Dealers may have it, or you can order directly from:

Automated Simulations
P.O.Box 4232
Mountain View, California 94040
415-964-8021

Other vendors mentioned in this review are:

Scott Adams' Adventures
Creative Computing Software
P.O.Box 789-M
Morristown, NJ 07960
201-540-0445

Mumford Micro Systems
Box 435-C
Summerland, California 93067
805-969-4557

Bill Archbold
TRS-80 Speed-up Kit
105 SnyderDrive
Mather, California 95655
916-363-3627 □

4K Microchess

Attention, Chess Phreaks!

Les Palenik

MICROCHESS is written in Z-80 machine language and it fits in 4K of memory, so you can run it on the smallest TRS-80 system. It can be loaded into the TRS-80 computer using the standard CLOAD.

It is advisable to clear the screen before typing CLOAD. The loading of the program takes a little bit longer than one would expect after loading standard programs written in BASIC.

Since this program is written in machine language it will automatically start executing after successfully loading.

First, all available options and instructions will be displayed on the screen. Take a good look at this display, or even better, copy it on a piece of paper, because once you press "ENTER," you won't see it again.

After you press "RETURN" a graphic depiction of the chessboard will appear on the left side of the screen (approximately 2/3 of the screen is used for the chessboard).

The right hand side of the screen is used for communication between the player and the computer. All the messages and prompts will be printed on this part of the screen.

The player can select the color, but not the side of the board. The computer's pieces will be displayed always at the top and the player's pieces at the bottom of the screen.

There are three different levels of play, ranging from beginner to an expert. You can decide on the level of play by typing:IQ=1,2, or 3. Usually one would select level 1 or 2, since the program responds quite quickly playing at this level. Level 3 is the best level of play, but is considerably slower and some players may lose their patience playing at this level. You can switch the levels of play anytime between the computer's moves.

There is a very interesting feature in this program which will allow the player to reverse the sides. It is the exchange command and you can execute it by typing an "X." Both sides will be reversed in a fraction of a second and you can use it to let the computer play a move again itself. Well, I admit it's cheating, but it can be quite interesting to see how the computer analyzes the opponent's side, and it can be used for simulation and learning how to play a better game of chess. In another extreme, one could play a game against himself by using the same command.

Once the program has been loaded, it will disable the break key, so if you want to break, the only way is to switch off the machine. Of course, then you'll lose the program and you have to load it again. This seems to be a rather clever protection of the program.

The program consists of the chess-playing logic and the graphic driver which displays the chessboard. The graphic driver is somewhat limited by TRS-80 video-display and its resolution (48x128 addressable locations), but all pieces on the screen can still be easily recognized.

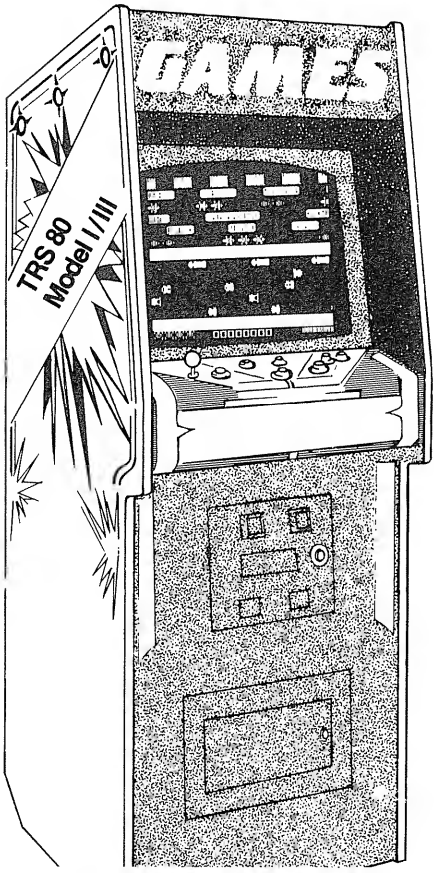
The moves (especially the computer's) are done in a very neat way. When it is time for the computer to make a move, the cursor is moving on the screen, to indicate that the computer is still "thinking." Once the computer decides on its move, the particular piece will be flashed several times to draw the attention of the player and then it will be moved to its new location.

In summary, I think this is a very interesting program which will bring you many hours of enjoyment and, at the same time, improve your chess game. It demonstrates in a nice way the capabilities of TRS-80 and all in all, is an excellent program to have in your library. I would highly recommend it to you. ■

Les Palenik, 25 Silversprings Blvd., Suite 512, Scarborough, ONT M1V1M9, Canada.

Games for the TRS-80

Harry McCracken & Owen Linzmayer



Very few computers are advertised as game machines, but we all know that more home computers are being used to battle alien forces than to balance checkbooks. The TRS-80, although a relatively primitive computer for game playing, has a wealth of arcade software currently available for it. Reviewed here are the six best TRS-80 games I have seen during the last month.

Owen Linzmayer is a staff writer for *Creative Computing* magazine.

Harry McCracken, 47 Carleton St., Newton, MA 02158.

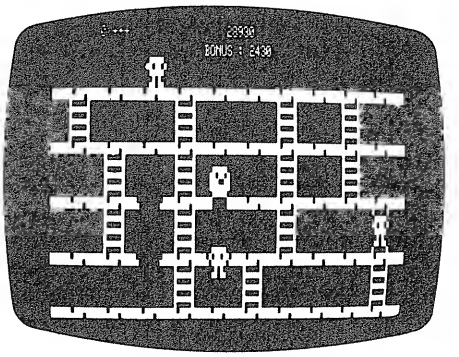
Apple Panic

In *Apple Panic*, you play the role of a space farmer whose crop of apples has mutated and turned against him. To combat these creatures, you dig holes in the platforms that make up the screen. If an apple falls into a hole, you must bash it until it plummets to its death on the concrete below. If an apple touches you, you lose a life.

The *Apple Panic* packaging promises voice and sound effects. This is a bit misleading. The only time the computer speaks (through the AUX port), is when it displays the banner page. At this time, it says only two words, "Apple Panic." Most of the other games on the market that advertise voice effects offer a much larger vocabulary.

In addition to the printed instructions, *Apple Panic* has thorough documentation accessible from within the program. This includes a scoring table and a review of the controls. The game can be played by one or two players with either the keyboard or a joystick.

The playscreen is divided into five platforms that are connected by ladders.



The position and length of the ladders is random, allowing for an almost infinite number of board configurations.

The graphics are detailed, and the animation is very clear. When one object passes in front of another, it overlays the object in the background, rather than blocking it out.

The sound effects in *Apple Panic* are sparse, but come at appropriate times. The computer breaks into a rich, full-bodied musical number as an aural reward for completing a screen.

If you set a high score, you may enter your name or initials (up to 10 characters) to be added to the high score table. These names and scores are saved permanently on the disk version, and are displayed on the instruction screen.

SOFTWARE PROFILE

Name: Apple Panic
Type: Arcade
System: 16K Mod I/III TRS-80
Format: Tape/disk
Language: Assembly
Summary: Enjoyable ladder-climbing game
Price: $19.95/$24.95
Manufacturer:
FunSoft Inc.
28611 Canwood St.
Agoura, CA 91301

Apple Panic from Funsoft may be well on its way to the top of the charts.

Crazy Painter

Crazy Painter brightens up a TRS-80 software library just as a fresh coat of paint brings new life to a drab room. It is, as far as I know, an original arcade game unlike any other.

Your job is to maneuver a paintbrush around the playfield, trying to "white out" the entire screen. Your mission is hindered by a group of pests that remove the paint that they walk, slither, and crawl over.

Some of the creatures that you encounter are deadly to the touch, others may be run over by your paintbrush. After you finish painting the screen, you advance to a special bonus round in which all of the monsters are vulnerable to your paintbrush. As the game progresses, the action increases as more aggressive foes attack your paint job.

Crazy Painter is a very professional program designed with user-friendliness in mind. There are three pages of internal documentation complete with animation. The game supports both one- and two-player games. If you want to skip the easy rounds of play, don't worry, there are ten selectable levels of starting difficulty. In addition to this, each player can choose his own level.

Crazy Painter is written entirely in machine language by Robert Pappas, author of *Bounceoids* (see review, December 1982 issue). The major attraction of this game is that instead of being represented on the screen by a little graphics character, you actually create the graphic images on the computer screen. The movement of all of the elements in *Crazy Painter* is remarkably smooth, even at high speeds.

I like *Crazy Painter* because as the game progresses in difficulty levels, different creatures are brought into play, thus adding variety to the game. The bonus rounds that you encounter after every screen break up the tension and offer a chance to augment your score considerably.

The one complaint I have with *Crazy Painter* is that there are very few sound effects. Additional sound effects would greatly enhance this game. It has been proven by coin-op game manufacturers that over 50% of the appeal of a game depends on the audio output. I hope more TRS-80 programmers take note of this fact.

Crazy Painter offers a refreshing break from shoot'-em-up games. The idea is novel, and Robert Pappas deserves credit for taking the time to do some innovative programming.

SOFTWARE PROFILE

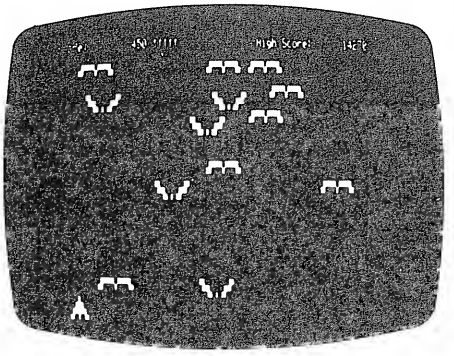
Name: Crazy Painter
Type: Arcade
System: Mod I/III TRS-80
Format: Tape/disk
Language: Assembly
Summary: Very nice original game
Price: $15.95/$19.95
Manufacturer:

The Cornsoft Group
6008 N. Keystone Ave.
Indianapolis, IN 46220
(317) 257-3227

Demon Seed

During the summer of 1982, the theaters were filled with horror films. When I first heard of *Demon Seed*, I mistakenly thought it was the title of a new "insane convict murders entire town" movie. *Demon Seed* is, in fact, a TRS-80 adaptation of Centuri's coin-op arcade game Phoenix.

In *Demon Seed* you control a lone space fighter that traverses the bottom of the computer screen. This ship is equipped with an unlimited supply of ammunition and a protective shield that



can be activated for short periods of time.

The enemy takes the shape of large winged bats and demons. Each attack is made up of five separate waves. The first two attack waves consist of bats flying in formation. During the third and fourth waves, you are confronted by swooping demons that you must hit dead-center to destroy. On the fifth wave, you face the demon attack ship. Before you can shoot its pilot, you must blow a hole through both the belly of the ship and the revolving rim. After you destroy this ship, a new attack wave begins. If you survive two attack waves, you get to try your hand at a special challenge round.

Demon Seed is designed for only one-player. Until you become familiar with the workings of the game, it is difficult to attain high scores. If you set one of the top ten scores, you may enter your name (up to 20 characters) to be saved permanently on disk.

The ship is controlled by using the keyboard. In addition to the game controls, there are a few special keys that you should be aware of. You can pause the game at any time by pressing P. If you want to abort the game entirely, hold down both the BREAK and CLEAR keys. To turn off the sound effects, press the BREAK key. Options such as this take little time to add to a program, but they make a game much more friendly.

SOFTWARE PROFILE

Name: Demon Seed
Type: Arcade
System: 16K Mod I/III TRS-80
Format: Tape/disk
Language: Assembly
Summary: Excellent adaptation of Phoenix
Price: $19.95/$24.95
Manufacturer:

Trend Software
Box 741
Bloomfield Hills, MI 48013

The animation in *Demon Seed* is very good. Attacking creatures flap their wings and drop bombs as they swoop down at your ship. At times, game elements flicker, thus detracting from an otherwise excellent graphic display.

Anyone who enjoys playing Phoenix will find *Demon Seed* a game well worth his money.

Frogger

After Frogger, from Sega Electronics, proved itself in the arcades, manufacturers began clamoring for the home rights to the game. The Cornsoft Group acquired the rights to produce a TRS-80 adaptation of Frogger, and they did a

great job. This licensed version is the best I have seen.

In *Frogger*, you control a small frog that you must maneuver across a bustling highway and past a rushing river. You can move in any of the four compass directions using either the keyboard or a joystick. You must avoid traffic, snakes, crocodiles, and diving turtles. If you get five frogs safely onto their lilypads on the far side of the river, you advance to a more difficult level.

SOFTWARE PROFILE

Name: Frogger

Type: Arcade

System: Mod I/III TRS-80

Format: Tape/disk

Language: Assembly

Summary: Licensed version of Frogger, coin-op game from Sega/Gremlin

Price: $19.95/$22.95

Manufacturer:

The Cornsoft Group
6008 N. Keystone Ave.
Indianapolis, IN 46220
(317) 257-3227

The sound effects of *Frogger* are every bit as crisp and whimsical as those found in the arcade. The program beautifully emulates frog sounds of the coin-op game.

One problem many game designers face is how to fit an arcade game onto a computer screen. Remember, most video games have screens that are longer than they are wide; the opposite is true of the TRS-80. Rather than squeeze the playfield down to size, programmer Robert Pappas simply split it in two. When a game begins, you see only the highway. If you reach the other side, the river section scrolls down into place. This is a new, effective way to handle an old problem.

This split-screen technique provides for much more detailed graphics than the versions I have seen that use only one screen. Even with the increased detail, it is easy to lose sight of your frog on the river screen, especially when you are riding on a log. If it were easier to differentiate between graphic elements, the game itself would be greatly enhanced.

One or two people can play *Frogger*, each chosing his own difficulty level (0-4). If a high score is set, you can enter a name or message (up to 17 characters) which will be saved on the disk.

The thing that bothers me about *Frogger* is that the controls aren't as responsive as I would like them to be. At times you must wait before you can move. This is maddening and results in many miscalculated jumps—usually into the grillwork of an oncoming car and flat frogs.

Frogger lives up the standard for arcade programs set by the Cornsoft Group. I have never been very fond of the coin-op *Frogger*, but I recommend the TRS-80 version to anyone who is.

Mad Mines

I have heard it said that borrowing from one source is plagiarism, but borrowing from two sources is research. If this is true, then *Mad Mines* is an excellent piece of software research. *Mad Mines* combines elements of two popular Apple II computer games: *Ceiling Zero* and *Space Eggs*.

In *Mad Mines*, you are in charge of a small space cannon that slides along the bottom of the screen. Your mission is to survive as long as possible while eliminating the mines that infest inner-space. If you shoot a mine, its occupant bursts forth and dives down to attack you. Anything that passes below the force field is restricted to this ever-shrinking space. Each time you kill a flock of mines, a new group appears and the force field is lowered. As the action speeds up, your margin for error diminishes rapidly.

You face a total of four different aliens, each with its own pattern of behavior. The most dangerous of these are the ones encountered on the fourth attack wave. If you shoot an egg during the fourth wave, the creature drops straight down at you. If you don't hit the little bugger before it reaches the bottom of the screen, you are dead. This is a perfect example of a "be perfect or die" situation.

SOFTWARE PROFILE

Name: Mad Mines

Type: Arcade

System: 16K Mod I/III TRS-80

Format: Tape/disk

Language: Assembly

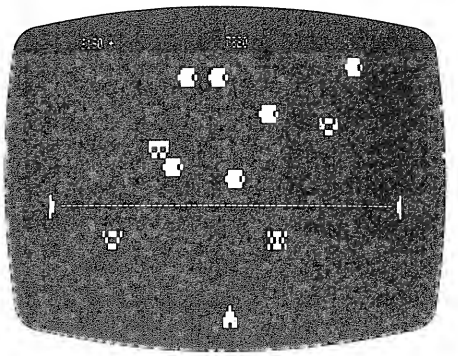
Summary: Combination of two Apple games

Price: $19.95/$24.95

Manufacturer:

Funsoft Inc.
28611 Canwood St.
Agoura, CA 91301
(213) 991-6540

The graphics in *Mad Mines* are absolutely state-of-the-art. The animation of the aliens is excellent and nothing



flickers, even when the screen is filled with moving elements. One thing that I do find annoying is that the stars in the background look a lot like the alien bombs. This is confusing and could be corrected by eliminating the scrolling stars.

Rather than simply switch from one screen to another, *Mad Mines* has a variety of transitions that dazzle the player. The graphics demonstrated during these transitions are original and accompanied by sound effects.

Yves Lempereur, author of *Mad Mines*, did an excellent job of creating an all-around enjoyable product. When I showed the game to my co-workers, most of them commented on how much the style resembled a Big Five program. If you know anything about the TRS-80 game world, you know how respected the Big Five name is. To have one's program compared to a Big Five game is a great compliment.

Mad Mines is one of the most professional arcade games available. It is constantly challenging because it requires both dexterity and strategy. I have a special bunch of games that I keep on-hand to show off to friends—*Mad Mines* has earned its place in that limited group.

Outhouse

So, you have just joined the Intergalactic Defense Force. The recruiter promised that you'd see Mars, maybe even Venus. Your orders came in today, and guess what—you've been placed in charge of defending an outhouse located somewhere in Iowa. It's a cushy job, but someone has to do it.

For some strange reason, the enemy has launched a full scale attack on the outhouse. Some of the aliens want to destroy the outhouse, others want to destroy you. All the while, vandals and squatters are trying to use up your limited supply of toilet paper.

You control, via a joystick or the keyboard, a laser-equipped fighter that can move and shoot in eight directions. Running into anything, as well as being shot by an alien, causes you to lose a ship. When you run out of ships or toilet paper, the game is over.

The action starts off slow, but becomes challenging after the first three attack waves have been disposed of. As you progress, the game brings more aliens into play. There is a total of seven different game elements that are programmed to eliminate you, each in its own special way.

Although *Outhouse* is actually a shoot'-em-up game, it has a strange scenario which elevates it above the usual death and destruction arcade game. Both the game concept and the graphics are original, and there is enough variety here to satisfy even the most jaded game player.

The sound effects are crisp and add much to the program. In addition to the normal complement of space war sound effects, the disk version of *Outhouse* is enhanced with voice effects. During the introduction and intermissions, the computer speaks through the AUX port. The voice is a bit coarse, but everything that is said is understandable.

One or two players can play *Outhouse*, alternating turns at the controls. If you set a high score, you are allowed to add your name to the scoreboard. There are two high score charts: all-time, and daily. The top eight all-time scores are saved to disk, whereas the daily scores disappear when the system is turned off.

If *Outhouse* wasn't a good program, it would at least deserve credit for being original. Luckily, it is a great program. I recommend *Outhouse* to anyone looking for a fresh idea in game playing, as well as the hardened arcade addict.

SOFTWARE PROFILE

Name: Outhouse
Type: Arcade
System: 16K Mod I/III TRS-80
Format: Tape/disk
Language: Assembly
Summary: Interesting new game
Price: $15.95/$19.95
Manufacturer:

Soft Sector Marketing Inc.
Box 340
Garden City, MI 48135
(800) 521-6504

Bounceoids

SOFTWARE PROFILE

Name: Bounceoids
Type: Asteroids variation
System: TRS-80 Model I/III
Format: Cassette/disk
Language: Machine
Summary: Very good arcade game
Price: Tape $15.95, disk $19.95
Manufacturer:

The Cornsoft Group
6008 North Ketstone Ave.
Indianapolis, IN 46220

In *Asteroids*, leaving the edge of the screen in any direction results in the player reappearing on the opposite side. This feature is called "wrap-around."

What do you think would happen if you took the popular arcade game, Asteroids, and removed the wrap-around function? *Bounceoids*, the new machine language program written by Robert Pappas, is very similar to that, but much, much more.

The concept of a game of Asteroids without the wrap-around is novel enough, but *Bounceoids* has a slew of added features to tempt the arcade addict in us all. *Bounceoids* offers the traditional features such as bonus ships, high scores, multiple skill levels, a two-player mode, and fast graphics. It also has *many* different challenge stages, sound effects, two sets of keyboard controls, and a variety of strange bounceoid creatures out for blood—your blood.

Bounceoids comes on a self-booting disk that works on both the TRS-80 Model I and III. The internal documentation is well written and explains the controls, point system and how the game is played. You are then prompted to enter the number of players (1 or 2) and the skill level (0-9). Regardless of the level you enter, it will increase at every 10,000 points, until eventually it reaches nine.

If you are playing a two-player game, each player may choose a different starting skill level. This is similar to the start-up sequence found on the coin-op game, Tempest by Atari. It is nice to see programmers making their games function more and more like true arcade games.

The graphics on *Bounceoids* are fast and smooth—even when the screen is filled with objects in every possible direction.

Controlling your ships is easy. You have the choice of using one of two different sets of keyboard controls or a joystick. One of the sets of keyboard controls is identical to those found in Big Five's *Super Nova* program. This makes it easy for those of us who have become accustomed to those controls.

By using an amplifier connected through the cassette port, you will be treated to sound effects. Using sound effects is the only way to really know how much shield power you have left. Each time your ship is struck while you are using the shields, you lose power. By listening to the sound when you engage the shields, you can tell how many more collisions your ship can survive. At every 10,000 points you are awarded a new ship.

Since *Bounceoids* does not increase significantly in difficulty once you reach skill level nine, it is theoretically possible to play forever, as long as you keep winning new ships.

Bounceoid boulders do not break into smaller pieces when shot; but it takes four hits to eliminate one. There are also Tiny Bounceoids Clusters in which many, many rocks all occupy the same space and follow the same path. These can become quite confusing, not to mention difficult to destroy.

In addition to all this, there is a special creature that when hit, breaks up into four space stations. Each station goes on one of the four edges of the screen and then follows your every move, occasionally shooting at you. These guys get to be a real pain in the asteroid.

Another thing *Bounceoids* has, that Asteroids doesn't, is Challenge Stages. At every 20,000 points you are treated

to one of these showdowns. A string of aliens come on-screen and proceed to swirl around in strange flight patterns. These creatures can't kill you, but they sure provide good target practice for you. There are many different challenge stages and unless you are *very* good, you will not see them all. This variety gives you an incentive to keep on trying just to see what will happen next — a feature that many games lack.

I recommend *Bounceoids,* it's fun, it's different, it's good. What more can I say? *Bounceoids* is a well written arcade game. It is a good example of how a clever programmer can take a rather bland game concept like Asteroids, and turn it into a great new game in its own right. If you are sick of all the different maze type games, *Bounceoids* offer a refreshing change. Give it a try, you'll be coming back for more.

Video game madness is sweeping America. News magazines publish cover stories on video games, bookstores sell books on how to beat them, and chart-topping records are recorded about them. It is not surprising, then, that many software publishers offer versions of the most popular arcade games for home computers.

Not surprising, at least, in the case of such computers as the Apple II, Atari, and TRS-80 Color Computer. Those machines all have high-resolution color graphics, sound routines in ROM, and paddles or joysticks. Not so the TRS-80 Model I and Model III. They have low-resolution black and white displays, no sound routines, and no game controllers. That is why it is remarkable that there are quite a few good arcade-style games for the TRS-80 Model I and III.

Let's look at four games which make especially good use of the limited potential of the TRS-80 as an arcade game machine.

Planetoids

Planetoids, published by Adventure International, is, as the name suggests, inspired by the classic Atari arcade game Asteroids: you control a ship which shoots at floating asteroids and enemy spacecraft.

Adventure International has released versions of *Planetoids* for both the TRS-80 and the Apple II. Amazingly, the TRS-80 version is by far the better of the two. Greg Hassett, known primarily for his adventure games, has written one of the best arcade-style games available for any home computer with or without, high-resolution color graphics.

Many home computer games which try to simulate arcade games get the graphics of the original down, but *Planetoids* is one of the very few which do a good job of simulating the *timing* of the original. That means that playing *Planetoids* is much like playing Asteroids.

Planetoids is enhanced by its special features and playing modes. Like many games, it has more than one skill level, but it also has the option of saving high scores to tape, the ability to abort a game in progress, and the ability to "freeze" a game, so that you can answer the phone and come back to an uninterrupted game.

SOFTWARE PROFILE

Name: Planetoids

Type: Arcade game

System: TRS-80 Model I and III, 16K

Format: Cassette

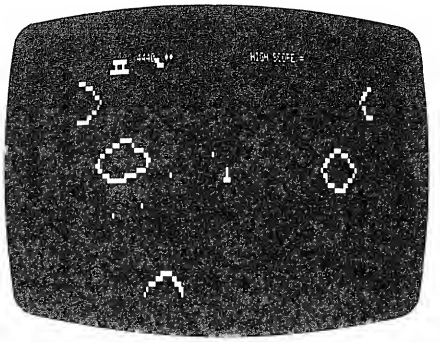
Summary: Excellent version of Asteroids

Price: $19.95

Manufacturer:

Adventure International
Box 3435
Longwood, FL 32750

The special modes include "dogfight" mode, in which there are no asteroids and you fight one-on-one with the enemy ships; "cruise" mode, in which the asteroids move very, very slowly, so that you can practice steering your ship; and "practice" mode, in which game play starts out at a higher skill level than normal—*Planetoids*—gets harder as you play—and your score is not recorded on the high score chart.



Planetoids.

All of these features ensure that *Planetoids* will hold your interest for a long time.

I do have two small complaints about the game. It does not have sound, and there is no two-player mode. Neither of these is a big problem, but the game would be better if it had sound and two-player capability, as most TRS-80 arcade games do.

Alien Defense

Another game which does a good job of adapting an arcade game to the TRS-80 is *Alien Defense* from Soft Sector Marketing. Inspired by Williams's arcade hit *Defender*, it is probably the most difficult arcade game around for the TRS-80.

As in *Defender*, you control a ship which flies above a horizontally scrolling landscape and shoot down many kinds of aliens, some of which fly down and kidnap people who stand on the landscape.

The graphics in this game aren't as good as those in *Planetoids*; they flicker and the movement is a bit too jerky. That isn't surprising, though; there is a great deal of movement going on in this game.

Larry Ashmun, the author of *Alien Defense*, has done a good job of putting the ship controls onto the TRS-80 keyboard. While you must manipulate six keys to control your ship, the pattern in which the keys are arranged makes it easy to do.

The Model III version of *Alien Defense* has an added feature: the special charac-

SOFTWARE PROFILE

Name: Alien Defense

Type: Arcade game

System: TRS-80 Model I and III 16K cassette, 32K disk

Format: Cassette or disk

Summary: Good version of Defender

Price: $19.95 cassette; $24.95 disk

Manufacturer:

Soft Sector Marketing
6250 Middlebelt
Garden City, MI 48135

ter set of that computer is used, so that the little men look like men, and the alien ships look more like alien ships than they do on the Model I. This is the only Model III game I know of which uses the Model III graphics so well.

All in all, *Defender* fans should be very pleased with *Alien Defense*. Like *Defender*, it is a very challenging game, so beginning arcade game players might find it a bit complicated at first.

Defense Command

Big Five Software has released a steady stream of superb games, over the last couple of years. One of their most recent games, *Defense Command*, is their best yet. It might be described as *Space Invaders* crossed with *Defender*. You control a base at the bottom of the screen, *Space Invaders* style, and protect fuel cells from attacking aliens who try to steal them.

SOFTWARE PROFILE

Name: Defense Command
Type: Arcade game
System: TRS-80 Model I and III, 16K cassette, 32K disk
Format: Cassette or disk
Summary: Big Five's best yet
Price: $15.95 cassette; $19.95 disk
Manufacturer:
Big Five Software
Box 9078-185
Van Nuys, CA 91409

Just about everything which has distinguished Big Five games in the past is in *Defense Command*. Explosions are more realistic; the high score chart lets you type in your full name, rather than just a few characters; and the infamous Big Five Flagship is more evil than ever. Like *Robot Attack*, *Defense Command* talks through the tape port, but the speech is far clearer.

While *Defense Command* is a difficult game to master, it is not complicated; beginners may play terribly, but they will understand and enjoy the game. And because *Defense Command* requires more true strategy to master than the other Big Five games, even advanced arcade game enthusiasts will find it a challenge.

If you have trouble formulating a strategy to play the game well, watch the demonstration mode; the computer plays the game very well, and it is easy to pick up pointers from watching it play itself.

Armored Patrol

In writing a TRS-80 version of an arcade game the biggest problem is usually the creation of an acceptable graphics display. This is a difficult task with almost any hi-res game (what current arcade game isn't hi-res?) but, when a TRS-80 version of a vector graphics game is desired, it borders on the impossible—or so I thought. *Armored Patrol* is an adaption of the popular arcade game Battlezone, which uses vector graphics.

Written by Wayne Westmoreland and Terry Gilman, *Armored Patrol* is an amazing demonstration of how far creative programming can stretch the limited graphics capabilities of the TRS-80. In *Armored Patrol* the player assumes the guise of a tank commander positioned in a desolate battlezone. This death field is littered with indestructible blockhouses which serve as obstacles during a shoot out. Hidden in this barren land is an endless supply of enemy tanks and killer robots bent on destroying the player.

Armored Patrol allows one or two players, alternating turns, to test their abilities against the enemy forces. The program constantly displays and updates the players' scores and also keeps track of the high-score. The sound effects are an added touch to enhance the captivating 3-D screen display.

Never before had I imagined that my TRS-80 was capable of such a realistic simulation.

The object of this game is survival. By weaving skillfully among the blockhouses, the player tries to gain position from which he can safely blast the enemy without risking destruction himself.

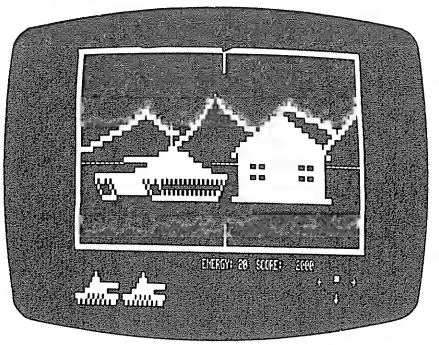
An enemy tank can combine forces with a killer robot to attack simultaneously from different vantage points. The player is aided by a radar scope that gives him a rough idea where the enemy is in relation to his position, but visual confirmation is usually required for accurate shooting.

As the enemy tank approaches it grows in size and its surface details become clearer right up to the point where you are looking down the cannon barrel. By then it is usually too late to do anything but pray.

Armored Patrol is a classic in every sense of the word. Its value as a simulation is tremendous but it is also a superb arcade game. Although the controls are a bit difficult to master, it is a wonderful program for game players with even modest coordination. I recommend this package without reservation.—*O.W.L.*

SOFTWARE PROFILE

Name: Armored Patrol
Type: Arcade simulation
System: 16K Model I or III TRS-80
Format: Cassette
Language: Machine
Summary: Superb arcade game
Price: $19.95
Manufacturer:
Adventure International
Box 3455
Longwood, FL 32750



Armored Patrol.

Super Nova

Super Nova is a TRS-80 version of Atari's popular arcade game, Asteroids. Nova was the link between arcade and TRS-80 for which Asteroids addicts had been waiting. I ordered Nova expecting a good, but slow game that I would be able to tolerate. Boy, was I ever surprised: Nova exceeded my expectations. In fact it was so good, I couldn't wait for the next game from Big Five to appear on the market. To date, Big Five has marketed five quality arcade-

type programs all of which are written in machine language by Bill Hogue. Certain features are common to all of the games. They all:

1. Allow two players to compete against each other (one at a time).
2. Keep track of the top ten scores and the initials of the scorers.
3. Give the player a starting allotment of three ships.
4. Award a free bonus ship for each 10,000 points scored.
5. Offer infinite play; as long as one ship is left intact, you may play forever.
6. Have a built-in demonstration mode.

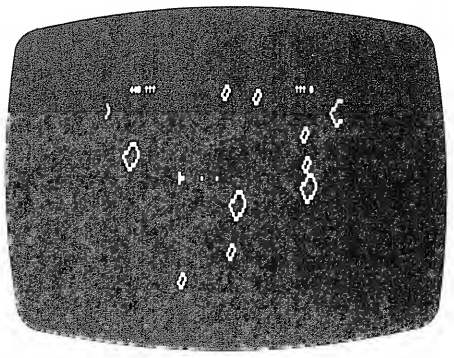
Super Nova is a fast-moving game which requires a great deal of dexterity and quick reflexes.

The program starts with a graphics display which catapults you through space. When the game begins, your ship is situated in the center of the screen. Huge asteroids drift aimlessly through your sector. To acquire points you must shoot at and destroy them. Sounds easy enough, but there is one small catch: the meteors come in three sizes. If you hit a large one it splits into two medium-sized asteroids, which when shot, break into two of the smallest size. These small meteors move quicker and are, thus, harder to destroy. They also carry a higher point value than the larger ones.

You can be killed by letting either a stray asteroid or an alien ship collide with your fighter, or by getting hit by enemy fire. There are five types of hostile ships which try to do you in. These range from the slow and clumsy JLK (Jidyan Land Kruiser) to Big Five's trademark, the Flagship. The Flagships carry a secret-weapon, an omnipotent laser bolt cannon which fires a realistic lightning bolt.

To avoid death, you fire at asteroids and enemy ships to destroy them before they get you. Your fighter rotates freely on its own axis a full 360° in 45° increments and is equipped with rocket-thrusters which propel you at amazing speeds. If by thrusting, dodging or shooting, you cannot avoid a crash, there is still one desperate last hope, Hyperspace. Entering Hyperspace is dangerous because your place of re-entry is unknown.

Super Nova is as challenging as the arcade version, but is unfortunately missing sound effects. The program has smooth, non-jumpy graphics and is definitely a must for the TRS-80 owner who enjoys Asteroids.



Galaxy Invasion

Galaxy Invasion is the TRS-80 version of Midway's famous "Galaxian" arcade game.

The game begins with your ship sitting directly under a large convoy of invaders. As you glide left and right, methodically destroying the alien ships, small groups of invaders break away from the flock and dive toward your ship. You can almost hear the little suckers inside screaming, "Banzai!" If you shoot a kamikaze alien it is worth twice as much as if it were flying with the pack. If the attacking alien doesn't get shot or ram your ship, it will wrap-around to the top of the screen and drift back into its original position.

SOFTWARE PROFILE

Name: Galaxy Invasion, Attack Force, Super Nova, Cosmic Fighter, Meteor Mission Two

Type: Arcade Games

System: 16K TRS-80 Model II or III

Format: Disk or Tape

Language: Machine Language

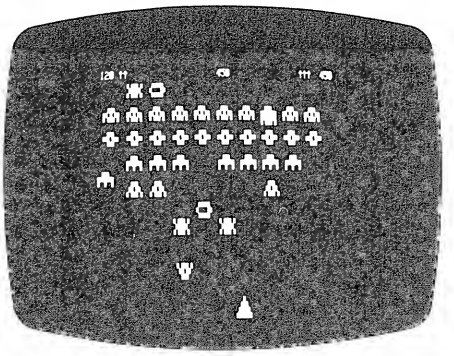
Summary: Top games with high quality graphics

Price: $15.95 each on tape, $19.95 per game on disk

Manufacturer:
Big Five Software
P.O. Box 9078-185
Van Nuys, CA 91409

The Flagships in this game always attack with escorts if possible. The more screens you clear, the more Flagships appear. The Flagships occupy the uppermost row of the convoy, as far away from your laser cannon as possible.

Your demise can take place if you collide with an attacking ship or get hit by a dropped bomb, or you may meet a special death. Occasionally during play, the words "Flagship Attack Alert" flash in the center of the screen. This is to warn you that in a few seconds, the cruel Flagships will unleash their full fury on your little fighter. The only way to avoid being killed is to shoot



a Flagship quickly before time runs out. If you don't do this rapidly enough, the Flagships will one at a time open fire on you with their laser bolts, and they always hit—always!

The fleet continues to slide across the screen until you have destroyed all of the alien intruders. The next convoy is a bit quicker, slightly more intelligent and tougher to kill.

Galaxy Invasion, although a big seller, is slower than the other Big Five programs. After 250,000 points the game doesn't increase in complexity to any measurable degree, but is nevertheless quite enjoyable and will keep you fighting over who gets to use the computer.

Attack Force

Attack Force is the TRS-80 version of Exidy's "Targ" arcade game. The object of this game is to maneuver your ship through a death-field of allies formed by 35 blocks arranged in a 7 x 5 matrix.

At the beginning of each board, your ship starts out in the lower right corner and one of the eight enemy ramships is situated at the top of every vertical alley.

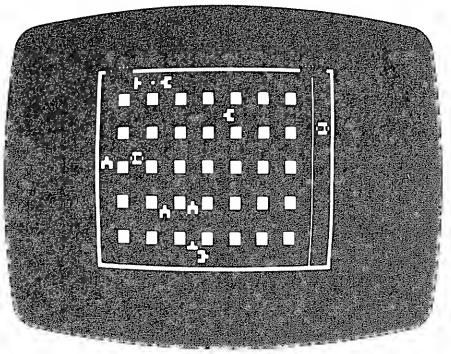
You must bob and weave through the back streets and shoot the ramships until the pathways are empty. These ramships cannot return your fire, but can as their name implies, "ram" you to death.

On the right of the field is a thin vertical side street from which your ship is restricted. The feared Flagships are stored

there. Once in a while the Flagships will shoot one of the ramships and thus transform it into either another Flagship or a ship identical to yours (if you collide with or shoot this mirror image, you are, in reality, killing yourself).

As long as there are still ramships in the maze, Flagships have no special ability, but when all ramships are destroyed, the extra Flagships from the storage area enter the death-field. The commander of these has the ability to fire laser bolts at you from any point in the maze. If you are hit six times by the same Flagship, you are blasted into stardust.

After you have killed all of the enemy ships, the battle is over and you are treated



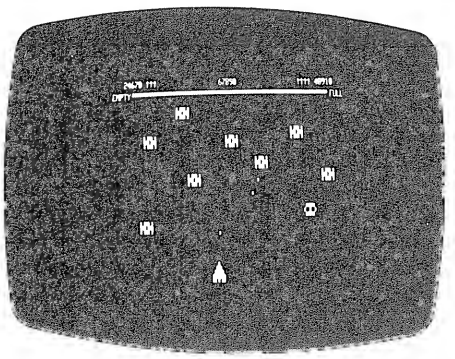
to a dazzling graphics display that fans across the screen. You are then awarded bonus points that increase by 1000 after every battle. The value of each ramship increases by 10 points in the same manner.

It is possible for this game to last forever, but reaching anything over 75,000 points is quite a feat. Attack Force is probably the most challenging of the five programs. If you enjoy beating a game, then try this one on for size, it doesn't give up easily!

Cosmic Fighter

This program is unique among the five because it isn't copied from any one arcade game, although there are traces of Astro Fighter embedded in it. Bill Hogue deserves special thanks for his ingenuity in combining bits and pieces of various games with some ideas of his own to form an exciting home computer game.

As with most arcade games, your ship travels across the bottom of the screen. At the top of the screen is a graphic fuel gauge. If it reaches empty your fighter's fuel tanks won't be able to stand the change in cosmic pressure and will implode, ending



the game. Moving and shooting gradually eat away at your fuel reserve.

The first group of aliens sways up and down as they slowly descend toward your ship. You must shoot them as quickly as possible with minimum firing and maneuvering so that you conserve fuel—remember, there is an energy crisis in the cosmos also!

After killing the first set of invaders, another group appears. Four separate groups of aliens comprise an attack wave. When you finally kill off the entire wave, a mothership that resembles a checkered taxicab enters from the right. You must now attempt to dock your ship in the niche in the bottom of this station. If you dock successfully, your ship will be refueled and you will be ready for another attack wave. Hint: a piece of tape correctly positioned on your CRT will aid you in docking quickly.

After each of the first four blitz attacks, the aliens build up resistance to your laser fire. On the first wave, one shot blows an enemy away, but on the next attack they take an extra shot before dying; finally they can be killed only if hit four times. But the difficulty is taken into account and these harder-to-kill invaders have appropriately higher point values.

When the player reaches 100,000 points, the difficulty of the game increases a great deal. It is because of this that a hard-to-reach score is anything over 130,000 points. Cosmic Fighter is one of my favorites.

Meteor Mission

As of this writing Meteor Mission II is Big Five's newest entry. There are two different arcade games I have seen which are almost identical to MM II, Lunar Rescue and Escape From Mars.

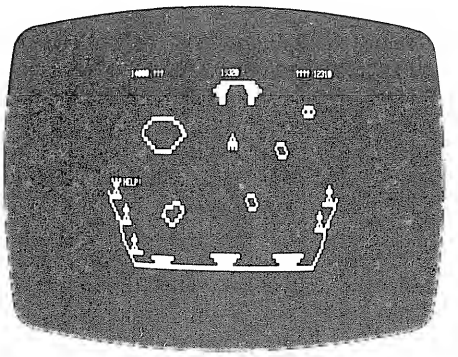
When the game begins, you are sitting in your rescue pod which is in the hull of a larger mothership. The captain informs you that your mission is to retrieve six stranded astronauts from the ravine below. When the path to the surface looks clear, you release your ship and descend toward the landing pads. Your descent is hindered by a field of slowly moving meteors which, if even gently nudged, will prove lethal.

In the beginning of the game there are three complete and separate landing pads, but whenever you take-off from one, the blast of the lift-off engines deteriorates the surface, thus making the pad smaller, but giving it a higher point rating if you land successfully on it later.

If you manage to land safely, an astronaut runs over to your ship and climbs aboard. Your next task is to get your small craft back into the hull of your mothership, intact.

You must shoot a path through the meteors to the safety above. The speed of ascent can be increased by keeping the firing key depressed. Should you, by skill and/or luck, rescue all six astronauts, you are awarded an extra 1000 bonus points.

The next mission is identical to the previous one except that it is somewhat harder with more numerous and faster moving meteors. And if that's not enough to deter you, another danger is added;



occasionally a concentrated asteroid shower will cascade through the sky and devastate your ship if it is in the path of the flying debris.

All of these complications make Meteor Mission II a challenging and captivating game that won't collect dust in your software library. It will probably become one of your favorites.

If you don't own a Big Five program, I urge you to buy at least one, but I must also warn you that you will probably not be able to rest until you know that you've conquered the forces of evil and made your galaxy safe.

As you can see, Big Five has brought much of the excitement, tension and enjoyment of the arcade games to your TRS-80. Now Big Five offers joysticks that attach directly to the TRS-80 Model

I to enhance the arcade realism of their games. The joysticks sell for $39.95.

These five programs come on either 16K cassette ($15.95) or 32K disk ($19.95). There is a Model I and a Model III side on each tape/disk. The disk version has a special feature which, after every game, saves the high scores to the disk automatically. These scores are permanent and reappear each time you reload the game.

Big Five offers a 10% discount for two items ordered at one time and a 15% discount for three or more. Add $1.50 for shipping and handling.

Big Five Software, P.O. Box 9078-185, Van Nuys, CA 91409. (213) 782-6861. □

Robot Attack and Forbidden Planet

Talking Games for the TRS-80

Owen Linzmayer

Until a few months ago, if you wanted to give your TRS-80 the gift of speech, you had to purchase an expensive voice synthesizer, but not any more. Two new games on the market produce voices through the cassette port, these are *Robot Attack*, an arcade game from Big Five, and *Forbidden Planet* adventure from Fantastic Software.

Robot Attack

The object of *Robot Attack* is simple: you are stranded on an alien space station and you must destroy as many robots as you can before they kill you. Escape is impossible and death inevitable.

At the beginning of the game you are in charge of four men; an extra man is awarded every 5,000 points. You maneuver the men, one at a time, through an infinite series of partitioned rooms. As you travel from room to room, enemy robots try to do you in, either by shooting or ramming your man. Your only defenses against these mechanical monsters are your laser gun and superior intelligence.

By pressing the four arrow keys in various combinations, you can make your man run in eight directions. Up and down movements are the slowest, left and right are somewhat faster and the diagonals are the quickest. Using diagonal movement is a handy way to dodge enemy fire and outrun the Fiendish Flagship which appears if you tarry in one room too long.

To shoot your laser pistol, just aim in any direction and press the space bar. Only three of your shots are allowed on the screen at a time. Any robot hit by a shot is instantly blasted into space trash, but alas, you cannot harm the Fiendish Flagship.

Your intellect is almost as powerful a weapon as your laser gun. The robots, after all, can only do what they have been programmed to do. After playing a few games, you learn some of their habits, and begin to be able to predict what they will do when you make a specific move. Using what you know, you can cause the robots to run into themselves, and the electrified walls, and even shoot other robots. It is possible to clear a roomful of enemies without firing a single shot!

If you manage to destroy everything in a room, you are awarded 10 bonus points for every robot that was in the room when you entered. Just because you have cleared a room, don't think for a moment it will stay that way. The master security computer has been programmed to re-stock each room after you leave it.

Have I forgotten anything? Let's see, oh yes, the voices. The voices are terrific! Robot Attack has crisp, understandable voices which add a new dimension to game playing.

The cassette version of *Robot Attack* has a vocabulary of about a dozen words, the disk version has three times as many. Incidentally, if you've ever wondered what Bill Hogue sounds like, it is his voice that talks to you throughout the game.

In conclusion, *Robot Attack* is an excellent TRS-80 adaptation of the popular arcade game. It is somewhat slower than other Big Five games, but stands strong among them.

SOFTWARE PROFILE

Name: Robot Attack

Type: Arcade game

System: TRS-80 Model I or III, 16K cassette, 32K disk

Format: Cassette or tape

Language: Machine

Summary: Excellent talking arcade game

Price: Cassette $15.95, disk $19.95

Manufacturer:
Big Five Software
P.O. Box 9078-185
Van Nuys, CA 91409

Forbidden Planet

You wake up in a plastic container

Owen Linzmayer is a frequent contributor to Creative Computing magazine.

aboard a malfunctioning star-cruiser. The ship's alert wails in the background, frantically you race around trying to find the right objects to repair the ship. Once you have fixed the cruiser you must crash-land it on a strange "forbidden planet" filled with angry ogres, caves, modern cities, radioactive lakes and much more.

This is the scenario for the first talking TRS-80 adventure, *Forbidden Planet.*

Forbidden Planet is a machine language program written by William Demas, the co-author of Scott Adams' Adventure #12, and is sold by Fantastic Software. Adams' style is evident throughout the program, from the split screen to the immediate response time. If I didn't know better I would have sworn that *Forbidden Planet* was Adams Adventure #13.

The program is a text-oriented adventure; you input commands in a verb/noun format, try to unlock mysteries, find treasures, fight battles and survive to enjoy it all. *Forbidden Planet* is exciting, sometimes frustrating, fast moving, challenging and just plain fun. It boasts a vocabulary of over 140 words plus 50 or so spoken words.

Don't be misled into thinking you are "talked along" the adventure. Only key phrases or clues are spoken, therefore if the computer talks, you had better listen carefully to what it says.

If you want, you can toggle the voice on or off in the middle of a game. The voice in *Forbidden Planet* isn't a drab monotone robot voice, but rather, a lively emotional human voice. The voice quality is good but slightly harsher than that of *Robot Attack.* Even so, every word is audible and comprehensible.

The space setting used in this program isn't novel, but Demas has added many original twists to the adventure. Even experienced adventurers will find *Forbidden Planet* very difficult to solve.

There is a variety of ways to die in the first 20 or so locations, and, needless to say, survival is tough. Anyone who enjoys a challenging adventure will find this program a delightful array of puzzles and obstacles waiting to be solved and overcome.

Forbidden Planet is a handsome addition to any collection of computer adventures and is worth the $39.95 price. □

SOFTWARE PROFILE

Name: Forbidden Planet

Type: Text Adventure

System: TRS-80 Model I or III, 48K

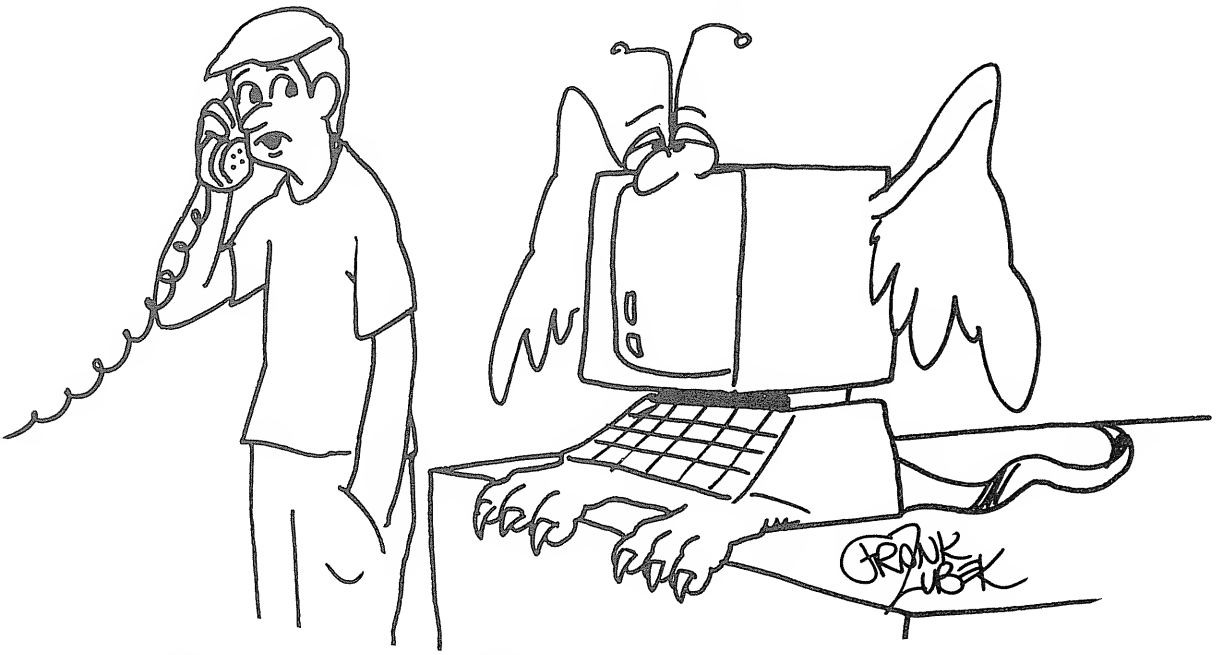
Format: Disk only

Language: Machine

Summary: Talking adventure

Price: $39.95

Manufacturer:
Fantastic Software
P.O. Box 27734
Las Vegas, Nevada 89127



"Hey Bob — I just bought that game called "Metamorphosis" — Guess what??"

Super Vaders and Space Intruders

Invaders for the TRS-80

Owen Linzmayer

The software market has been saturated with versions of Space Invaders for the TRS-80; few great, some OK and many horrendous. Rather than criticizing the deplorable, this review is intended to praise two exemplary programs modeled after the famous arcade game. The programs are Super Vaders from Soft Sector Marketing and Space Intruders from Adventure International.

Space Invaders is the grand-daddy of arcade games. In it, rows of aliens march left and right, criss-crossing the screen, launching lethal missiles at the lone defender below. When the army of invaders reaches either side of the screen, it drops down one step closer to the player's laser base. If the invaders manage to get down to the level of the cannon, the game is over and the Taito Corp. is twenty-five cents richer.

As the number of intruders decreases, the speed of the remaining aliens increases. When the first screenful of invaders has been cleared, a second, faster group appears; only this time they start off a notch lower than the preceding wave.

Even though there are four shelters to protect him, the player must constantly be on the alert as he glides across the bottom of the screen picking off any unfortunate invader that gets caught in his line of fire. Occasionally, a UFO will transverse the uppermost portion of the sky, daring the player to blow it away.

Although these UFOs are worth big points, their objective is to draw your attention away, in the hopes that you won't notice an oncoming missile. The arcade version of Space Invaders is endless, if you can manage to keep one base intact you may play forever, racking up scores in excess of hundreds of thousands of points.

Super Vaders

Super Vaders is a machine language program written by Larry Ashmun of Soft Sector Marketing. It is a modified and greatly enhanced version of Invaders Plus. There are nine levels of play, not including the Blitz mode. Blitz is not for the timid; bombs are hurled towards you at dizzying speeds as the invaders whiz through the heavens. I'm told by the author that the number of boards/screens in the Blitz mode is infinite, but I have never gotten past the second set of insuperable invaders.

If you are playing any level other than Blitz, you must destroy only four waves of invaders before the game is over. This is a slight drawback, but with nine levels of play, the game is challenging to novice and expert alike. At the beginning of each new onslaught, the number of laser bases is always four (these will disappear quicker than you think).

Every time one of your bases is destroyed, the number of ships left is flashed momentarily where your last ship was hit. This is a novel feature that more programs should use. In the old version of Invaders Plus, you could only do one thing at a time, move or shoot. In Super Vaders you can do both simultaneously, with rapid-fire shooting no less!

SOFTWARE PROFILE

Name: Super Vaders
Type: Arcade
System: TRS-80 Model I/III, 16K
Format: Cassette or disk
Language: Machine
Summary: Top of the line Space Invader game for the TRS-80
Price: $15.95 (cassette); $19.95 (disk)
Manufacturer:
Soft Sector Marketing
6250 Middlebelt
Garden City, MI 48135

The sound-effects are exceptional. Something is always coming out the cassette port, from the zapping of an invader to the ever-increasing background tempo which intensifies the game.

There are only two small complaints I have about this game. The first dealing with the scoring: your score is only shown after the game is completed. This is a minor problem but I can see why it was done this way. To have on-screen scoring, the top row would have to be set aside for the score section instead of being reserved for the UFO ship.

My second complaint is that the two-player option is not really that at all. The second player doesn't get to man the controls until after his opponent has been annihilated four times and is finished with his game.

Space Intruders

Space Intruders, also a machine language program, is written by Doug Kennedy. It differs from Super Vaders in that it is modeled after Space Invaders Part II (commonly know as Deluxe Space Invaders). Intruders replicates every aspect of the arcade game, and very well, I might add.

One of the first differences between the original and deluxe arcade games you will notice is that there are "splitting invaders." That is, if you hit one, it duplicates itself and a clone appears beside it. These two do not split again if hit.

There are three different types of UFOs in Space Intruders; the regular ship, a flashing ship and a reinforcement ship. The flashing UFO blinks on and off as it flies across the screen. To destroy it, your laser blasts must hit it when it is "on." The reinforcement ship periodically comes by to drop extra aliens into the empty slots in the uppermost row. This can be nerve-wracking when you thought you were almost finished with a wave and suddenly more intruders are strewn in your path to victory.

Space Intruders is so much like Deluxe Space Invaders that it even has two features many people don't even know exist in the arcade game. The first is "counting your

Owen Linzmayer is a frequent contributor to Creative Computing magazine.

shots." Contrary to popular belief, the UFO point values are not random, but rather follow a pattern depending on the number of shots you have fired. By counting shots and hitting the UFO at the right time, you can consistently get the maximum point rating.

The other feature, one that only a few people know about, is referred to as the "rainbow effect." If the last alien on the screen is one from the bottom row and you destroy it, you are awarded bonus points and treated to an interesting graphics display (the rainbow).

There is a two-player option in this program in which players alternate turns after being destroyed, but the time allotted for changing positions is not sufficient.

One extra ship is awarded at 2000 points; that's the only freebie you'll get, so use it well. Unlike Super Vaders, this program does have on-screen scoring and also lets the high-scorer input his name (eight letters maximum). On the lower left, the number of ships remaining is shown and at the right, the number corresponding to the wave you are presently battling.

Both Super Vaders and Space Intruders use excellent sound routines and lightning-fast, smooth graphics. They are the top-of-the-line Space Invader games for the TRS-80. I don't recommend one over the other because they are modeled after different games.

If you like the original Space Invaders then get Super Vaders, if you prefer the Deluxe arcade game, then by all means, get the Space Intruders program. Better yet, buy them both. Then you'll have all the invader games you'll ever need. Both games are virtual black-holes, capable of sucking up hours and hours of play time while improving your game. □

SOFTWARE PROFILE

Name: Space Intruders

Type: Arcade

System: TRS Model I/III, 16K

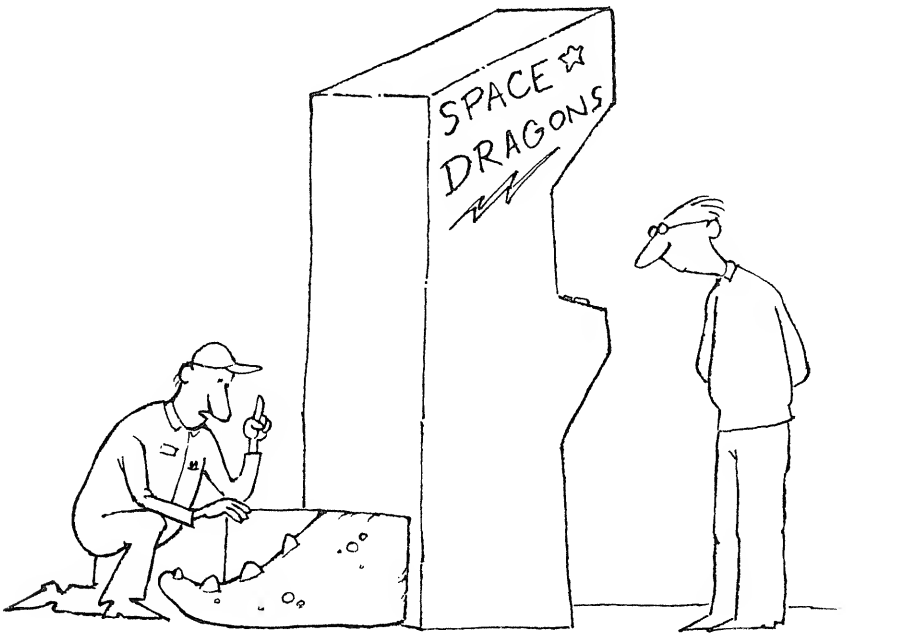
Format: Cassette or disk

Language: Machine

Summary: Excellent TRS-80 rendition of Deluxe Space Invaders

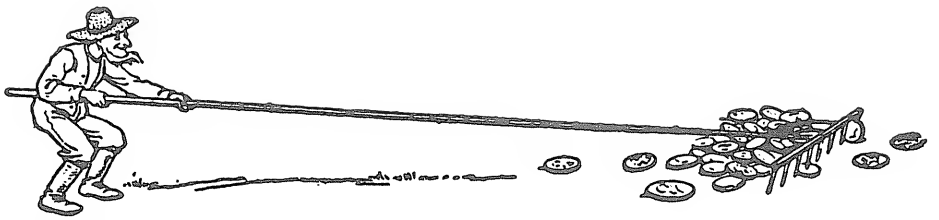
Price: Space Intruders $19.95 tape Model I and III, $20.95 disk Model I

Manufacturer:
Adventure International
Box 3435
Longwood, FL 32750



"AHA!"

Chapter III
Business



Getting Started With a TRS-80

Small Business Computing

Chet Behrman

So you're thinking of going into business as a consultant and programmer for small business applications?

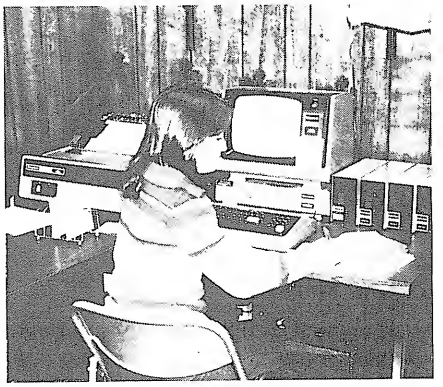
Twelve pages have been ripped off the wall calendar, signalling the completion of one year of making a living with the TRS-80. Everyone has his own definition of "making a living," so the statement is safe. Beginning in June, 1978, I officially launched a company offering microcomputer services in the Evansville, Indiana, area. The initial objectives have changed slightly, profits and peripherals have both grown, and the number of error messages continues to decrease.

What started as a vague offer of "computer services" has now become a strong emphasis on custom business programming for the TRS-80. I've dabbled in games for national syndication, but the real money-maker has been programming and corporate hand-holding among small and not-so-small businesses in the area.

My situation isn't typical, but there must be some lessons to learn from the past twelve months. After spending more than thirty years in broadcasting, the last 22 as program manager of the NBC-TV station in Evansville, I made the move to a second career. In your mid-fifties, you don't fill out work application forms no matter what the EEO people say. So I turned to an old love - data processing - and a business of my own.

Chet Behrman, The Program Manager, P.O. Box 45, Inglefield, IN 47618.

Two off-and-on years of computing science at the local university provided the official background, but I had been poking into computers and punchcards for more than two decades, always with broadcast applications in mind. In 1977, I decided to buy a microcomputer, and Radio Shack's announcement of the TRS-80 that fall prompted an immediate order.



Typical TRS-80 multi-drive business system.

Starting with Level I and a handful of cassettes I began feeding in some modified programs I had run on the IBM 370 and was pleased to see that the TRS-80 came up with the same answers. The decimal points weren't aligned, but PRINT USING was still a few months off.

When I saw Level II performance, the whole idea of making something practical out of the TRS-80 began to take shape. With a disk drive and a line printer on order, I sat down with 16K Level II and looked around the community. Without a printer, turning out reports is rough. (Although my first self-appointed assignment was a television syndication report laboriously copied off the CRT and sold to a number of TV stations around the country.) While waiting for a printer to materialize I struck up chatting acquaintances with all the Radio Shack managers in the area, including parts of Kentucky and Illinois. They mentioned potential customers who had come in to see the TRS-80, but they were properly reluctant to send me knocking on business doors in violation of Tandy policy.

The New Business Computerists

So I literally followed the advice of, "Don't call us . . . we'll call you." The calls began coming in - from businesses that had already bought a TRS-80 and didn't know how to proceed from there - and from business people who were thinking of buying but needed more information about software possibilities. Since then, the prospect list has become a long one.

My first customer was a plumbers' local . . . asking for a mailing list program. Actually, what was termed a mailing list turned out to be a complete membership file, displaying and printing name, address, age, telephone number, card number, work classification, Social Security number, initiation date, apprentice/journeyman labels and active/retired designations. The program was set up to search the files by name, city, state, county, card number, work classification and active status. In these areas, it could print mailing labels, display on the screen, print lists and summarize classifications. All of this was done with 16K, one drive and a line printer; and they've just recently expanded to another drive.

```
                    U. S.  TREASURY
                PAYROLL FOR WEEK ENDING JULY 4 1777
   REG          OVT      FED       FICA      STATE      OTHER     NET

312-22-1732 GEORGE WASHINGTON  42 HRS 35 MINS
   190.00     18.41     19.88     12.78      3.01      2.25    170.49

305-11-1776 THOS JEFFERSON  40 HRS 0 MINS
   160.00      0.00      8.45      9.81      1.66      5.00    135.08

371-13-4848 BETSY ROSS  41 HRS 15 MINS
   150.00      7.03     23.11      9.63      2.76      1.15    120.38

302-04-1313 AARON BURR  42 HRS 10 MINS
   140.00     11.38     17.88      9.28      2.26      0.00    121.96

 TOTALS
   640.00     36.82     69.32     41.50      9.69      8.40    547.91


GROSS TOTALS:
GEORGE WASHINGTON        208.41
THOS JEFFERSON           160.00
BETSY ROSS               157.03
AARON BURR               151.38
   TOTAL PAID            676.82

12.26% TAX THIS WEEK IS    152.30

00/00/00
```

Figure 1. Sample custom payroll program.

Every TRS-80 user knows what waiting for equipment delivery is like; so the initial program for the union local was a Level I creation, using cassettes and CRT. Later, this was modified for Level II with printer, and then modified again for disk operation.

By the time my disk and printer arrived in the fall of 1978, local business was brisk. All of us learned later about the tremendous sales success of the TRS-80 in 1978; I was feeling that local impact, surprised at how many of the systems were being purchased in the area for business purposes.

The absence of business systems ready for demonstration in Radio Shack stores brought my clients and potential clients to my home for a hands-on demonstration. An all-day seminar set up by Radio Shack in downtown Evansville also spurred a great deal of business interest. People like to see the equipment, press the keys, and see CRT or printer response before they sign on the line.

32K and two drives seem to be a good minimum layout for business with the Model I. The DOS takes a lot out of 16K, and I've since expanded my own layout to 32K. Even with a one-drive application, two drives ease the BACKUP chore — and BACKUPs do become a way of life. If inventory is involved, its size, of course, dictates the required number of drives.

Subsequent jobs included my own payroll program and a great variety of inventory approaches, including such diverse businesses as upholstery, light manufacturing, automotive parts, telephone answering, country club, beverage distributor and tax return preparer. One customer, a vending machine company, was lost due to a long wait for equipment. This was unfortunate because the application, processing a few hundred vending machines, was a natural for the TRS-TRS-80.

Custom Programming and Systems Analysis

The most difficult part of the past year hasn't been the programming but rather the systems analysis - understanding how the client operates. Actually, custom programming is the simplest approach - tailoring systems to the customer's particular needs rather than creating programs that must satisfy dozens of potentially conflicting situations. But learning what those needs are takes a great deal of consultation. My years of on-the-air interviewing have helped in keeping a discussion on the track and probing for hidden pieces of information. It isn't easy for people to explain to you what their business is all about. I listen, make notes, and then debrief myself into a cassette recorder (not the one that goes with the TRS-80!).

After initial discussions, an outline is prepared showing just what the system will do. (By that time, we're using the term "system" because a half a dozen or more individual programs are involved.) This is the critical point - getting the customer to agree that **this** is what he wants in the way of reports, and **this** is the type of entry required to feed the data into the files. The written proposal quickly pinpoints misunderstandings on either side, and it's revised, several times, if necessary, until an agreement is reached.

In the meantime, you've already turned out a few quickie demo programs with miniature files to give the customer an idea of what he can expect. To be honest, you point out that the larger, actual files won't process as fast as these smaller demo jobs. But, hopefully, he's still impressed. (The male pronoun is used for convenience. About half of my contacts have been female.)

All of the consultation, demonstration and initial flowcharting is done while the customer is waiting for

MAXIMUM TRS-80 FILE SIZES

Sub Records	Bytes per Record	Records on DOS 2.1	Records on DOS 2.2*	Records on Full Disk
1	255	230	205	325
2	127	460	410	650
3	85	690	615	975
4	63	920	820	1300
5	51	1150	1025	1625
6	42	1380	1230	1950
7	36	1610	1435	2275
8	31	1840	1640	2600
9	28	2070	1845	2925
10	25	2300	2050	3250
11	23	2530	2255	3575
12	21	2760	2460	3900
13	19	2990	2665	4225
14	18	3220	2870	4550
15	17	3450	3075	4875

*DOS 2.2 figures refer to a disk with all accessible programs removed.

Table 1

equipment delivery. As soon as it's delivered, initialization programs start him into the process of file building while you're working on the main program structure.

Then there's the question of space. One of the first questions asked by a potential customer is, "How many inventory pieces can you store on one disk?"

"It all depends" isn't a very satisfactory reply. I put together the chart in Table 1 which shows how many random records can be accommodated with a given record length. A record length of 44 bytes, for example, shows you can put a maximum of 1150 records on the TRS-80 DOS disk. Table 1 shows that an "open" disk with the same record length can accommodate 1625 records. This also means that if you can possibly shorten the record to 42 bytes, the DOS disk can be expanded to 1380 records. Without a long explanation of what a byte is, this is good, factual information that a would-be user can appreciate.

User Programming

It's probably coincidence that the word "custom" is also found in the word "customer." Custom programming certainly keeps the customer in mind. In dealing with first-time users, especially, you assume nothing. You create the program to help the person at the terminal in what may be, to him or her, a rather frightening experience. If your opening menu has a choice of five options, be prepared for the user to choose a sixth option that doesn't exist. If choosing 6 automatically spills you to 1, you've done the user a disservice. Explanations on the screen have to be complete but concise, saving the detailed remarks for the written manual. INKEY can be a very fancy way to input data, but it can be confusing to a new user to whom you've emphasized again and again, "Be sure to press Enter after you've made your entry!"

I've even abandoned the time-honored word "menu" in favor of "select" and "master select" which seem to be more meaningful and less whimsical to a first-timer.

I'm careful to avoid the "domino setup": Press this button and you start a routine that you wished you hadn't! For example, after the payroll is printed, the screen asks, "Are you ready to update the file?" In most cases, the answer is yes. But this gives the user a chance to backtrack on an unfortunate mistake and redo the entries without inevitably spilling all that wrong information into the file.

The true custom program, of course, needs less explanation than something off the shelf. The customer has already asked for many of the features that the program contains. Yet, documentation is still necessary. Every program has a date at the top. When a change is made, the date changes. Modifications are inevitable when you have a continuing working relationship. "Could you change it so that . . . ?" is a common question. So a custom program is much like a painting - you never know when you're finished. A dab here and a dab there, and the system becomes more and more polished.

Hardware Considerations

Equipment problems need some mention. When a disk drive or an interface act up, they're easy to remove and send off to a service center in another city. But a business user can't go down for two weeks or even two days without serious repercussions. If one drive in a four-drive system goes out, the program may be temporarily modified to limp along on three drives until the fourth is returned. But the real answer to this problem is standby equipment. This is not included in Radio Shack's policy: if there's a drive on the shelf, someone wants to buy it. However, the very fact that the TRS-80 (Model I) is modular makes a standby arrangement very attractive. The burden of such an arrangement probably falls on the company providing support; but it's to everyone's advantage - Radio Shack, the business user and the software supplier - that the system continues to operate.

In the 18 months I've worked with my own TRS-80, only one problem developed - a chip flaw in the CPU that knocked out the upper part of RAM. This was immediately correctable by designating full memory size, and the chip was eventually replaced.

Another system developed major problems finally traceable to a bent cable connector plus dive misalignment, but the majority of the systems have worked well. The chief annoying factor for the user is the disk read/write error message that breaks the program, yet I've seen systems turn out hour after hour of printouts without a single mishap.

One of my customers, an automotive parts company, eventually switched from a four-drive TRS-80 system to the Tandy 10, and the improved reliability of performance was immediately apparent. The Tandy 10 is a modified System 70 made by Applied Digital Data with the Tandy label attached. With its 8-inch disks and special-function keys it bears a striking operative resemblance to the new TRS-80 Model II. Or maybe it's the other way around. If the Model II can match the Tandy 10's reliability quotient (it already has a potential of twice the capacity of the Tandy 10), Radio Shack has another winner. The Tandy 10 has received little publicity and is reportedly being phased out, overshadowed now by the lower-priced Model II.

After the superb performance of Level II, one is somewhat taken aback by the appearance and reappearance of error messages in disk operation. Hopefully, the latest DOS version will ease some of this anxiety (and the introduction of 8-inch disks in the Model II may put it completely at rest). After losing two programs while SAVEing them - a read-write error spilling into the system level - I adopted a new practice. When I plan to spend an hour or two creating at the keyboard, I operate at Level II. When I'm finished, I feed it to a cassette, activate the disk system, CLOAD it into CPU and then SAVE it to the disk.

APEX MANUFACTURING CO INVENTORY AS OF JUNE 29 1979

STOCK	VENDOR	VEND STOCK	DESCRIPTION	ON HAND	AVG COST	TOTAL VALUE
10024	93	800T-N40	RED LENS CAPS	5	48.50	242.50
10025	93	800T-XD1	CONTACTS	0	0.00	0.00
10026	93	800T-XD2	CONTACTS	21	13.95	292.92
10027	93	800T-XD5	CONTACTS	7	36.81	257.64
10028	93	800T-XD6	CONTACTS	15	18.52	277.80
10029	93	800T-N129	MUSHROOM HEAD GUARDS	2	126.72	253.44
10030	93	800T-N229	MOUNT RING KIT	12	20.84	250.08
10031	93	P271	SEL SWITCH KIT	1	242.75	242.75
10032	93	P272	SEL SWITCH KIT	2	124.15	248.30
10033	93	RRD	CONTACTS	12	22.40	268.80

TOTAL VALUE OF INVENTORY IS $2334.23

Figure 2. Sample custom inventory program.

Although most of my business customers have abandoned their cassettes, I make active use of mine in storing programs. Disk backups, yes . . . but the cassette is still a very efficient way of storing programs. I have literally hundreds of programs, including various versions of a single program, stored on cassettes, with more than forty disks dedicated to individual clients and projects.

The Money Angles

How do you make money at all of this? Very deliberately! I give an initial estimate for programming a specific system. By a system I mean a series of related tasks, a working package, in other words. My programming rates have gone up this past year, not due to inflation but rather to a growing recognition of the time it takes to produce a quality system. I'd like to charge more, but charging more than hardware costs doesn't seem practical at this level of the profession.

In addition to programming charges, I request a monthly "service and support fee." This keeps me on call to answer questions, make minor program modifications, do a little hardware troubleshooting, and generally keep the system operating. So far, no one has really taken advantage of me on this. The support fee begins when the equipment arrives, and it serves a double purpose. One, it permits the customer to call me at any time without having to watch the clock and worry about a mounting hourly expense. Second, it prompts customer cooperation in putting the system together. If he drags his feet, calls off appointment after appointment, delays in initial file building, it costs him nothing but money. I already have the incentive to get the system operating as quickly as possible: I can't bill for programming until it's finished.

When I first announced that I was going into business for myself with a microcomputer from Radio Shack, I was tactfully asked, "Have you checked with people like IBM or Burroughs?" Somehow, the name Radio Shack doesn't have the authoritative ring of a Honeywell or a Control Data. Apparently, the Tandy Corporation has been worrying about this all the way to the bank. I have great respect for IBM and all the other DP giants, but the computer on the table next to me is a Radio Shack TRS-80. It's the only computer I can afford, and it's done, and is doing, a great job for me.

Summary

The future of custom programming isn't necessarily bright. Right now, custom software fills an obvious need in the microcomputer field. But DP history shows a trend toward standardization - a package for every need right off the shelf. In fact, many other industries display the samd trend. How many people buy custom suits as compared with those buying clothing off the plain pipe racks?

So I'm working on a series of specialized, off-the-shelf programs. In the meantime, as far as user satisfaction is concerned, custom programming is difficult to beat. The customer remains in control - he remains an individual in a growing world of sameness. He may buy his car off the showroom floor, his noon lunch from a franchised restaurant, his very computer from a busy assembly line; but his custom software is distinctively his own. Its operation echoes strong links with his past operation, something with which he can identify. How long the independent small business person will continue to buy this kind of personal identification is a question worthy of feeding into the nearest TRS-80. The answer, of course, depends upon who is doing the programming. □

"The last electrical storm did quite a bit of damage. We fixed the hardware that saps the life-force from half the planet at the price we quoted you. The section that controls your living dead, however, has to be extra. And . . . if you don't mind my saying so, sir, this is quite a piece of programming!"

Analyzing Statistical Data

Multiple Regression Analysis Simplified

David M. Chereb

While the LEVEL I BASIC from Radio Shack is limited, it does have enough capability to handle some rather advanced programs. The example included here (and listed in Figure 4) is a multiple regression routine. The key to the program is the single allowable vector A(n). The length of this vector is limited only by memory size (memory/4).

Because multi-dimensioned arrays are not supported by LEVEL I, special techniques are needed to simulate the N by K matrix which is the basic starting point for the multiple regression routine. Also since the program uses a modified Gauss-Jordan elimination technique, there are many sections which use nested FOR loops. Doing all this with a single A(n) vector brings one closer to understanding the nature of insanity. The basic technique is to manipulate the vector as if it *were* an N by K matrix, letting special counters do all the hard work of locating the correct numbers (see Figure 1).

Multiple Regression

The multiple regression program finds the statistical relationship between the K independent variables and a single dependent variable. What does that mean? It means that the statistical technique of multiple regression will give us the *best* equation possible. There are certain restrictions which must be met for this to be true. For our needs, we can assume these conditions are met in most practical situations. An example of a multiple regression problem is shown in Figures 2 and 3.

Dr. David M. Chereb, 4005 Locust Ave., Long Beach, CA 90807.

FIGURE 1

MATRIX MANIPULATION WITH THE A(n) VECTOR

Let's suppose we wanted to add two matrices B and C and call the resultant matrix D. The matrices must be of the same size to do this, so let's say they are both 4 by 3 (4 rows by 3 columns).

$$B = \begin{vmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 2 & 3 \\ 1 & 3 & 1 \end{vmatrix} \qquad C = \begin{vmatrix} 1 & 3 & 4 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \\ 1 & 2 & 0 \end{vmatrix}$$

If D = B + C then

$$D = \begin{vmatrix} 2 & 3 & 5 \\ 1 & 2 & 0 \\ 3 & 2 & 4 \\ 2 & 5 & 1 \end{vmatrix}$$

For LEVEL I BASIC this is done by:

Assume A(1) .....A(12) = B matrix
A(13).....A(24) = C matrix
A(25).....A(36) = D matrix

$$\text{Then } B = \begin{vmatrix} A(1) & A(5) & A(9) \\ A(2) & A(6) & A(10) \\ A(3) & A(7) & A(11) \\ A(4) & A(8) & A(12) \end{vmatrix} = \begin{vmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 2 & 3 \\ 1 & 3 & 1 \end{vmatrix}$$

The routine is:

```
FOR J=1 TO 3
    FOR I=1 TO 4
                        K=I+(J-1)*4
                        L=12+I+(J-1)*4
                        M=24+I+(J-1)*4
                        A(M)=A(K)+A(L)

    NEXT I
NEXT J
```

Notice that when J=1 then K=I, L=12+I and M=24+I so that the data is allocated to the correct element of the A(n) vector. While there are other ways to do this problem the one presented here closely simulates a two dimensional array program and can be easily modified to handle all forms of matrix operations with any number of columns and rows.

In the example shown in Figures 2 and 3, United States imports are analyzed. The variable to be explained is U.S. imports in constant 1972 dollars (the dependent variable). The explanatory variables are

1) the change in Gross National Product,
2) the size of the labor force, and
3) a variable representing time.

FIGURE 2

MULTIPLE REGRESSION INPUTS

Time Period: 1965 to 1974 by quarter—40 observations

Dependent Variable: U.S. imports in billions of 1972 dollars

Independent Variables (Chosen by author, based upon experience):

1) Change in Gross National Product in 1972 $ from the previous quarter * 10
2) U.S. labor force in millions
3) Time variable: 1000/time, where time=1 in 1957 first quarter and advances by one in each quarter.

These variables were chosen because they are the important variables which affect imports. The first variable is included because rapid growth in U.S. production usually causes imports to accelerate. The labor force variable accounts for the gross level of imports. If the labor force is only 50 million people instead of 100 million, then imports would be much less. The time variable represents a non-linear response over time for U.S. imports. A *negative* coefficient for this variable would mean that imports increase over time even if the GNP and the labor force did not increase. This reflects the generally increasing interdependence of national economies over time (the import share in GNP has doubled in the last 9 years).

The results reveal that all of the independent variables are important (t values over + 2) and that an increase in the labor force by one million people will cause imports to increase by $1.43286 billion. To get an estimate, values for all the independent variables are entered into the equation. If the values for the independent variables are

1) GNP-1 * 10 = $101 billion,
2) Labor Force = 84.2 million, and
3) 1000/Time = 20 (1969:2)

then next quarter the level of imports is expected to be $62.18 billion (at an annualized rate).

We must remember that this is only an estimate. The higher the t values and the coefficient of determination (the R squared adjusted), the more confidence we usually have in the validity of the equation. In this case the estimates of the next period's imports are accurate to within + $3.8 billion (90% confidence interval).

Since the LEVEL I BASIC has only six significant digits, there are roundoff problems. For most uses the estimated coefficients from the program are accurate to 3 or 4 places. Given our current knowledge about economic relationships, we rarely need more than three places of accuracy.

The Basic program uses 6K of memory with each data point adding 4 more bytes. For the example shown, less than 9K of memory is required. To execute this problem takes about 50 seconds. While this seems long in comparison to a large mainframe computer, it is generally acceptable since this is not a problem that is run many times with the same data. Once the equation has been estimated, the regression routine has done its job. From thereon U.S. imports can be easily estimated on any calculator using the equation in Figure 3. The only time the regression routine need be used again on *this* problem is when new information leads you to believe that forecasting accuracy can be improved by re-estimating the equation using the new information (usually more data or more variables). In a business forecasting environment, even higher accuracy would be the goal.

FIGURE 3

MULTIPLE REGRESSION RESULTS

Equation to be estimated:

Imports = B0 + B1 * GNP - 1 + B2 * Labor Force + B3 * 1000/Time
(LF) (T)

= -32.6641 + .0118321 * GNP - 1 + 1.43286 * LF - 1.34998 * 1000/T

Std. Errs	(33.4041)	(.0038404)	(.314984)	(.333134)
t Values	(-.977845)	(3.0809)	(4.549)	(-4/05235)

R^2 = .974
R^2(adj.) = .972
S.E. = 2.2594

Multiple regression uses the data to solve for the unknowns: B0, B1, B2, B3.

The actual technique is to minimize the sum of squared errors of the actual data points of the dependent variable and the estimated data points from the regression equation (i.e. $y-\hat{y}$). There are many textbooks which explain the details of regression analysis.

The Future

Radio Shack's LEVEL II BASIC allows multi-dimensional arrays, transcendental routines and double precision variables. This will make statistical analysis much easier. It will be interesting to compare the execution speed of a multiple regression routine for LEVEL II versus the LEVEL I program shown here. In any event now that you've seen the power of the A(n) vector, perhaps you'll shun all multi-dimensional arrays in the future and stick with A(n). If you do, don't count me in with you. ■

```
10 REM*************************************************
20 REM***  MULTIPLE REGRESSION ANALYSIS               ***
30 REM***     ( LEAST SQUARES ANALYSIS )              ***
40 REM***   BY   DAVID M. CHEREB  5/5/78              ***
50 REM*************************************************
52  REM
53 CLS : PRINT:PRINT TAB(10);"MULTIPLE REGRESSION ANALYSIS"
55  REM  *** DATA FOR CONSUMPTION FUNCTION  ***
56  REM  *** VAR 1=CONSTANT  2=DIS. INCOME  3=CONSUMPTION(-1) ***
58  DATA "U.S.  CONSUMP."
60  DATA 1,1,1,1,1,1,1,1,1,1
62  DATA 1,1,1,1,1,1,1,1,1,1
64  DATA 2.862,2.877,2.910,  2.911,2.946,2.961,2.933
66  DATA 2.913,2.926,2.999,3.021,  3.059,3.125,3.113,3.132
68  DATA 3.154,3.203,3.210,3.201
70  DATA 2.632,2.637,2.634,2.669,  2.689,2.704,2.734,2.721
72  DATA 2.689,2.709,2.744,2.787,  2.838,2.897,2.908,2.928
74  DATA 2.954,2.995,2.986
76  REM  *** DEP.  VAR  (CONSUMPTION) IS NEXT  ***
78  DATA 2.637,2.634,2.669,  2.689,2.704,2.734,2.721
80  DATA 2.689,2.709,2.744,2.787,  2.838,2.897,2.908,2.928
82  DATA 2.954,2.995,2.986,2.996
90  PRINT : PRINT"THIS PROGRAM ESTIMATES A LEAST SQUARES EQUATION "
92  PRINT"OF THE FORM  Y = XB    WHERE  Y = (N * 1) VECTOR"
```

```
94  PRINT"                                  X = (N * K) MATRIX "
96  PRINT"                                  B = (K * 1) VECTOR"
98 PRINT" Y = DEPENDENT VAR.      X = INDEPENDENT VAR. "
101  REM
102  PRINT" N= NUMBER OF OBSERVATIONS   K= NUMBER OF INDEPENDENT VAR. "
103  PRINT:PRINT"A SAMPLE PROBLEM IS INCLUDED  TO SEE HOW THE PROBLEM WORKS"
104  PRINT"THE EXAMPLE HAS 19 OBS.  & 2 INDEP.  VAR. "
105  INPUT"DATA SOURCE : 1 = KEYBOARD   2 = EXAMPLE PROBLEM ";B
107  IF B=2 THEN N=19 : K=2 :GOTO 110
108  INPUT"INPUT  N  AND  K  ";N,K
109  INPUT"INPUT NAME OF DEPENDENT VAR. (UP TO 16 CHAR. )";A$
110  FOR I=1 TO 2*N*(K+2)+4*K*K:A(I)=0:NEXTI
111  IF B=2 THEN READ A$
112  IF B=2 THEN 120
```

MULTIPLE LEAST SQUARES ANALYSIS PROGRAM

This program computes the least squares coefficients for the following equation:

Given Y = XB

where Y=Nx1 vector of N observations of the dependent variable
X=NxK matrix of N observations of K independent variables
B=Kx1 vector of coefficients

Then the least squares solution is:

$$\hat{B} = (X'X)^{-1}(X'Y)$$

The specifics of the program are:

1) Will take K independent variables (where K is limited by system memory not program)
2) Will take N observations (where N is limited by system memory not program)
3) Allows the user to input the name of the dependent variable
4) A constant is assumed to be included in the equation and is automatically read into the A() vector (when program asks for K do not count the constant as one of the indep. var)
5) The program displays the $(X'X)^{-1}$ matrix. This helps in analyzing multicolinearity.
6) A complete example of a regression analysis is included in the program for examining the outputs. The data is United States quarterly figures from 1956:I to 1960:IV. The data is in dollars($x10^{11}$). The estimated equation is a consumption function which estimates consumer consumption using income(disposable) and consumption from the previous period as independent variables. The resulting estimated equation should be

$$C_t = -.34 + .76Y_t + .30C_{t-1}$$
(.082) (.079) — standard errors)
(9.27) (3.80) — t-value

This equation was reported by Zvi Griliches, et al., in the July 1962 *Econometrica* journal.

The TRS-80 results from the current program are acceptable given the difficulty of the problem (19 obs. & 3 coeff. to estimate and a 3x3 matrix inversion)

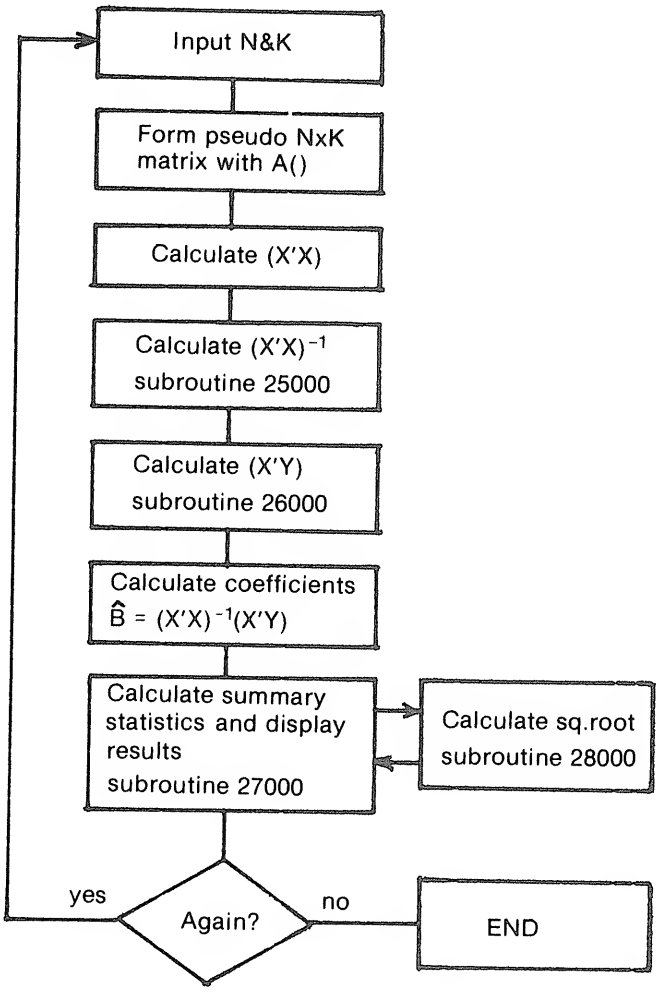
7) The outputs from the route are:
   a) name of dependent variable
   b) least squares coefficients (indep.var.)
   c) standard errors of coefficients (est.of σB)
   d) t-value of coefficients
   e) R-squared (coefficient of determination)
   f) R-squared (adj. for degrees of freedom)
   g) standard error of equation) est.of σ)

```
114  FOR J=2 TO K+1
116  PRINT"VAR # ";J-1 :GOTO 140
120  FOR J=1 TO K+1
130  PRINT"VAR # ";J
140  FOR I=1 TO N
142  IF B=2 THEN READ A(I+(J-1)*N+N):GOTO 180
160    INPUT"INPUT OBS.   ";A(I+(J-1)*N+N)
170   A(I+N)=1
180  NEXTI
200  NEXTJ
220  PRINT" NOW INPUT Y "
230  FOR I=1 TO N
232  IF B=2 THEN READ A(I):GOTO 250
240  INPUT A(I)
250  NEXT I
400  CLS
500  PRINT:PRINT" COMPUTING  X'X MATRIX ELEMENT # "
600  REM  ***
690  G=0
695  K=K+1
700  FOR H=1 TO K
800    FOR J=1 TO K
810      G=G+1
820      PRINT@ 95,G
900       FOR I=1 TO N
1000         L=H*N+I
1100         P=J*N+I
1200          Q=K*N+G+N
1300         A(Q)=A(L)*A(P)+A(Q)
1400       NEXTI
1500   NEXTJ
1600  NEXTH
1700  REM
1800  CLS
```

```
3020   CLS
3030  PRINT@ 0," INVERTING MATRIX          J ="
3100  GOSUB 25000
3200  GOSUB 26000
3210  REM
3220  REM  ****************************************
3230  REM  *  B-HAT FROM  (X'X)INV  & (X'Y)       *
3240  REM  ****************************************
3245  REM
3247  CLS
3250  PRINT:PRINT"THE DEPENDENT VAR. = ";A$
3260  G=0
3270  FOR J=1 TO R
3300    G=G+1
3310    FOR I=1 TO R
3320      L=N*(R+1)+R*R+I+(J-1)*R
3330      P=N*(R+1)+R*R*2+I
3340      Q=N*(R+1)+R*R*2+1+(R-1)+G
3400      A(Q)=A(L)*A(P)+A(Q)
3410    NEXT I
3600  NEXT J
4000  REM
4010  REM  **** GOTO SUB  FOR SUMMARY STATISTICS ****
4020  GOSUB 27000
4030  REM
4940  IF B<>2 THEN 9990
9990  PRINT : INPUT"AGAIN (TYPE 1 ELSE 0)";B
9992  IF B=1 THEN 101
9999  END
25000  REM  ****************************************
25002  REM  *  INVERSE BY GAUSS-JORDAN METHOD      *
25004  REM  ****************************************
25006   REM
```

```
25010   REM  **  INPUT IS A  (K,K) MATRIX   OF  X'X  **
25020   REM  **  STARTING POINT OF MATRIX IS N*(K+1)  **
25110   R=K
25120  FOR J=1+N*(R+1) TO R+N*(R+1)
25150     M=R*R+J+(J-(N*(R+1))-1)*R
25160      A(M)=1
25170  NEXT J
25175  REM
25180  REM  ***  INVERT MATRIX  ***
25190  REM
25210  REM
25220  REM  ****  BIG LOOP STARTS ****
25230  REM
25235  G=0
25240  FOR J=1+N*(R+1) TO R+N*(R+1)
25242     G=G+1 : PRINT@ 29,G
25250    FOR I=J TO R+N*(R+1)
25260      H=I+(J-(N*(R+1))-1)*R
25270      IF A(H) <> 0 THEN 25310
25280    NEXT I
25290    PRINT:PRINT"SINGULAR MATRIX  -  CANNOT BE INVERTED
25300    GOTO 9990
25310    FOR K=1+N*(R+1) TO R+N*(R+1)
25320      M=K+(J-(N*(R+1))-1)*R
25330      S=A(M)
25340      V=K+(I-(N*(R+1))-1)*R
25350      A(M)=A(V)
25360      A(V)=S
25370      S=A(M+R*R)
25380      A(M+R*P)=A(V+R*R)
25390      A(V+R*P)=S
25400    NEXT K
25410    P=J+(J-(N*(R+1))-1)*R
25420    T=1/A(P)
25430    FOR K=1+N*(R+1) TO R+N*(R+1)
25440      M=K+(J-(N*(R+1))-1)*R
25450      A(M)=T*A(M)
25455      A(M+R*R)=T*A(M+R*R)
25460    NEXT K
25470    FOR L=1+N*(R+1) TO R+N*(R+1)
25480      IF L=J THEN 25570
25490      M=J+(L-(N*(R+1))-1)*R
25500      T=-A(M)
25510      FOR K=1+N*(R+1) TO R+N*(R+1)
25520        V=K+(L-(N*(R+1))-1)*R
25530        Q=K+(J-(N*(R+1))-1)*R
25540        A(V)=A(V)+T*A(Q)
25550        A(V+R*R)=A(V+R*R)+T*A(Q+R*R)
25560      NEXT K
25570    NEXT L
25580  NEXT J
25590  REM
25600  REM ***  PRINT RESULTANT MATRIX  ***
25610  REM
25615  PRINT:PRINT"  (X'X) INVERSE IS . "
25620  FOR I=1+N*(R+1) TO R+N*(R+1)
25630     FOR J=1+N*(R+1) TO R+N*(R+1)
25640    L=J+(I-(N*(R+1))-1)*R+R*R
25650       PRINT A(L);"  ";
25660    NEXT J
25670    PRINT
25680  NEXT I
25700 INPUT"PRESS ENTER TO CONTINUE";Q
25990     RETURN
26000  REM
26100  REM  ******************************************
26120  REM  *  SUBROUTINE FOR COMPUTING X'Y          *
26130  REM  ******************************************
26140   REM
26200   G=0
26210   FOR J=1 TO R
26220     G=G+1
26225      FOR I=1 TO N
26230       L=J+N+I
26240       P=I
26250       Q=(R+1)*(R+N)+(R-1)*R+G
26260       A(Q)=A(L)*A(P)+A(Q)
26270     NEXT I
26280   NEXT J
26990  RETURN
27000  REM
27010  REM  ************************************************
27020  REM  *  SUBROUTINE FOR COMPUTING LS SUMMARY STATS  *
27030   REM  ************************************************
27040   REM
27100   REM  ***  R SQR  ***
27105  K=R : S=0
27110   FOR I=1 TO N
27120     S=S+A(I)
27130   NEXT I
27140   M=S/N
27154   V=0:S=0
27160   FOR I=1 TO N
```

8) The program proceeds as follows:



```
27450  X=S : GOSUB 28100
27460  PRINT"STD ERROR = ";Y
27990   RETURN
28000  REM
28010  REM  ****************************************
28020  REM  * SUB FOR SQUARE ROOT  -  X IN  Y OUT  *
28030  REM  ****************************************
28040  REM
28100  Y=X/2 : Z=0
28110  W=(X/Y-Y)/2
28120  IF (W=0) + (W=Z) THEN RETURN
28130  Y=Y+W : Z=W :GOTO 28110
```

```
27178       Y=0
27180       FOR J=1 TO K
27184         D=N+I+(J-1)*N
27186         C=N*(K+1)+K*K*2+K+J
27200         Y=Y + A(C)*A(D)
27210       NEXT J
27220         E=A(I) - Y
27240         E=E*E
27250         S=S+E
```

```
27260         T=A(I)-M
27270         T=T*T
27280         V=V+T
27290       NEXT I
27300        R=1 - (S/V)
27302        S=S/(N-K)
27340  REM
27342  PRINT
27345  PRINT"   LS COEFF        STD ERROR         T-VALUE"
27347  PRINT"  ------------   -------------     ----------"
27350  FOR I=1 TO K
27360     Q=N*(K+1)+K*K+(I-1)*K + I
27370      L=N*(K+1)+K*K*2+K + I
27400      PRINT"B(";I-1;") = ";A(L);
27402      X=S*A(Q) : GOSUB 28100
27404      PRINTY;
27406      PRINTA(L)/Y
27410  NEXT I
27430  PRINT : PRINT"R-SQR =";INT(R*1000+.5)/1000
27432  R=1-(1-R)*(N-1)/(N-K)
27440  PRINT"R-SQR (ADJ) = ";INT(R*1000+.5)/1000
```

Keeping Inventory Costs Under Control

Determining Economic Order Quantity

William M. Lowerre, Jr.

Microcomputer magazines are full of ads for business inventory systems. Most of these systems are essentially accounting systems. That is, they account for inventory transactions and maintain on-hand and on-order balances. Inventory systems are logical candidates for computerization because of the relatively large volume of data that must be manipulated frequently.

William M. Lowerre, Jr., 7458 Colton Lane, Manassas, VA 22110. Lowerre is a fellow of The American Production and Inventory Control Society.

Some inventory systems include such data as order quantity, maximum quantity, minimum quantity, reorder point, unit price and annual demand forecast. These data are not transaction data but are necessary to the reorder decision process. That is, they help decide how much to order and when to order. Unfortunately, many inventory software packages available for micros today do not help the manager determine what these variables should be.

Fortunately, there are well and widely known models for calculating 1) the forecast of demand, 2) the most economical quantity to order, and 3) the inventory level at which the reorder should be placed. And unit cost can be obtained either from vendors or from the business cost accounting records. Computers, of course, are very adept at manipulating such models and are a necessity where the model must be exercised repeatedly, as for an inventory of a large number of different items reordered frequently.

This article describes a program that demonstrates that the economic order quantity minimizes the costs affected by the order or lot size decision.

it was written for a TRS-80 with Level II Basic and 16K of core without a printer. Most small businessmen should find it interesting, useful and profitable.

EOQ Costs

There are two kinds of costs incurred as a result of ordering inventory: 1) The costs associated with placing and receiving each order, and 2) the costs associated with carrying inventory in stock after it is received. It is not easy, or sometimes even feasible, to extract these costs from most accounting systems. However, they can be estimated reasonably accurately.

Costs associated with placing and receiving an order include: postage, forms and stationery; buyer time, telephone cost, and shipping costs; receiving and receiving inspection; posting and payment of accounts payable. These costs must be estimated on a cost-per-item-order basis. The cost per-item-order of procurement telephones might be estimated, for example, as:

$$\frac{\text{total annual procurement telephone expense}}{\text{Total number of P.O.'s issued X average number of items/order}}$$

Costs associated with carrying inventory include: interest, insurance, space, heat, light, shelving, obsolescence, theft and breakage. These carry costs must be estimated on a rate per year basis, i.e. as a dollar cost per year per dollar of inventory carried. The carry cost rate for warehouse space, heat and light, for example, is calculated as:

$$\frac{\text{annual cost of warehouse space, heat, light}}{\text{average dollar inventory}}$$

Figure 1 illustrates the case where an order arrives in inventory just as the inventory is depleted, and where the inventory is withdrawn regularly in small increments. The average inventory on hand is equal to 1/2 the order quantity, Q. If the dollar cost per unit is CU and the carrying cost rate is CC then the annual carrying cost, AC, is QxCUxCC/2. AC increases linearly with Q, as seen in Figure 2.

If annual requirements (demand) are forecast as RA, and the cost to place an order for one item is CO, then the annual cost to order the item AO, is COxRA/Q, since RA/Q equals the number of orders placed per year. Annual ordering cost total

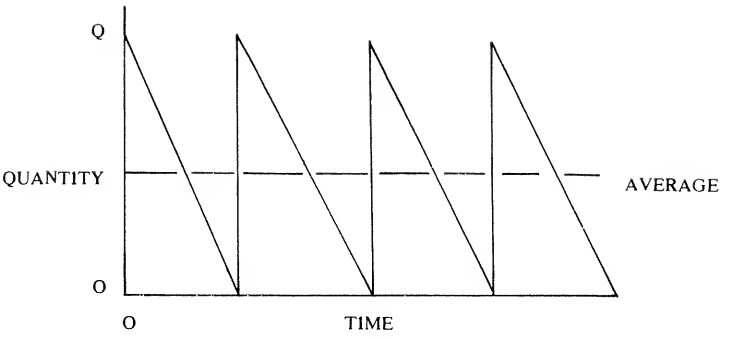


Figure 1. Item Inventory Over Time.

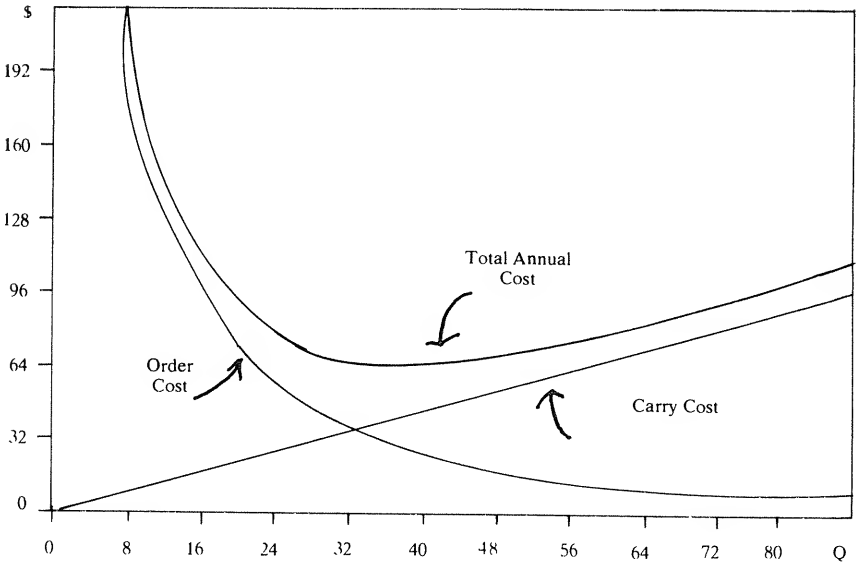


Figure 2. Cost vs. Order Quantity.

Part Number	Units Rqd. Yrly.	Unit Costs	Order Costs
428301	200	5.75	5.58
428302	462	12.76	10.05
428303	311	10	30

Figure 3.

Part Number	Yrly. $ Rqd.	Econ. Ord. Qty.	Month's Supply
428301	1150	31	1.86
428302	15135.1	26	.67
428303	3110	68	2.67

Figure 4.

for this item, AO, decreases curvilinearly, inversely with Q.

The total annual cost, TAC, is the sum of the annual order and annual carry costs, or TAC = AO+AC. TAC first decreases with Q, reaches a minimum—at the economic order quantity (EOQ)—and then increases with Q.

The EOQ can be calculated utilizing calculus. Or it can be demonstrated through interative examples or graphically. The program presented here provides examples and graphics. The next paragraph summarizes the mathematics.

To find the order quantity Q, that gives the least total annual cost. TAC, take the derivative with respect to Q. Set it equal to zero, and solve. The result is:

$$(EO)Q = \frac{2xRAxCO}{CUxCC}$$

This equation gives the order quantity that results in the minimum total annual cost of ordering and carrying an inventory item. It is known as the Economic Order Quantity or EOQ.

The program shown here utilizes the equation above to calculate the EOQ for three sample items (part numbers). Data for part number, annual requirement, order cost and unit cost are stored internally in Data statements (at 1120-1140). Following the title (10-60), a table of variables (70-180) and introductory comments (200-210), the program displays the stored data (230-280), and then requests the operator to input the carrying cost rate (300). The display is shown in Figure 3.

The carrying cost rate (CCR) is not stored since it may vary for different sets of items or part numbers at a given firm, and almost certainly will vary from firm to firm. In addition, management may prefer to treat CCR as a policy variable to shift costs rationally between ordering and carrying as conditions, e.g., in the money or labor markets, dictate from time to time.

The program then calculates the yearly requirements in dollars, the EOQ and the equivalent month's supply to order for each item/part number, and displays them in a table (310-380). This table is shown in Figure 4.

The operator is then queried as to his preference for seeing a tabular display demonstrating the effect on TAC, AC, and AO of varying Q on either side of EOQ, (390-420). If he so prefers, then, for a specified part number the display in Figure 5 is shown on the monitor (450-610). Order costs are slightly different from carry costs since decimal quantities are not permitted for the quantity.

The operator is next queried as to his preference for seeing a graphic display of the data (620). If he so desires, the graphic display in Figure 2 will be constructed for the same part number as shown previously in the tabular format (630-1010).

Part Number 428301

Order Quantity	Yrly. Total All Costs	Order Costs	Carry Costs
24	74.1	46.5	27.6
28	72.05	39.85	32.2
31 EOQ	71.65	36	35.65
34	71.92	32.82	39.1
37	71.71	30.16	42.55

Figure 5.

The range of quantity values for the X scale is first defined as twice the EOQ, and the range of total cost is established as twice the total cost at the EOQ, (630-640). This forces the graphics to be centered on the screen regardless of the values of Q and TC. The X axis is drawn (660) and markers are positioned (670). The scale value of the X marker interval is next established by repeatedly dividing the range values by 2 (680-700) until the range of values will fit the available display columns (690). The first marker value is next set equal to the marker interval (710). The remaining X markers are then calculated and printed (720). The process is then repeated for the X axis (740-800). Finally, the frame for the graphics is completed (820-830).

The three superimposed plots of order costs, carry costs and total annual cost are now executed (840-1010). They are executed sequentially for clarity, not for efficiency. A "FOR-NEXT" loop is established (840) to generate a pointer to direct the program to the appropriate sub-routine. A nested loop then increments the value of X across the X axis (850). X = I- 1.999 rather than I-12 to avoid a subsequent division by zero. The approximation is adequate for the purposes of the display. Three subroutines (860-890; 900-930; 940-980) then calculate the corresponding Y value and convert it to the equivalent Y set point value. If set point values are off screen, they are invalid. They are thus limited to the maximum X and Y values (990-1000) before proceeding to set the X,Y points that plot the graph (1010). When all three lines have been plotted, the precise values of EOQ and total annual cost at EOQ are printed at the top of the screen (1020-1030). Since an endless loop is created to permit uninterrupted viewing of the full screen (1080), a flashing message is printed on the screen directing the operator to "PRESS ANY KEY FOR EGRESS" to proceed (1050-1070).

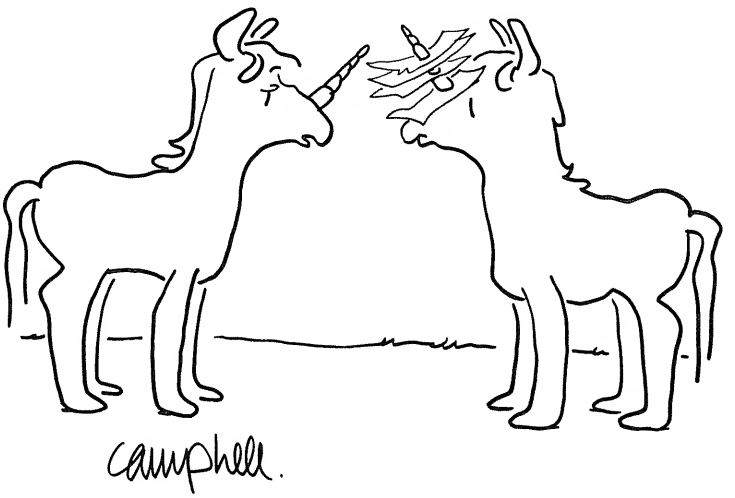
The operator is finally queried as to whether he would like to see the EOQ demonstration on another part number (1090). If "yes" the program repeats from line 220. If "no" the operator is instructed to press "ENTER" to terminate the program (1100). The "DATA" statements complete the program (1120-1140).

The program details are well described with PRINT, INPUT and REM statements throughout that should make it easy to understand and follow.

The tabular and graphic demonstrations of EOQ optimality are essentially tutorial. Thus, for decision making application only, the program might well be truncated after line 390, except for data statements, retaining only the decision-making routines and the necessary data. Some business users may also prefer to access existing files for annual requirements. DATA statements will be suitable for many, however, since the program is likely to be run only infrequently in an off-line mode in many instances. The program with the three data statements leaves 9985 bytes of memory, with the Keyboard Debounce program loaded first in a 16K machine. This should permit entering about 450 six-character part numbers. Of course some firms may wish to treat carrying cost as a constant rather than an operator input variable. And, hard copy print routines will be needed in most cases. Such changes should be simple for most users.

```
10 REM ** * * * * * * * * * * ** * * * * * * * * * * * * * * * * * **
20 REM **           ECONOMIC ORDER QUANTITY (EOQ)                     **
30 REM **                          BY                                 **
40 REM **                   WM. M. LOWERRE JR.                        **
50 REM **               COPYRIGHT  OCTOBER 1979                       **
60 REM ** * * * * * * * * * * ** * * * * * * * * * * * * * * * * * **
70 REM *                   TABLE OF VARIABLES                         *
80 REM *                   ------------------                         *
90 REM * PN = PART NUMBER          RA = ANNUAL UNIT REQUIREMENTS      *
100 REM* CU = UNIT COST             CO = COST TO ORDER                 *
110 REM* CCR = CARRY COST RATE      EQ = ECONOMIC ORDER QUANTITY       *
120 REM* MS = MONTHS SUPPLY         ADD = ANNUAL DOLLAR REQ'MENTS      *
130 REM* Q = ORDER QUANTITY         AO = ANNUAL COST TO ORDER          *
140 REM* AC = ANNUAL COST TO       TC = TOTAL ANNUAL COST AT EOQ      *
150 REM*       CARRY               TAC = TOTAL ANNUAL COST ANY Q      *
160 REM* R- = RANGE OF SCALE       MI = MARKER INTERVAL               *
170 REM* MV = MARKER VALUE                                            *
180 REM* * * * * * * * * * * ** * * * * * * * * * * * * * * * * * * *
190 DEFINT B,J
200 CLS: PRINT "THIS PROGRAM CALCULATES ECONOMIC ORDER QUANTITIES, AND DEMON-
STRATES OPTIMALITY OF THE SOLUTION."
210 PRINT: PRINT "THE FOLLOWING INVENTORY WILL BE USED:": GOTO 230
220 CLS: PRINT "TO SEE OPTIMALITY DEMONSTRATED ON ANOTHER PART NUMBER, SELECT
ANOTHER PART NUMBER FROM THE INVENTORY TABLE BELOW, OR ENTER 0": B=1: REM DIFFER
ENT ROUTE AFTER FIRST PART NUMBER
230 PRINT: PRINT "PART NUMBER","UNITS REQD YRLY","UNIT COST","ORDER COST"
240 PRINT "-----------","---------------","---------","----------"
250 RESTORE
260 READ PN, RA, CU, CO
270 IF PN = 0 GOTO 290  : REM NO MORE PART NUMBERS
280 PRINT PN,RA,CU,CO: GOTO 260
290 PRINT: IF B=1 INPUT "PART NUMBER OR 0"; A: GOTO 440  : REM DIFFERENT ROUTE A
FTER FIRST PART NUMBER
300 IF B=0 PRINT: INPUT "NOW ENTER THE INVENTORY CARRYING COST RATE"; CCR
310 CLS: PRINT "PART NUMBER","YRLY $ REQ","ECON ORD QUANT","MONTHS SUPPLY"
320 PRINT "------------","-----------","---------------","--------------"
330 RESTORE
340 READ PN, RA, CU, CO
350 IF PN = 0 THEN 390  : REM NO MORE ORDER QUANTITY CALCULATIONS
360 LET EQ = INT(SQR(2*RA*CO/(CU*CC))): MS = INT(100*EQ*12/RA)/100: ADD = INT(10
0*RA*CU)/100: REM CALCULATE AND ROUND TO TWO DECIMAL PLACES
370 PRINT PN, ADD, EQ, MS
380 GOTO 340
390 PRINT: PRINT "THE COMPUTER HAS CALCULATED THE ANNUAL REQUIREMENTS IN DOLLARS
  AND THE ECONOMIC ORDER QUANTITY (EOQ), AND HAS CONVERTED THE EOQINTO MONTHS SU
PPLY."
400 IF B=1 GOTO 430
410 PRINT: INPUT "IF YOU WISH TO SEE THE EOQ OPTIMALITY DEMONSTRATED, ENTER A PA
RTNUMBER.  IF NOT ENTER 0"; A
420 GOTO 440
430 PRINT: INPUT"DO YOU WISH TO SEE OPTIMALITY DEMONSTRATED? Y OR N"; A$: IF A$=
"N" LET A=0
440 IF A = 0 GOTO 1110
450 RESTORE: REM SEARCH FOR REQUIRED PARTNUMBER
460 READ PN, RA, CU, CO
470 IF PN = 0 THEN
480 IF PN <> A THEN 460
490 CLS: PRINT "PART NUMBER"; PN
500 PRINT "------------------"
510 PRINT TAB(32) "ANNUAL COSTS"
520 PRINT "ORDER","-------------------------------------------"
530 PRINT "QUANTITY","ALL COSTS","ORDER COSTS","CARRY COSTS"
540 PRINT "--------","----------","-----------","-----------"
550 LET EQ = SQR(2*RA*CO/(CU*CC)): REM CALCULATE RANGE OF ORDER QUANTITIES AROUN
D EOQ
560 FOR I = .8 TO 1.2 STEP .1
570 Q = (100*INT(I*EQ)/100): AO = (INT(100*CO*RA/Q)/100): AC = (INT(100*CU*CC*Q/
2)/100): TAC = AO + AC
580 IF Q = 100*INT(EQ)/100 PRINT Q; "EOQ", TAC, AO, AC: LET TC=TAC: GOTO 600
590 PRINT Q, TAC, AO, AC
600 NEXT I
610 PRINT: PRINT "NOTE: CARRY COSTS INCREASE; ORDER COSTS DECREASE; TOTAL ANNUAL
  COSTS DECREASE THEN INCREASE WITH ORDER QUANTITY."
620 PRINT: INPUT "DO YOU WISH TO SEE OPTIMALITY SHOWN GRAPHICALLY? Y OR N"; A$:
IF A$ = "N" GOTO 220
630 CLS: RX = 2*EQ: REM SO EOQ WILL APPEAR AT MIDPOINT OF X RANGE
640 RY = 2*TC: REM SO TAC MIN WILL OCCUR AT MIDPOINT OF Y RANGE
650 PRINT@965, "0";
660 FOR X=11 TO 127: SET(X,43): NEXT X: REM DRAW X AXIS
670 FOR X=11 TO 117 STEP 10: SET (X,44): NEXT X: REM SET X MARKERS
675 PRINT@ 1020,"Q";
680 XI = 1: REM ESTABLISH X MARKER INTERVALS
690 IF RX < 10.6 GOTO 710
700 RX = RX/2: XI = 2*XI: GOTO 690
```

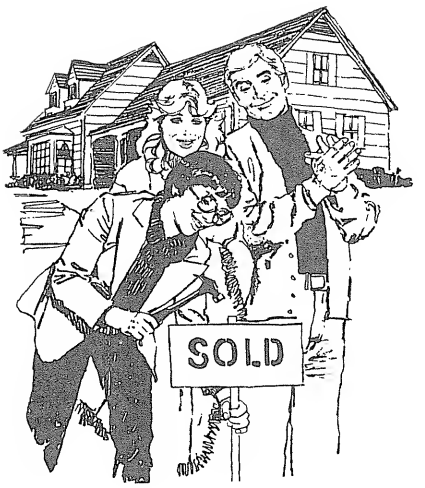
```
710 MV = XI: REM FIRST X MARKER VALUE EQUALS X INTERVAL
720 FOR X = 1 TO 10: PRINT@(963+(X*5)), MV;: MV = MV + XI: NEXT X: REM PRINT X M
ARKER VALUES
740 FOR Y=0 TO 44: SET(11,Y): NEXT Y: REM DRAW Y AXIS
750 FOR Y=7 TO 43 STEP 6: SET(10,Y): NEXT Y: REM SET Y MARKERS
755 PRINT@ 1, "$";
760 YI = 1: REM ESTABLISH Y MARKER INTERVAL
770 IF RY<7.5 GOTO 790
780 RY=RY/2: YI=2*YI: GOTO 770
790 MV=YI
800 FOR Y=1 TO 7: PRINT@(64*2*Y), (7-Y)*YI;: NEXT Y: REM PRINT Y MARKER VALUES
820 FOR X=12 TO 127: SET (X,0): NEXT X: REM COMPLETE FRAME
830 FOR Y=0 TO 43: SET(127,Y): NEXT Y
840 FOR J=1 TO 3: REM PLOT 3 SUPERIMPOSED GRAPHS
850 FOR I=12 TO 127: X=(I-11.999)/10*XI: ON J GOTO 860, 900, 940
860 PRINT@212,"(AO) ANNUAL ORDERING COST";: REM ESTABLISH Y SET VALUES FOR AO
870 AO=CO*RA/X
880 Y=INT(43-6*AO/YI): REM Y SET POINT ANNUAL ORDER COST
890 GOTO 990
900 PRINT@212, "                              ";: PRINT@276,"(AC) ANNUAL CARRYING COS
T";: REM ESTABLISH Y SET VALUES FOR AC
910 AC = CU*CC*X/2
920 Y = INT(43-6*AC/YI): REM Y SET POINT ANNUAL CARRY COST
930 GOTO 990
940 PRINT@276, "                              ";: PRINT@340,"PLOT TOTAL ANNUAL COST (
TAC)";: REM ESTABLISH Y SET VALUES FOR TAC
950 AO = CO*RA/X
960 AC = CU*CC*X/2
970 TAC = AO + AC
980 Y = INT(43-6*TAC/YI)
990 IF Y < 0 THEN Y = 0: REM CONSTRAINS Y VALUES TO GRID
1000 IF Y>44 THEN Y=44
1010 SET(I,Y): NEXT I, J: REM PLOTS GRAPHS
1020 PRINT@82,"EOQ ="; INT(10*EQ)/10;: REM GIVES VALUES NOT DETERMINABLE FROM GR
APH
1030 PRINT@97,"MIN TAC = $";TC;
1040 PRINT@214,"PRESS ANY KEY FOR EGRESS   ";: A$ = INKEY$: IF A$ <> "" GOTO 109
0 : REM HOW TO EXIT GRAPH AND CONTINUE
1050 FOR X=1 TO 700: NEXT X: REM FLASHES LINE @ 341
1060 PRINT@214, "                              ";: A$ = INKEY$: IF A$ <> "" GOTO 1090

1070 FOR X=1 TO 300: NEXT X
1080 GOTO 1040
1090 CLS: INPUT "DO YOU WISH TO SEE OPTIMALITY DEMONSTRATED ON ANOTHER PART
 NUMBER? Y OR N"; A$
1100 IF A$="Y" THEN 220
1110 CLS: PRINT@386, "TYPE IN THE WORD 'RUN' AND PRESS 'ENTER' TO RUN PROGRAM AG
AIN": END
1120 DATA 428301, 200, 5.75, 5.58
1130 DATA 428302, 462, 32.76, 10.05
1135 DATA 428303, 311, 10.00, 30.00
1140 DATA 0, 1, 1, 1
```



"A small computer would eliminate all that clutter."

For buying, selling, and renting property

Real Estate Analysis



Mayer D. Liebman

A comprehensive look at real estate investment techniques. The program was written for a TRS-80 but does not use any special graphics.

The ideal investment is where you commit funds (preferably other people's funds) with the intention of minimizing risk, and safeguarding your capital while earning a generous return, (preferably a tax-free return), and at the same time being able to control, to the best of your ability, the outcome of your investment.

In our opinion, real estate is the only investment which, when bought properly, sold properly and managed correctly, can fit that description. When you consider the many factors working in real estate's favor, you will come to realize, as we did, that real estate is as close to a guaranteed investment as you can get.

To fully understand how real estate can serve as the perfect investment vehicle, take each advantage and consider it:

Safety

When was the last time you heard that someone had sold his home for less than he paid for it? Properly selected real estate is one of the safest investments you can make. Most real estate owners realize that their property values are continuing to increase every year. Real estate will hold or increase its value, and avoid a downslide loss better than almost any other form of investment. The demand for land will always continue to grow.

Also, you will never awaken one morning and find your investment stolen, lost, missing, or the "company has gone bankrupt." You can always see the property and no one can move it away.

Appreciation

Real estate, and most notably the single family house, has been appreciating at an average rate of 10 per cent for the last few years. Improved real estate continues to increase in value because of the demand for housing, increases in construction labor costs and increases in the cost of raw materials.

Yield

Sound real estate investments have always provided a larger overall yield than almost all other investments. First, we have the consistent real estate appreciation. Second are the tax advantages such as depreciation and capital gains treatment.

Many experienced real estate investors like to see at least a 20 per cent return on their investment (including cash flow, market value appreciation and tax savings). And it is obtainable.

Liquidity

It has been written many times that real estate is not liquid. This is true to a certain extent. Of course, real estate is not as liquid as a savings account, but properly priced single family homes can be sold in from 30 to 90 days. Much delay in the selling of a piece of real estate could be avoided if the asking prices were set realistically.

Leverage

Leverage means investing the least amount of capital possible when acquiring a property to earn the maximum percentage return on that investment by obtaining a mortgage (financing) for the highest practical amount.

As has always been characteristic in real estate investment, you can obtain financing for a part of the purchase price. The owner of a piece of real estate can control his investment with as little as 10 to 20 per cent of the purchase price being invested.

No alternate investment allows such a low down payment without the corresponding disadvantages. Real estate leverage allows the investor to receive the entire benefit of the property's appreciation in the market and depreciation of the entire value of the property for tax purposes. Under the new tax law, it is the only investment with those advantages.

Depreciation

Depreciation of real estate is the one factor that produces the high yields which are not found in other investments such as stocks and bonds. It is a tax break which gives you the right to deduct a certain percentage of the building's value each year (but not the land, since in theory land does not depreciate).

This is a non-cash expense. Actually, as the property is increasing in value, you are deducting a portion of its value on your tax form. If you buy the property from someone who has

Mayer D. Liebman, 8105 Buckspark Lane East, Potomac, MD 20854.

already depreciated it, you can start the depreciation cycle over again, as can the person who buys the property from you.

Depreciation, because it is an expense which costs you no money, is what real estate investors look for as a tax shelter. The effect of depreciation is to shield a part of your ordinary income from taxes.

Why single family houses, instead of raw land, shopping centers, or multi-unit dwellings? Because you may already be a homeowner yourself. You have an idea of what to look for in a house. You can easily understand the finances involved as they are quite similar to the financing you did on your own home. They are easily rented, easily maintained and can be managed by yourself. Dollar for dollar they usually out-perform all other modes of real estate for appreciation.

Just think of how much you paid for your home and compare it to the price your neighbor just got for the home he recently sold. The single family home is affordable, depreciable, rentable, easily maintained and managed and, most important of all, appreciates over and above recent inflation.

In buying a single-family house as an investment, you'll have to consider many things, among them:

- Location
- Condition of the house and utilities
- Price
- Financing
- Projected return on investment

In this short article, we have time to consider only what is the most difficult to analyze and yet the most important in weighing the investment — that is, your projected return. This is what is known as "Property Analysis." This is where you record all the pertinent facts about your new found property down on paper so that you can see in black and white what this investment is going to produce. And you can use these same figures to compare it with other potential purchases. This is the "bottom line" in any real estate investment.

Suppose you spot a single-family attached house (known commonly as a "townhouse") with an asking price of $40,000. The down payment is 10 per cent, or $4,000, and you expect you could rent it for $350 per month, plus utilities. You can assume the existing mortgage on the house for $32,000 @ 8% for 30 years (Note: The lower rates on existing mortgages are one of the advantages of buying an older house. New mortgages on recently constructed houses now range over 10%), and the seller will take back a 2nd mortgage of $4,000 at 11% for 8 years. Analyze it as follows while referring to Table 1:

Line 1—Gross Income: This is the monthly rental that you expect to get for the property multiplied by 12 to arrive at the annual amount of rent that you plan to receive.

Enter: $350 x 12 = $4,200

Line 2—Vacancy: This is the percentage that will have to be subtracted from the gross income figure to arrive at the more realistic income figure. The best average is about a 5% vacancy factor. That is, one month out of 20 you will fail to rent the property.

Enter: 5% of $4,200 = $210

Line 3 — Gross Operating Income: This is the amount obtained by subtracting the vacancy factor from the gross income amount: This figure represents the actual amount of money you have to work with.

Enter: $4,200 - $210 = $3,990

Line 4 — Expenses: You will now have to list the expenses that will be incurred in renting and maintaining the property.

```
                PROPERTY ANALYSIS

Address of property.....34 Park Street
Purchase price.....$40000
First mortgage.....$32000 @ 8% for 30 years
Second mortgage.....$4000 @ 11% for 8 years

1.  Gross Income.....$4200
2.  Less: Vacancy.....$210
3.  Gross Operating Income.....$3990
4.  Less expenses:
5.  Taxes.....$700
6.  Insurance.....$100
7.  Utilities.....$20
8.  Advertising.....$60
9.  Management.....$0
10. Civic Association.....$100
11. Maintenance.....$300
12. Cleaning Services.....$100
13. Legal & Accounting.....$150
14. Total Expenses.....$1530
15. Net Operating Income.....$2460
16. Less Loan Payments:
17. 1st Mortgage.....$32000
    2nd Mortgage.....$4000
18. Interest.....$3000
19. Principal.....$571.66
20. Total loan payment.....$3571.66
21. Gross Spendable Income.....-$1111
22. Plus: Principal Payment(s).....$571.66
23. Gross Equity Income.....-$540
24. Less Depreciation.....$1333.33
25. Taxable Income.....$-1873.33

            Tax Analysis

1.  Gross Equity Income(line 23).....-$540
2.  + Tax consequences.....$749.33
3.  After Tax Income.....$209.33
4.  +Growth.....$3200
5.  Net Equity Income.....$3409
6.  Rate of return.....$56.82% per year

Formula:

        Net Equity Income          X 100  =
      ----------------------
Down Payment + Closing Costs
```

Table 1

Line 5 — Taxes: On this line you should enter the real estate taxes that are levied on the property. In our area, this property would carry about $700 in taxes.

Line 6 — Insurance: The insurance that you normally will need on a rental property are fire, vandalism and malicious mischief. You need these to protect the actual improvements on the property. You will also need personal liability insurance to protect yourself against a negligence claim. The cost for the year will be about $100.

Line 7 — Utilities: Normally there will be no entry on this line since the tenant will be paying the costs of utilities. However, you may have to pay a minimal amount during the period of time the property is vacant and waiting for a tenant. Depending on the time of year when the property is vacant will determine the costs. During the summer we can estimate about $20 for utilities when the air conditioning will not have to be used.

Line 8 — Advertising: This would be the costs of placing ads in newspapers to try and secure a tenant. We would estimate a cost of about $60 for advertising in the example.

Line 9 — Management: Since we recommend that for maximum profit that you manage your own property, there normally will be no entry on this line. If you plan on using a management concern, then you can plan on paying a management fee of about 7% per year on the gross rental receipts.

Line 10 — Civic Association: Many developments have a civic association fee. This may cover maintenance of common areas, snow removal, etc. The costs on this house will be about $100 per year.

Line 12 — Cleaning Services: Normally when you are starting out with your first properties you will be doing your own cleaning of the house and your own painting, at least of the interior. We would estimate the initial cleaning of the house to be about $100.

Line 13 — Legal and Accounting: The accounting fees should be about $150 per year. This is providing nothing serious occurs and you don't have any legal problems.

Line 14 — Total Expenses: =$1,530

Line 15 — Net Operating Income: This is the figure that is obtained by subtracting the figure for the total expenses from the figure that you obtained from the gross operating income. This figure is $2,460.

Line 16 — Less Loan Payments: Now you will have to subtract the loan or mortgage payments from the net operating income.

Line 17 — On this line is indicated the amounts for the different mortgages that may be placed on the property.

Line 18 — On this line will go the yearly interest that you are paying on the property. The interest on the second mortgage for the first year is $440. The interest on the first mortgage for the first year is $2,560.

Line 19 — On this line will go the yearly principal payments. To determine the amount of the yearly mortgage payment that goes towards the principal is to subtract the interest from the total mortgage payment. On our property, the total yearly payment for the second mortgage is $745.01. For the first year the principal will be $745.01 - $440 = $341.01. The total yearly payment can be obtained from a table of loan amortization or from a computer program. The amount of principal reduction for the first mortgage will be the total yearly mortgage payment of $2817.67 - $2560 = $257.67.

Line 20 — Total Loan Payment: This line is the summation of the interest and principal payment or the total mortgage payments for the year for both the first and second mortgages.

Line 21 — Gross Spendable Income: By subtracting the mortgage payments from the net operating income will give us the gross spendable income or "cash flow."

In our situation the total loan payments as noted above are $3571.66. Also, as noted on Line 15, the net operating income is $2460. If we subtract our loan payments of $3571.66 from the net operating income of $2460 leaves a negative <$1111>, or you will have to come up with $92 per month to carry your investment. How can you be on your way to financial security if you have to come up with almost $100 per month to let someone else live in your house? Since we are planning for the appreciation of the property and the retirement of the principal of the loan, this monthly payment can be looked at as a forced saving. The only thing you

```
              *** PROPERTY ANALYSIS ***
34 PARK STREET
PURCHASE PRICE:   $40,000
FIRST MORTGAGE:    $32,000 AT  8.00 % FOR 30 YEARS
SECOND MORTGAGE:     $4,000 AT 11.00 % FOR  8 YEARS


GROSS INCOME EXPECTED                        $4,200.00
VACANCY LOSS                                   $210.00
GROSS OPERATING INCOME                       $3,990.00
TAXES                                          $700.00
INSURANCE                                      $100.00
UTILITIES                                       $20.00
ADVERTISING                                     $60.00
MANAGEMENT                                       $0.00
CIVIC ASSOCIATION                              $100.00
MAINTENANCE                                    $300.00
CLEANING SERVICE                               $100.00
LEGAL AND ACCOUNTING                           $150.00
TOTAL EXPENSES                               $1,530.00
NET OPERATING INCOME                         $2,460.00
INTEREST AFTER ONE YEAR                      $3,000.00
PRINCIPAL AFTER 1ST YR.                        $571.66
TOTAL LOAN PAYMENT                           $3,571.66
GROSS SPENDABLE INCOME                      -$1,111.66
GROSS EQUITY INCOME                           -$540.00
DEPRECIATION                                 $1,333.33
TAXABLE INCOME                              -$1,873.33

              *** TAX ANALYSIS ***
TAX CONSEQUENSES                               $749.33
AFTER TAX INCOME                               $209.33
APPRECIATION                                 $3,200.00
NET EQUITY INCOME                            $3,409.33

            RATE OF RETURN 56.82 % PER YEAR
```

This is an actual printout of the Property Analysis as produced by the enclosed program.

have to remember is that you will be making this extra payment per month in the early years of the investment. As inflation drives prices up a smaller portion of your rent will go to the mortgage payment.

Line 22 — Plus Principal Payment: At this point you will add back the principal payments. Since this money is going to retire the loans you are really not spending the money but are saving it. Thus, go to line 23 to actually find out what the property is costing.

Line 23 — Gross Equity Income: By adding the gross spendable income which is a negative $1111 plus the principal payments which are $571.68 will give us a negative $540 . Thus, the actual cost to carrying the property is $540 divided by 12 or $45 per month.

Line 24 — Less Depreciation: As noted elsewhere, depreciation is a means that the government gives you to recover your original investment. In theory, the government allows you to set aside some money each year so that when the house wears out you will hve enough money to purchase a new house. In reality, however, the house will appreciate in value and so this merely becomes an accounting procedure. For used residential property we have chosen the 125 per cent declining balance method of depreciation and the useful life of the house to be 30 years. Since only the house can be depreciated, you will have to look at the most recent tax assessment notice to find out how much value has been attached to the land and how much for the house and improvements.

In the situation of our example, the house represents 80 per cent of the value and the land 20 per cent. Thus, since the house costs us $40,000, the amount that can be depreciated is $40,000 × 80 per cent = $32,000. Since we gave the house a life of 30 years, this means that the house will lose 1/30 per year of its value. (Example: $32,000 divided by 30 = $1066.67). But since the government allows you to take 125 per cent of the depreciation amount for used homes, the actual amount that you can take off on your taxes is 125 per cent times $1066.67 or $1333.33. Thus, you will subtract this depreciation from the gross equity income to have the figure for either the total profit or the total loss on the property.

In our situation, since we have a gross equity income of a negative $541 (line 23) and since depreciation is a loss (or a negative number), the total loss (both real line 23 and imaginary line 24) is <$1873.33>.

At this point we know that we will be able to write $1873.33 off the income tax. If you are in the 40 per cent bracket, then you save $749 on taxes. If you go back to line 21, you see that you have to pay $92 per month out of pocket of $1111 per year to carry the house. But you save $749.33 in taxes, so in reality you only had to come up with $363 or $30.25 per month.

Tax Analysis

In order to project how much profit you can expect in the future, we now do what is called a tax analysis. This will give us the approximate rate of return on our investment.

Line 1 — Gross Equity Income: This figure is taken from line 23 of the Property Analysis. This line gives us a negative <$540>.

Line 2 — Tax Consequences: As we noted, if you are in the 40 per cent bracket, you will save $749.33 on the taxes.

Line 3 — After Tax Income: By adding Line 1 and Line 2 you will get your after tax income which is $209.33. (Remember that this figure takes principal repayment into account.

Line 4 — Growth: We will estimate at this point that our property will increase by 8 per cent (conservative) annually for the foreseeable future. $40,000 at 8 per cent increase annually is $43,200, or a growth of $3,200 for the first year.

Line 5 — Net Equity Income: By adding line 3 and line 4 you will now get the total buildup of equity of your property:

209.33 + $3,200 =
$3,409.33 increase annually.

Line 6 — Rate of Return: By following this formula you will be able to estimate the rate of return on your investment.

FORMULA:

$$\frac{\$3,409 \times 100}{\$4,000 + \$2,000} = 56.82 \text{ per cent}$$

It should be noted that there are large income tax deductions the first year because of the costs involved with settlement. Closing costs in our area average about 5 per cent of the purchase price.

To be sure, the rates of return of your real estate investment will vary

```
5 '********   PROPERTY ANALYSIS   *********
6 'WRITTEN BY KENNETH KAPLAN AND MAYER LIEBMAN
7 'JULY 1979
10 CLEAR 500
20 DEFINT J
30 DEFDBL L
40 DIM M(2),R(2),R1(2),R2(2),MO(2),PPMT(2),AMT(31,2),
50 DIM IN(30,2),B(14),A$(14),Y(2)          NET(30,2)
60 JFLAG=1
65 CLS
66 PRINT" INVESTMENT PROPERTY ANALYSIS"
67 PRINT" BY KENNETH KAPLAN & MAYER LIEBMAN
68 FOR N=1 TO 500: NEXT N
69 CLS
70 INPUT "ADDRESS OF PROPERTY";P$
80 INPUT "PURCHASE PRICE $ ";PUR
90 INPUT "FIRST MORTGAGE $ ";M(1)
100 INPUT "INTEREST RATE % ";R1(1)
105 INPUT "YEARS";Y(1)
110 R(1)=R1(1)/100
130 MO(1)=Y(1)*12
140 INPUT "IS THERE A SECOND MORTGAGE ";C$
150 IF LEFT$(C$,1)="N" THEN GOTO 230
155 CLS
160 INPUT "SECOND MORTGAGE $ ";M(2)
170 INPUT "INTEREST RATE % ";R1(2)
180 R(2)=R1(2)/100
190 INPUT "YEARS";Y(2)
192 CLS
195 PRINT "COMPUTING MORTGAGE INFORMATION"
200 MO(2)=Y(2)*12
210 JFLAG=2
220 REM COMPUTE YEARLY PAYMENTS
230 FOR J=1 TO JFLAG
240 R2(J)=R(J)/12
250 K=1. + R2(J):L=K^MO(J):N=M(J)*R2(J)*L
255 PPMT(J)=N/(L-1)
256 PPMT(J)=PPMT(J)*12
260 NEXT J
270 REM COMPUTE ANNUAL INTEREST AND PRINCIPAL
280 REM FOR 30 YEARS
290 FOR JK=1 TO JFLAG
300 JL=Y(JK)
310 AMT(1,JK)=M(JK)
320 FOR J=1 TO JL
```

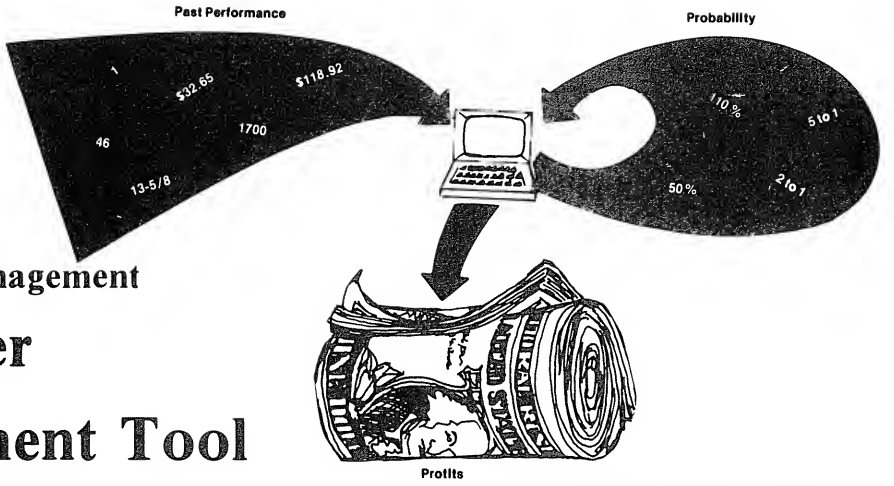
over time. As years go by, the amount you can depreciate will drop and so will the interest deduction on your mortgage loan. Most importantly, you'll face a capital gains on your profits when you choose to sell. Still, your real estate investments should outperform virtually any other vehicle. And it depends on only a few conditions:

- That you buy a home in good condition at fair market value in a good location.
- That you leverage the house, borrowing at least 80 per cent of the purchase price.
- That Congress continues to grant depreciation deductions for the full amount of the purchase price.
- That inflation continues at a rate of about 8 per cent per year.
- That the American Dream will never end, and that people will continue to want the joys of living in a private home. □

This article is based on the book, "How To Invest In The American Dream For Financial Security," by Mayer Liebman, D.D.S., and Barry Feldman.

```
330 IN(J,JK)=AMT(J,JK)*R(JK)
340 NET(J,JK)=PRNT(JK)-IN(J,JK)
350 AMT(J+1,JK)=AMT(J,JK)-NET(J,JK)
360 NEXT J
370 NEXT JK
375 CLS
380 DATA "GROSS INCOME EXPECTED ","VACANCY LOSS
381 DATA "GROSS OPERATING INCOME","TAXES
382 DATA "INSURANCE              ","UTILITIES
383 DATA "ADVERTISING            ","MANAGEMENT
384 DATA "CIVIC ASSOCIATION      ","MAINTENANCE
385 DATA "CLEANING SERVICE       ","LEGAL AND ACCOUNTING
386 DATA "TOTAL EXPENSES         ","NET OPERATING INCOME
430 FOR J=1 TO 14
440 READ A$(J)
450 NEXT J
460 FOR J=1 TO 12
470 IF J=3 THEN GOTO 510
480 PRINT A$(J);
490 INPUT " $ ";B(J)
500 CLS
510 NEXT J
520 B(3)=B(1)-B(2)
530 FOR J=4 TO 12
540 B(13)=B(13)+B(J)
550 NEXT J
560 B(14)=B(3)-B(13)
570 REM INTEREST AFTER ONE YEAR
580 IT=IN(1,1) + IN(1,2)
590 REM PRINCIPAL AFTER ONE YEAR
600 PR=NET(1,1) + NET(1,2)
610 REM TOTAL LOAN PAYMENT
620 TLP=IT + PR
630 REM GROSS SPENDABLE INCOME
640 GSI=B(14)-TLP
650 REM GROSS EQUITY INCOME
660 GEI=GSI + PR
670 REM DEPRECIATION--WE ARE USING 80% IMPROVEMENT VALUE AND 30 YEAR LIFE
671 REM A 125% DECLINING BALANCE IS USED
672 INPUT "DO YOU WISH TO ENTER YOUR OWN FIGURES FOR DEPRECIATION"; U$
674 IF LEFT$(U$,1) = "Y" THEN GOTO 680
675 INPUT "WHAT IS THE VALUE TO THE LAND";D3
676 INPUT "WHAT % RATE WILL YOU BE USING (100,125,200) $";D4:D5=D4/100
677 INPUT "HOW MANY YEARS WILL YOU DEPRECIATE THE PROPERTY"; D6
678 DE=(D3/D6)*D5
679 GOTO 690
680 DE=((PUR*.80)/30)*1.25
690 REM TAXABLE INCOME
700 TI=GEI - DE
710 REM TAX ANALYSIS
715 CLS
720 INPUT "WHAT IS YOUR TAX BRACKET % ";TX
730 REM TAX CONSEQUENSES
740 TC=TI * (TX/100) * -1.
750 REM AFTER TAX INCOME
760 AT= GEI + TC
770 REM GROWTH
780 PRINT "WHAT IS THE PERCENT APPRECIATION FOR HOMES"
782 INPUT "IN YOUR AREA %";AP
790 GR=PUR * (AP/100)
800 REM NET EQUITY INCOME
```

```
810 NE= AT + GR
820 REM RATE OF RETURN
830 PRINT "WHAT ARE THE CLOSING COSTS ? "
835 PRINT "IF UNKNOWN KEY 2 "
837 INPUT CC
840 IF CC= 2 THEN CC = PUR * .05
850 R5=PUR-M(1)-M(2):RG=R5 + CC : R7 = NEI/RG
860 RR = R7 * 100
870 CLS
880 PRINT CHR$(23)
890 PRINT "THE RATE OF RETURN FOR THE"
900 PRINT "PROPERTY LOCATED AT" : PRINT : PRINT P$:PRINT
910 PRINT "WILL BE: ";
920 PRINT USING "##.## % PER YEAR";RR
930 INPUT "KEY ENTER";E$
935 CLS
940 PRINT "DO YOU WANT TO REVIEW THE"
950 PRINT "PROPERTY ANALYSIS ON THE CRT"
970 INPUT D$
980 IF LEFT$(D$,1)="Y" THEN GOSUB 1090
990 PRINT "WOULD YOU LIKE A PRINTOUT OF THE ANALYSIS ?"
1000 INPUT D$
1010 IF LEFT$(D$,1) = "N" THEN GOTO 1520
1015 CLS
1016 JP=1
1020 PRINT CHR$(23)
1030 PRINT "PLEASE MAKE SURE THAT THE"
1040 PRINT "***  PRINTER IS ON  ***"
1045 INPUT"KEY ENTER";E$
1050 P1=PEEK(16422):P2=PEEK(16423)
1060 POKE 16414,P1:POKE 16415,P2
1070 GOSUB 1090
1080 POKE 16414,88:POKE 16415,4
1085 GOTO 1520
1090 IF JP<>1 THEN INPUT "KEY ENTER";E$:CLS
1100 PRINT "*** PROPERTY ANALYSIS *** ":PRINT:PRINT P$
1110 PRINT "PURCHASE PRICE:";
1120 PRINT USING "$$###,###";PUR
1130 PRINT "FIRST MORTGAGE: ";
1140 PRINT USING "$$###,### AT ##.## % FOR ##
     YEARS";M(1),R1(1),Y(1)
1150 IF JFLAG <> 2 THEN GOTO 1180
1160 PRINT "SECOND MORTGAGE: ";
1170 PRINT USING "$$###,### AT ##.## % FOR ##
     YEARS";M(2),R1(2),Y(2)
1180 PRINT:F$="$$##,###.##":PRINT:PRINT:
1185 IF JP<>1THEN INPUT "KEY ENTER";E$
1186 CLS
1190 FOR J=1 TO 14
1200 PRINT A$(J);
1210 PRINT TAB(32);" ";
1220 PRINT USING F$;B(J)
1230 NEXT J
1235 IF JP<>1THEN INPUT "KEY ENTER";E$
1240 CLS
1250 PRINT "INTEREST AFTER ONE YEAR";,,
1260 PRINT USING F$;IT
1270 PRINT "PRINCIPAL AFTER 1ST YR.";,,
1280 PRINT USING F$;PR
1290 PRINT "TOTAL LOAN PAYMENT     ";,,
1300 PRINT USING F$;TLP
1310 PRINT "GROSS SPENDABLE INCOME ";,,
1320 PRINT USING F$;GSI
1330 PRINT "GROSS EQUITY INCOME    ";,,
1340 PRINT USING F$;GEI
1350 PRINT "DEPRECIATION           ";,,
1360 PRINT USING F$;DE
1370 PRINT "TAXABLE INCOME         ";,,
1380 PRINT USING F$;TI
1385 IF JP<>1THEN INPUT "KEY ENTER";E$
1390 CLS: IF JP=1 THEN PRINT:PRINT:PRINT
1400 PRINT TAB(10);" *** TAX ANALYSIS *** "
1410 PRINT "TAX CONSEQUENSES ";,
1420 PRINT USING F$;TC
1430 PRINT "AFTER TAX INCOME ";,
1440 PRINT USING F$;AT
1450 PRINT "APPRECIATION     ";,
1460 PRINT USING F$;GR
1470 PRINT "NET EQUITY INCOME";,
1480 PRINT USING F$;NE
1485 PRINT:PRINT
1490 PRINT:PRINT "RATE OF RETURN ";
1500 PRINT USING "##.## % PER YEAR ";RR
1505 IF JP<>1THEN INPUT "KEY ENTER";E$
1510 RETURN
1520 CLS
1530 PRINT CHR$(23):PRINT " *** GOOD LUCK ***"
1540 GOTO 1540
```



Stock Selection and Money Management

The Microcomputer As an Investment Tool

Robert M. Sharp

Applying computers to investment problems is a natural alliance, typified by accounting systems, information retrieval and screening of stock selections by user given criteria. The advent of the microprocessor extends that use to an intimate, interactive arrangement between the individual investor and his computer, enabling theory testing and game playing with investment opportunities. Ideas can be quickly explored in a laboratory environment before hard cash is risked on an unproven thesis. The following discussion illustrates the interfacing of the microprocessor to records of past performance to yield probabilities and then managing those probabilities to produce profit.

The investment problem consists of selection and money management. Selection is an art with small expectations while money management is a science capable of pyramiding a small advantage into significant profits.

First, let's look at the selection process. There are two major factors requisite to successful selection: Market direction and intrinsic value (quality) of the investment. Market direction tends to dominate but is more reversible and consequently is ignored by some investors who assume that quality stocks will eventually progress with good market conditions. I believe in determining odds of market advance and hedge investments accordingly, removing some risk when odds appear to be poor and leveraging heavily when odds appear to be favorable. The obvious question at this point is how do you determine the odds?

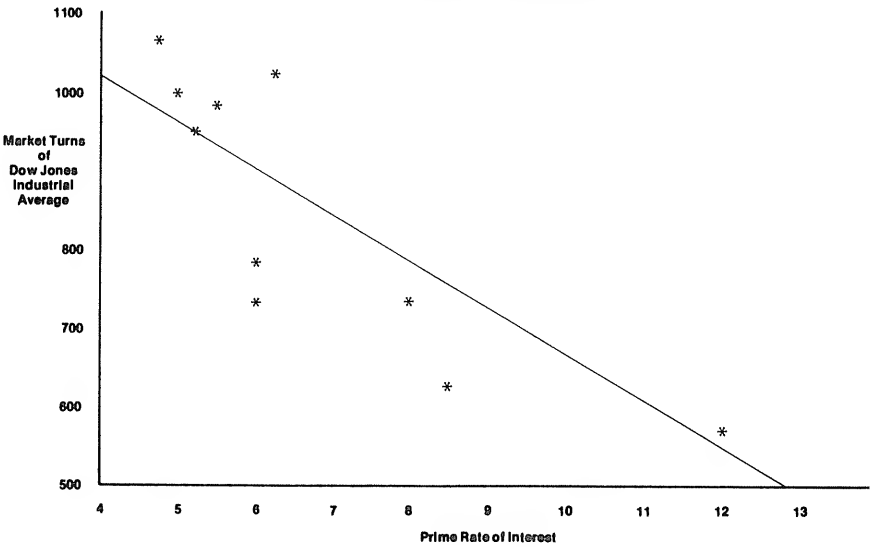
There are two theories concerning market movement:

1. An efficient market correctly representing current values under given economic conditions, and
2. A market of random movements behaving more or less like a roulette wheel.

I think that both are correct in a combination that develops as follows:

The market is obviously affected by economic conditions, particularly the availability of money, although it doesn't step precisely in time with those conditions. Random events, user psychology, etc. tend to create a random walk around the bias line dictated by the economic environment. The trick is to determine the bias line and the odds of significant departure in either direction from that line. The problem simplifies with selection of predictor variable(s) that are themselves more predictable and readily available (documented). Now the computer becomes useful as you employ regression analysis to inspect and discard a number of economic indicators to determine their relationship with market turns. My own microcomputer analysis of the period since the prime rate has been allowed to float indicated the prime rate as highly correlated with the market movement. (Exhibit 1). Remember, we are still in the art phase, so you may select other variables. The prime rate is a much smoother changing variable than the Dow Industrial average and tends to reach high points ahead of the Dow reaching bottoms (12% in July, 1974 and market bottom in December). When given a set of points, e.g., prime rates

Exhibit 1. Scatter Diagram with Progression Line
Dow Jones Industrial Average As A Function of Prime Rate of Interest



Robert M. Sharp, S. 4515 Stone, Spokane, WA 99203.

near turning points in the Dow Jones Industrial Index, Regression Analysis measures the effect the predictor (prime rate) has on the response variable (Dow Average) by a coefficient of correlation and computes a line of regression that best fits the given points. If you can estimate the extreme value of the prime rate, then the program returns a forecast of the turning points in the Dow Average and plots the given points and regression line for visual assurance (Exhibit 1).

Having used regression analysis to satisfy myself that the prime rate of interest was a good measure of economic influence on the market, I decided to apply random (unpredictable) daily fluctuations to the regression line constructed with prime rates. Using the RND function, I generated two sets of random numbers. The first set was partitioned into direction (up or down) of market movement proportionate to the expected change produced by the prime rate.[1] The second rate is the absolute amount of daily change of the Dow Industrial Index in the same percentages as actually occurred in the year 1967.[2] This then becomes a model of the stock market that combines economic influence as measured by the Prime Rate of Interest and a daily Random Walk, unpredictable but governed by the laws of random events. Each execution of the model creates a unique possibility for market movement.

We now have stock market simulation but how do we use it?

The answer is that we treat it just like scientists would treat a weather or traffic simulation. That is, we study it for extremes and repetition that establishes probabilities for a certain level to be reached (60% chance of rain today!). The extremes allow us to set our defenses against a market that falls completely out of bed, while our investment strategy can be adjusted to the probabilities of a favorable investment climate. It is fascinating to see how far the market can wander from the line of bias, but run enough cases and the probability pattern begins to emerge. Once the odds are established and when the edge is in your favor, then we look at the individual stocks.

There are many methods for screening stocks, some of which are hard to quantify and hence difficult to place on the computer. One quantifiable method was discussed in **Business Week** and has been incorporated into my investment package. This routine combines book value of the stock with current money rates to establish a market value. My experience indicates it works well with established, less volatile stocks, but not so well with others. The art becomes very foggy, but your micro shines some light for you as you use this routine as one more check of intrinsic value on your "hot stock." I combine this with other factors such as price compared to past cyclical extremes, price earnings ratios, product potential, etc., to estimate a "floor" price and a "potential" price. The differences between current price and potential divided by the difference between current price and floor is my expectation factor. That ratio has to be better than 2 to 1 to even consider the stock further. The more extreme this ratio (undervalued) the more money I invest (bet) on this stock; and that leads us into money management.

Before you can manage anything you have to understand it. That means reviewing past performance. If you want to understand the market, spend time in the library reading old **Wall Street Journals** until you find a period of time that has roughly the same concerns that we encounter today. If you want to adopt an investment philosophy look at your past record to secure the successful and work on the failures. Analyze your investment transactions for percentage of successful selections, time spent with both succeeding and failing ventures and annual rates of return netted. Look at commissions to see if churning your account ate away your profits. Most investors will probably learn that a good yielding investment socked away from sight would be the best course. However, if you intend to play the game know your numbers. For example, my success rate on selection has been 60.9 for nearly 20 years with hardly any variation. For some time though, I did not net enough profit to warrant my activity. I held losing situations too long (I'll sell as soon as I get even!), was too timid on good prospects and my timing was atrocious. I still pick at the 60.9 rate (art) but have significantly increased percentage profits (science). Specifically, the computer and I are concerned with the following money management matters:

Risk Analysis

One of the foremost principles of investment is to "protect your capital." A second useful concept is the application of leverage, i.e., borrowed money. The second item tends to aggravate the first. The solution is portfolio risk analysis using volatility factors, market expectations, margin maintenance rates and margin debt to determine threshold values where forced liquidation will begin. You must anticipate and have your investments maximized at market bottoms to maximize return. In 1974 I was continually in margin trouble but managed to struggle through fully invested, fully leveraged. The computer routines developed from that experience computes the market risk (beta) of individual stocks, combines them with the weight of each stock in your portfolio and then tells you what the portfolio value will be at certain market levels and what level will trigger the dreaded margin call. The routine allows you to iterate by selecting stocks for sale, recomputing expectations until you have found a combination that is comfortable.

Tax Analysis

The small investor must watch his commission situation carefully, avoiding odd-lots, buying higher priced stocks, buying fewer stocks to optimize the transaction amount; but as he becomes successful he must switch his alertness to taxes. The guiding principles become: every gain should be long term; every loss should be short term; your tax bracket must be continually in mind. The way to accomplish this is to have an up-to-date record of all yearly transactions and be ready to fill out the Form 1040 at the first hint of action. Terrible thought that it is, the computer can shorten this to a one minute process allowing you to check for the best year to liquidate your investment and giving your current tax brackets for both years as a result. With the change in tax law effected in November 1978, I wore this program out deciding whether to sell prior to November 1, prior to December 1, or after January 1. The action anticipated the same price received for my shares, as I went short against the box in October, and then decided when to deliver the stock for tax purposes. Computing next year's expected tax bracket allows you to

analyze your borrowed money against net yield of income producing investments to see if you can make anything on the other fellow's money. It is great to borrow from Merrill Lynch at 12%, buy a utility, with a record of dividends being treated as tax deferred capital appreciation, having a face yield of 9.5% and knowing that Uncle Sam will give you enough credit on interest charges to net you over 2% on the borrowed money. You assume the risk for the investment but you also pocket any capital gain likely to accrue to a depressed issue.

Since most stock dividends are only partially tax deferred the program produces a table given tax bracket, face yield, and current rate of borrowed money. Rows of data are produced, one for each ten percent of tax deferred dividend and a very revealing analysis falls out of a messy calculation. (See Exhibit 2).

ENTER YIELD OF INVESTMENT? 9.5
ENTER RATE OF BORROWED MONEY? 12
YIELDS FOR THE 45 PERCENT TAX BRACKET

EXEMPT 1	NET YIELD 2	BREAK 3	GAIN 4
0	5.225	9.5	— 1.375
10	5.6525	10.2773	— .9475
20	6.08	11.0545	— .52
30	6.5075	11.8318	— 9.25007E-02
40	6.935	12.6091	.335
50	7.3625	13.3864	.7625
60	7.79	14.1636	1.19
70	8.2175	14.9409	1.6175
80	8.645	15.7182	2.045
90	9.0725	16.4954	2.4725
100	9.5	17.2727	2.9

1 Percentage that is deferred as capital gain.
2 Effective yield after 45 percent tax.
3 Comparable fully taxed yield.
4 Net returned to investor after tax applied.

Exhibit 2 **TAX DEFERRED YIELDS**

Time Related Investments

Time is money and most of us forget this when we hold a losing investment until it returns to what we paid for it. The comparison to be made is what value would the principal have compounded over the period of time we held the stock. Hence, your computer tool kit must have a compound interest routine which enables you to make comparisons. This is incorporated into an exchange analysis program to see if swapping investments would be profitable by comparing capital gains or losses after taxes, dividends anticipated and reinvestment of any differential between two investments. A microcomputer with BASIC can undertake this exercise without oversight or error.

Other Money Management Matters

Obviously there are money management techniques not discussed here. Having selected a stock and a total amount to commit to it, I like to buy 5-10 times over a 3-9 month period to average out fluctuations and give me time to confirm my initial impressions. This might be regarded as diversification of time and has been more effective for me than diversification of issues. I also like to set bids below the market with the assumption that any entry into a stock will occur between the high point or low point of a short range cycle and therefore the price will be better in the near future. I will not go into detail on this technique because I have not automated the process, but for some time I have been searching for an equation that ties the investment analysis into a neat package for a microcomputer. The variables would be: total money available for investment, market probability factor, tax bracket, cost of money; and a set of investment candidates with volatility factors, expectation factors, current prices and yields. The program would indicate stock purchases to be made and amounts of each. All artistic effort would be reflected in the inputs given and the management science would be assigned to the computer. Mundane matters of tax law and commission rates would become grist in the profit equation as your portfolio expands and contracts in reaction to expected changes in market conditions. Now this will be some program!

This concludes a very brief treatment of how your microcomputer may be applied to investment questions. I have found it to be a profitable application, but am convinced it is only indicative of better things to come. □

Footnotes

1. This bias can be changed to a different variable or line by replacing one statement in the program.

2. 1967 was used as I had a record of all daily changes and the range was representative of more recent trading years. The percentages are in table form in DATA statements and can be replaced.

3. "Inside Wall Street," **Business Week**. January 23, 1978.

Stock Tracking Program

Adventures in Investing

William K. Mason

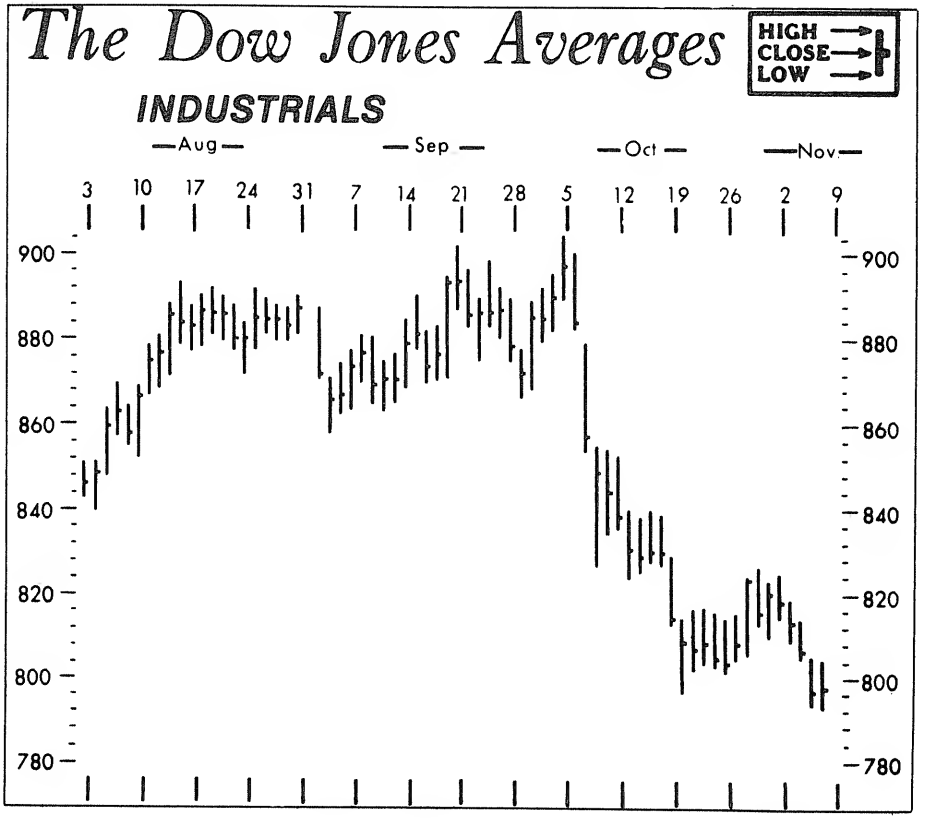
Stock trading, dividends and interest have been my main source of income for the last six years. I'm not broke but I haven't made a million either. When personal computers first began to appear I had a vague idea they might help me in picking stocks... but they were kits. The thing I solder best is my left forefinger, so I waited. In due course little computers proliferated like rabbits and I got my feet wet with a 4K, level I TRS-80. My subsequent career in investing by computer can be divided into three parts. First, the time of grandiose fantasy; second, the time of deepening cynicism; and third the time of healthy realism.

In the beginning there was multiple regression. Multiple regression is a mathematical technique. Your goal is to find an equation relating variables you know (e.g., the consumer price index and the gross national product at the end of 1978) to the variable you want to know, such as the price of IBM at the end of 1979. Multiple regression gives you the equation that would have worked best in the past.

You can buy software to do multiple regression, but I decided to write my own. After a month I had a program that would fit into the 4K memory of the TRS-80 with enough room left over for 19 years of annual data concerning 12 variables. I picked 12 variables from "Annual U.S. Economic Data" put out by the St. Louis Federal Reserve Bank. I typed in RUN, entered the data and the TRS-80 went to work. In five minutes the answer came on the screen. It said, "IN NINE MONTHS THE DOW JONES STOCK AVERAGES WILL BE AT 850."

It's hard to describe the feeling of power you get when you know the future. For a month nothing disturbed me. But then one day when I was glancing through a back issue of **Value Line Investment Survey**, the roof fell in. The folks at Value Line know about multiple regression too! Even worse, their equation fit the data better than mine! I found that **Value Line** has been using multiple regression and computers since the fifties, and that all sorts of firms have sprung up with words like "econometrics" in their titles. These outfits use computers a lot bigger than my 4K TRS-80. How could I compete with them?

At first, frustration led to cyni-



William Mason, Box 316, Hornitos, CA 95325.

cism. As a nasty test I loaded all the information I could about the thirty-one stocks in my portfolio into the computer's 4K memory. Then I wrote the same type of information onto a small index card file, one card for each stock. I compared the two systems in three categories: access time (including loading tape into memory), amount of information stored and cost. The card file won in each category. I tried to give the computer to my wife, suggesting she could store her recipes in it. She preferred index cards, pointing out a category I hadn't thought of: portability around the kitchen. I went to bed that night resolved to phone my brother-in-law in the morning and get him to take the silly thing off my hands.

During the night, however, I remembered a question my wife had asked. Did the profits we made on the Del Monte tender offer make up for the wad we had dropped in the Marine Midland debacle? She knew the answer, of course. The question was her way of taking a jab at Uncle Herbie. Uncle Herbie had recommended Marine Midland. We argued a lot about Uncle Herbie's stock recommendations.

Then I saw the light. I could divide my thirty-one stock portfolio into groups, or "Accounts," as I decided to call them. Account number 1 would be stocks recommended by Uncle Herbie, account number 2 would be stocks selected by my wife, and so on. The TRS-80 could compute the value of each of these accounts once a month and after a while I would know who was outperforming whom.

But there was a catch. Suppose we had $10,000 invested in stocks recommended by Uncle Herbie, then we sold $2000 of his stocks to pay taxes. Uncle Herbie's account would have dropped by $2000 but that wouldn't be his fault. To fix this I pretended each account was a tiny mutual fund with an arbitrary initial number of shares, for example, 1000 each. Then each share in Herbie's "fund" was worth $10,000 divided by 1000 or $10. I call this the "account value per share." Every time I put money in or took money out of the account I changed the number of shares or "account divisor" so the account value per share remained the same. If I took $2000 out of Uncle Herbie's account I would change the account divisor to 800 so that the account value per share would still be $10 = $8000/800. The account value per share varied only if the stocks in the account went up or down (mostly down in Herbie's case - why is my wife always right?).

Quickly I whipped up my Level I stock tracking program which is included in this article. Although the stocks listed in the data statements are mostly fictitious they provide good examples. If you look at instructions 50 through 70 you can see the choices offered by the program.

Four sample accounts are listed in instructions 100 through 106. Taking them in reverse order (starting with account number 4 which is all stocks on my list) is the most important, of course. Since I'm the nervous type I find its current value once a week.

Account number 3 (stocks I should have sold, but didn't), palliates a standard investor problem. You have a big loss on a stock and you know you should dump it, but you can't bring yourself to do so. Sell it to your computer by putting it in account number 3. If your judgement is good the value per share of this account will decline while the value per share of your other accounts advances. If your judgement isn't good the record will speak for itself.

Account number 2 is stocks chosen by Method Y. Method Y can be low price-earnings ratios, high past earnings growth, companies incorporated before the Age of Aquarius, coin flipping, or anything.

Account number 1 is stocks recommended by Mr. X. Our Mr. X is no longer Uncle Herbie. We switched to a firm using big computers with a vast data base. They do the multiple regression for us. So far their picks are running neck and neck with coin flipping.

Finally, a couple of technical remarks. The stock list is on Data statements as part of the program. The format is: DATA stock name, date bought, number of shares, purchase price per share, number of account it belongs in. When you add stocks or take stocks out of an account you must change the account divisor and put in new data statements. The steps are:

```
10 REM: STOCK MONITOR BY W. K. MASON
12 REM: N=NUMBER OF STOCKS, M= NUMBER OF BONDS
14 REM: A(N+1) THROUGH A (N+4) = ACCOUNT DIVISORS
20 N=15: M=3
25 A(N+1)=1000: A(N+2)=1000: A(N+3)=1000: A(N+4)=1000
30 FOR L=1 TO N: A(L) =0: NEXT L
40 CLS: PRINT: PRINT: PRINT "CHOICES": PRINT
50 PRINT "1= LIST ACCOUNT NUMBERS"
52 PRINT "2= INPUT CURRENT PRICES"
54 PRINT "3= LIST STOCKS WITH CURRENT PRICES AND PROFITS"
56 PRINT "4= FIND CURRENT VALUE AND VALUE PER SHARE OF AN
ACCOUNT"
58 PRINT "5= LIST STOCKS WITH ORIGINAL PRUCHASE DATA"
60 PRINT "6= LIST ACCOUNT DIVISORS/FIND NEW ONES"
62 PRINT "7= LIST BOND AND CASH DATA"
70 PRINT: INPUT "WHAT IS YOUR CHOICE";J:CLS
80 ON J GOTO 90,120,160,,180,210,230,250
90 PRINT:PRINT:PRINT:PRINT "ACCOUNT NUMBERS":PRINT
100 PRINT "1=STOCKS RECOMMENDED BY MR. X"
102 PRINT "2=STOCKS CHOSEN BY METHOD Y"
104 PRNT "3=STOCKS I SHOULD HAVE SOLD"
106 PRINT "4=ALL STOCKS ON MY LIST"
110 GOSUB 260:GOTO 40
120 FOR L=1 TO N
130 READ A$,B$,S,P,A
135 PRINT "ENTER PRICE OF ";A$
140 INPUT A(L):NEXT L
150 RESTORE:GOTO 40
160 B=1:C=1:GOSUB 170:GOSUB 260:CLS
162 B=12:C=N:GOSUB 170:GOSUB 260
164 RESTORE:GOTO 40
170 PRINT"STOCK","DATE BOT",""CURRENT PRICE","% PROFIT"
172 FOR L=B TO C
174 READ A$,B$,S,P,A
176 PRINT A$,B$,A(L) INT(100*(A(L)/P-1))
178 NEXT L:RETURN
180 GOSUB 190
182 PRINT:PRINT"ACCOUNT ";X;" CURRENT VALUE=";V:PRINT
184 PRINT"ACCOUNT ";X;" VALUE PER SHARE=";V/A(N+X)
186 GOSUB 260:GOTO 40
190 INPUT"ENTER ACCOUNT NUMBER ";X
192 V=0:FOR L=1 TO N
194 READ A$,B$,.S,P,A
```

1. Input the prices of the stocks in account just before the change (use choice 2).
2. Select Item 6. The computer asks you for the account number and dollar amount in or out of the account. Then it gives you the new account divisor.
3. Put the new account divisor into instruction 25 in place of the old one. Also, if you change the number of stocks in the list you must change instruction 20.
4. Put in the new data statements.

Well, there it is. This program provides a realistic use for the 4K computer in investing. For me it justifies the purchase of a TRS-80. In fact, after recording the value per share of each account for many months I think I have found a stock picking method better than coin flipping. But that's another story. □

```
196 IF(X=A)+(X=4) THEN V=V+S*A(L)
198 NEXT L
200 RESTORE:RETURN
210 B=1:C=11:GOSUB 220:GOSUB 260:CLS
212 B=12:C=N:GOSUB 220:GOSUB 260
214 RESTORE:GOTO 40
220 PRINT "SHARES", "STOCK", "PRICE BOT", "DATE BOT"
222 FOR L=B TO C
224 READ A$,B$,S,P,A
226 PRINT S,A$,P,B$
228 NEXT L:RETURN
230 PRINT"ACCOUNT NUMBER","DIVISOR"
232 FOR L=1 TO 4:PRINT L,A(N+L):NEXT L:PRINT
234 INPUT"DO YOU WISH TO FIND NEW DIVISOR? 0=NO,1=YES";Y
236 IF Y=0 THEN 40
238 PRINT:GOSUB 190
240 INPUT"ENTER DOLLAR AMOUNT IN (+) OR OUT(-) OF ACCOUNT";Y
242 PRINT
246 PRINT"NEW DIVISOR FOR ACCOUNT ";X;"=";Y*A(N+X)/V+A(N+X)
248 GOSUB 260:GOTO 40
250 FOR L=1 TO N:READ A$,B$,S,P,A:NEXT L
252 PRINT"AMOUNT","NAME","DATE DUE"
254 FOR L=1 TO M:READ A$,B$,S
256 PRINT S,A$,B$:NEXT L
258 GOSUB 260:RESTORE:GOTO 40
260 PRINT:INPUT"PRESS ENTER TO CONTINUE";A$:RETURN
480 REM: FORMAT FOR STOCK DATA IS
490 REM: STOCK NAME, DATE BOT, SHARES, PRICE PER SHARE,ACCOUNT #
500 DATA ATT,3-4-76,200,52,2
510 D. ABC,1-2-30,100,10,1
512 D. DEF,1-2-70,100,10,2
514 D. GHI,1-2-70,100,10,3
516 D. JKL,1-2-70,100,10,2
518 D. MN,1-2-70,100,10,2
520 D. OP,1-2-70,100,10,1
522 D. QR,1-2-70,100,10,1
524 D. ST,1-2-70,100,10,3
526 D. UV,1-2-70,100,10,3
528 D. WX,1-2-70,100,10,1
530 D. YZ,1-2-70,100,10,1
532 D. AB,1-2-70,100,10,2
534 D. CD,1-2-70,100,10,3
538 D. IBM, 2-3-70,50,150,2
540 REM:FORMAT FOR BOND DATA IS
550 REM:NAME,DATE DUE, AMOUNT
560 DATA WAR LAM, 4-1-85,10000
562 D. XYZ SAVINGS,NONE,5000
564 D. ABC,1-2-80,10000
```

For Level II, change all "D." to "Data."

Chapter IV
Personal Productivity

A Double Entry System For the Home Accountant

Setting the Record Straight

C.A. Johnson

Everyone who has several small sources of income in addition to his primary employment probably finds, as I did, that keeping books is an unwelcome chore, but one which cannot be avoided if he is to keep on the right side of the Internal Revenue Service.

I had been keeping my books in a standard, double-entry ledger. The system I was using was taken from an example published in a college accounting text book. I made a few minor modifications in the account structure to suit my personal needs, and put it to the test. I found that the double-entry system was delightful for maintaining accuracy, but more time-consuming than I would have liked. Even so, it was a big improvement over the crude, single entry method I had been using. Anyone who has tried to find a subtle error in a single-entry system knows what I am talking about.

When personal computers began to appear on the market at something approaching affordable prices, I knew I had found the answer I needed. The problem remained, however, of how to justify an expenditure of $1000-2000 to keep track of an annual income of $3000-5000. Then word processing programs became available. As a writer, I had my justification (but that is another story). I shopped and finally bought my TRS-80, 16K Level II.

I searched the available software for a suitable accounting program. I found that they all required a minimum of 32K, disk and printer. And they were not cheap. I did not feel ready to put another $2000 into software and more hardware, so I decided to program the system I had been using.

The first problem I encountered was how to handle the required data storage. I estimated that the program would take about 10K of my RAM. Clearly the 5-6K of memory left after loading a program was not enough data storage for much of an accounting program. Tape seemed to provide the answer. Once the decisions were made and the system designed, the programming was a simple matter. It is straightforward programming with no fancy frills.

C.A. Johnson, 3619 Sugarhill Dr., San Antonio, TX 78230.

The Journal Subroutine

The heart of the system is the double-entry journal subroutine. The program is conversational, asking for each data item in turn. For each transaction, debits and credits must balance. When the transaction has been entered, if it balances the program will tell you that it balances and ask if you have another transaction. Regardless of the answer, it will remind you to prepare your recorder for recording.

When you press ENTER, it transmits the data to the recorder. If your reply to the transaction question is "Y," it will initialize and accept new data. If the reply is "N," it will take you back to the menu. Each step is guided by a computer prompt. If you have input a complete transaction, and do not get a "transaction balanced" message, you should check the transaction to see if your data balance, then enter "0" (zero) for the date to reinitialize the subroutine and reenter the entire transac-

81002	PROPRIETORSHIP	31	0.00	4000.00
81002	CASH	11	2500.00	0.00
81002	INVENTORY	12	1000.00	0.00
81002	OFF FURNITURE	14	500.00	0.00
81002	RETAIL SALES	42	0.00	20.00
81002	CASH	11	20.00	0.00
81005	INVENTORY	12	15.00	0.00
81005	CASH	11	0.00	15.00
81006	RETAIL SALES	42	0.00	30.00
81006	CASH	11	30.00	0.00
81007	WHOLESALE SALES	41	0.00	50.00
81007	CASH	11	50.00	0.00
81008	RETAIL SALES	42	0.00	35.00
81008	CASH	11	35.00	0.00
81008	INVENTORY	12	60.00	0.00
81008	CASH	11	0.00	60.00
81012	RETAIL SALES	42	0.00	25.00
81012	CASH	11	25.00	0.00
81012	ADVERTISING	53	12.00	0.00
81012	CASH	11	0.00	12.00
81014	WHOLESALE SALES	41	0.00	80.00
81014	CASH	11	80.00	0.00
81014	INVENTORY	12	125.00	0.00
81014	CASH	11	0.00	125.00
81015	RETAIL SALES	42	0.00	22.00
81015	CASH	11	22.00	0.00
81019	RETAIL SALES	42	0.00	45.00
81019	CASH	11	45.00	0.00
81020	WHOLESALE SALES	41	0.00	65.00
81020	CASH	11	65.00	0.00
81023	RETAIL SALES	42	0.00	68.00
81023	CASH	11	68.00	0.00
81028	WHOLESALE SALES	41	0.00	57.00
81028	CASH	11	57.00	0.00

Figure 1. Sample output from journal subroutine.

tion. The program will accept up to 15 entries in each transaction. The final transaction on each journal tape should be a single record containing a date, "LAST ENTRY" and zeroes in the other three fields.

I make my date entry in Julian date form (YYDDD). For those unfamiliar with it, the Julian date is the date given on desk calendars in the form: "This is the xxx day." The first two characters refer to the year. This format makes it easy to select date ranges for reports from consolidated data tapes. However, the date in YYMMDD form works just as well.

When reading the journal tape, the program will continue to read (or try to) until it encounters a record entry with "LAST ENTRY" in the description field. Consequently, it is a good idea to prepare a tape with a single "LAST ENTRY" record on it as described above. This tape will be used to stop the computer from reading the data tape after all the data have been read if you have neglected to put a "LAST ENTRY" record at the end of the journal tape. Otherwise you will have to reset the computer and start over. With it you can stop reading data at any time.

The Balance Sheet Routine

The "balance sheet" routine can be used to provide summary data, if desired, or it can be used as a trial balance. (Figure 2 shows sample balance sheet output.) In the latter case, the trial balance is adjusted by entering new transactions into a journal tape and rerunning the "balance sheet" routine. Naturally, this process may be repeated as many times as necessary to produce the final balance sheet. The adjustment transactions could be retained as a permanent part of the journal record, or the summarized balance sheet may be input as the beginning transactions so that the accounts are initialized at the beginning of each accounting period.

The "balance sheet" routine will ask for a beginning inventory and a closing inventory. If your business does not have an inventory, just press "ENTER" in response to those questions and it will pass them. Those entries, however, when made *do not* become part of the journal tape.

The program is written for the small cash-basis business. However, it does provide "accounts receivable" and "accounts payable" accounts to accommodate those businesses which are basically cash-basis but involve a bonus which is receivable and/or payable at some time following the close of the accounting period, or for those businesses which borrowed money to get started.

To reduce the number of data tapes which might be tied up with accounting transactions over a period of time, a tape consolidation routine is included. This routine runs independently of the main accounting program. It will accept up to 50 entries at one time and output them to a fresh tape or to the end of an existing tape. This routine is designed to operate comfortably in 16K. If you want to expand it, just change the DIM and CLEAR statements to suit your situation.

Both programs are written in double precision. If none of your accounts totals $10,000 or more for your accounting period, you may speed the programs significantly by converting them to single precision. This is done by loading the program and entering the statement number "40" before entering "RUN."

If the accounts used in the program do not include all of those required and do include some not needed, a substitution on a one-for-one basis is called for. This is a simple matter so long as the account numbers are not changed, or a substitution of one type of account for another type of account is not made. For example, a direct substitution of one kind of expense account for another would be proper, but a substitution of a sales account for a needed expense account would not (at least not without changing the computation in the "balance sheet" routine.)

I have used this program for about fourteen months (six months overlap with my manual system) and have found it trouble-free. I would have liked to have had a printer during the whole time, but there are not many figures to copy to develop the balance sheet, so the absence of the printer was not a severe limitation. The only trouble I have had has not been with the program, but with accounting theory—I keep getting my debits and credits mixed up. I have almost cured myself of that habit, since it is a bit cumbersome to correct that type of mistake once it has been added to the journal tape. The simplest means is to offset it with a correcting entry and then reenter the transaction properly. It can be done with the tape consolidation routine, but I do not recommend it.

A final caution is, if you do not understand double-entry accounting you should get a copy of a good college accounting text book. You will find it a good investment.□

```
ASSETS FOR  81001 THRU  81031
ACCOUNT                DEBIT        CREDIT
CASH                 $2785.00       $0.00
INVENTORY            $1200.00       $0.00
SUPPLIES               $0.00        $0.00
EQUIPMENT             $500.00       $0.00
ACCTS RECVABL          $0.00        $0.00
  TOTAL               $4485.00       $0.00

LIABILITIES FROM  81001  THRU  81031
ACCOUNT                DEBIT        CREDIT
ACCTS PAYBL            $0.00        $0.00
  TOTAL                $0.00        $0.00

PROPRIETOR FROM  81001  THRU  81031
ACCOUNT                DEBIT        CREDIT
CAPITAL                $0.00     $4000.00
DRAWING                $0.00        $0.00
  TOTAL                $0.00     $4000.00

INCOME FROM  81001  THRU  81031
ACCOUNT                DEBIT        CREDIT
WHOLESALE              $0.00      $252.00
RETAIL                 $0.00      $245.00
BONUSES                $0.00        $0.00
  TOTAL                $0.00      $497.00


EXPENSES FROM  81001  THRU  81031
ACCOUNT                DEBIT        CREDIT
SUPPLIES EX            $0.00        $0.00
TRANSPORT EX           $0.00        $0.00
ADVERTISING           $12.00        $0.00
MISC EX                $0.00        $0.00
EQUIP EX               $0.00        $0.00
DISTRIB BON            $0.00        $0.00
  TOTAL               $12.00        $0.00
```

Figure 2. Sample output from balance sheet subroutine.

Program Listing

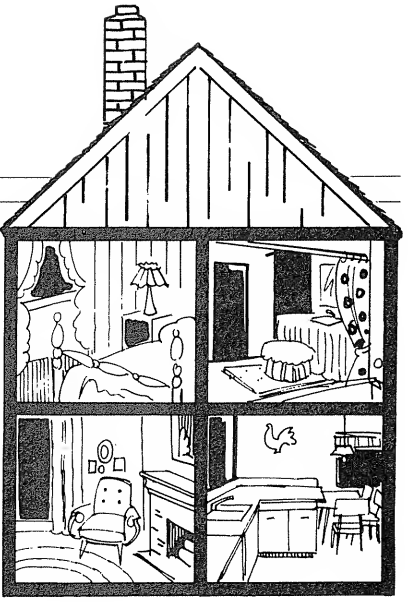
```
10 'PROGRAMMED BY C.A. JOHNSON,PO BOX 5876,SAN ANTONIO,TX 78201
20 CLS
30 CLEAR 1000
40 DEFDBL B,C,D,P
50 A$="$$#######.##":B$="######  %                %   ###   ######.##
 ######.##"
60 C$="ACCOUNT                DEBIT      CREDIT"
70 DEFINT A:DIM JD(15),D$(15),AG(15),DR(15),CR(15)
80 PRINT"* * *  AGGOUNTING SYSTEM FOR AMWAY DISTRIBUTORS * * *":PRINT
90 PRINT"THE SYSTEM CONSISTS OF SEVERAL MODULES.   THE JOURNAL KEEPS
THE DETAILED  RECORDS.   THE OTHER  MODULES  EITHER  COMPUTE NEEDED
VALUES,  PRODUCE  REPORTS  FOR  MANAGEMENT  OR  TAX"
100 PRINT"PURPOSES, OR PERFORM UTILITY FUNCTIONS. IT IS A DOUBLE
ENTRY    SYSTEM,  REQUIRING AT LEAST TWO ENTRIES PER TRANSAGTION.
FOR EACH TRANSACTION,  DEBITS AND GREDITS MUST EQUAL  OR THE SYSTEM
WILL NOT TRANSMIT THE DATA TO TAPE."
110 PRINT:PRINT"IF YOU ARE UNFAMILIAR WITH DOUBLE ENTRY ACGOUNTING,
REFER  TO A COLLEGE AGCOUNTING TEXT.":PRINT:INPUT"PRESS ENTER TO
CONTINUE";T$
120 CLS:PRINT"THE ACCOUNT STRUCTURE IS:":PRINT:PRINT"1.
ASSETS":PRINT"  11. GASH":PRINT"  12. INVENTORY":PRINT"  13.
SUPPLIES":PRINT"  14. EQUIPMENT":PRINT"  15. ACCOUNTS
REGEIVABLE":PRINT
130 PRINT"2. LIABILITIES":PRINT"  21. ACCOUNTS
PAYABLE":PRINT:PRINT"3.PROPRIETORSHIP":PRINT"  31. GAPITAL":PRINT"
32. DRAWING":INPUT"PRESS ENTER TO GONTINUE";T$
140 CLS:PRINT"4. INCOME":PRINT"  41. WHOLESALE SALES":PRINT"  42.
RETAIL SALES":PRINT"  43. BONUSES":PRINT:PRINT"5. EXPENSES":PRINT"
51. SUPPLIES EXPENSE"
150 PRINT"  52. TRANSPORTATION EXPENSE":PRINT"  53. ADVERTISING
EXPENSE":PRINT"  54. MISCELLANEOUS EXPENSE":PRINT"  55. EQUIPMENT
EXPENSE":PRINT"  56. DISTRIBUTOR BONUSES":PRINT
160 PRINT:INPUT"PRESS ENTER FOR THE MENU AND SELECT THE MODULE
WANTED.";T$
170 CLS:PRINT"THE FOLLOWING MODULES ARE AVAILABLE:":PRINT
180 PRINT"1.  JOURNAL ENTRY.":PRINT"2.  GOMPUTE BALANGE
SHEET.":PRINT"3.  JOURNAL DATA TAPE CONSOLIDATION.":INPUT"ENTER
MODULE WANTED ";N
190 ON N GOSUB 210  ,  630  ,  1470
200 CLS:GOTO 170
210 CLS:PRINT"* * *  JOURNAL MODULE  * * *":PRINT
220 PRINT" JULIAN DATE IS ENTERED IN YYDDD FORMAT. IF TRANSACTION
ENTERED BALANCES,  THE PROGRAM TRANSMITS THE DATA TO TAPE.  IF YOU
MAKE A MISTAKE IN DATA ENTRY, ENTER '0' FOR JULIAN DATE AND RE-ENTER"
230 PRINT"ENTIRE TRANSACTION.    AFTER ENTERING LAST TRANSACTION,
MAKE  ONE ADDITIONAL  ENTRY  OF  VALID  JULIAN DATE,  'LAST ENTRY' IN
DESGRIPTION,  AND '0' IN OTHER FIELDS.   DESCRIPTIONS SHOULD BE
LIMITED TO FIFTEEN(15) CHARACTERS."
240 PRINT:INPUT"PREPARE RECORDER TO REGORD DATA.  PRESS ENTER WHEN
READY";T$:GLS
250 FOR I=1TO10
260 INPUT "ENTER JULIAN DATE ";JD(I)
270 IF JD(I)=0 THEN 410  : 'ALLOWS CORRECTION OF KEYING ERROR
280 INPUT "ENTER BRIEF DESCRIPTOR ";D$(I)
290 INPUT "ENTER ACGOUNT NUMBER ";AC(I)
300 INPUT "ENTER DEBIT ";DR(I)
310 INPUT "ENTER CREDIT ";CR(I)
320 D1=D1+DR(I):C1=C1+CR(I)
330 IF C1=D1 THEN 350
340 NEXT I
350 INPUT "TRANSACTION BALANCES. ANOTHER TRANSACTION (Y/N)";T$
360 FOR J=1TOI
370 PRINT#-1, JD(J),D$(J),AC(J),DR(J),CR(J)
380 NEXT J
390 IF T$="Y" GOTO 250
400 RETURN
410 C1=0:D1=0:GOTO 250
420 RETURN
630 GLS:PRINT"* * *  BALANCE SHEET MODULE  * * *":PRINT:PRINT
633 INPUT"DO YOU WANT JOURNAL ENTRIES OUTPUT TO PRINTER(Y/N)";PT$
635 PRINT:PRINT:INPUT"DO YOU WANT BALANCE SHEET OUTPUT TO
PRINTER(Y/N)";PR$:PRINT:PRINT
640 INPUT"INPUT RANGE OF TIME PERIOD FOR BALANGE SHEET
DESIRED(DATE,DATE)";JB,JE
650 PRINT"OPENING AND CLOSING INVENTORIES ARE REQUIRED TO BE"
660 PRINT"MEANINGFUL.":INPUT"INPUT BEGINNING INVENTORY VALUE";DX
670 INPUT"INPUT ENDING INVENTORY VALUE";DY
680 D2=DX-DY:PRINT:PRINT
690 INPUT"PREPARE TO READ DATA TAPE.  PRESS ENTER WHEN
READY.";T$:GLS:PRINT"* ONE MOMENT,PLEASE *":PRINT:PRINT
700 INPUT#-1, JD,D$,AG,DR,CR
710 IF D$="LAST ENTRY" THEN 1100
720 IF JD<JB THEN 700
730 IF JD>JE THEN 700
740 IF AG=11 GOTO 910
750 IF AG=12 GOTO 920
760 IF AC=13 GOTO 930
770 IF AC=14 GOTO 940
780 IF AC=15 GOTO 950
790 IF AG=21 GOTO 960
800 IF AC=31 GOTO 970
810 IF AC=32 GOTO 980
820 IF AC=41 GOTO 990
830 IF AC=42 GOTO 1000
840 IF AC=43 THEN 1010
850 IF AC=51 THEN 1020
860 IF AG=52 THEN 1030
870 IF AG=53 THEN 1040
880 IF AC=54 THEN 1050
890 IF AC=55 THEN 1060
900 IF AC=56 THEN 1070
910 D1=D1+DR:C1=G1+CR:GOTO 1080
920 D2=D2+DR:C2=C2+CR:GOTO 1080
930 D3=D3+DR:C3=C3+CR:GOTO 1080
940 D4=D4+DR:C4=C4+CR:GOTO 1080
950 D5=D5+DR:G5=C5+CR:GOTO 1080
960 D6=D6+DR:C6=G6+CR:GOTO 1080
970 D7=D7+DR:C7=G7+CR:GOTO 1080
980 D8=D8+DR:C8=C8+CR:GOTO 1080
990 DA=DA+DR:GA=CA+CR:GOTO 1080
1000 DB=DB+DR:CB=CB+CR:GOTO 1080
1010 DC=DC+DR:CG=CG+CR:GOTO 1080
1020 DD=DD+DR:CD=CD+CR:GOTO 1080
1030 DE=DE+DR:CE=CE+GR:GOTO 1080
1040 DF=DF+DR:CF=CF+CR:GOTO 1080
1050 DG=DG+DR:CG=CG+CR:GOTO 1080
1060 DH=DH+DR:CH=CH+CR:GOTO 1080
1070 DI=DI+DR:CI=CI+CR:GOTO 1080
1080 PRINT USING B$;JD,D$,AG,DR,CR
1085 IF PT$="Y"THEN LPRINT USING B$;JD,D$,AC,DR,CR
1090 GOTO 700
1100 PRINT:INPUT"LAST ENTRY SENSED. DO YOU HAVE MORE DATA(Y/N)";T$
```

```
1110 IF T$="Y" THEN INPUT"PREPARE RECORDER. PRESS ENTER WHEN
READY";T$:GOTO 700
1120 INPUT"PRESS ENTER FOR BALANCE SHEET";T$:CLS
1130 PRINT:PRINT "ASSETS FOR";JB;"THRU";JE:PRINT C$
1135 IF PR$="Y"THEN LPRINT:LPRINT "ASSETS FOR ";JB;"THRU
";JE:LPRINTC$
1140 PRINT "CASH",;:D1=D1-C1:C1=O:PRINT USINC A$;D1,C1
1145 IF PR$="Y" THEN LPRINT "CASH",;:LPRINT USINGA$;D1,C1
1150 PRINT "INVENTORY",;:D2=D2-C2:C2=O:PRINT USINC A$;D2,C2
1155 IF PR$="Y" THEN LPRINT "INVENTORY",;:LPRINT USINC A$;D2,C2
1160 PRINT"SUPPLIES",;:D3=D3-C3:C3=O:PRINT USINC A$;D3,C3
1165 IF PR$="Y" THEN LPRINT"SUPPLIES",;:LPRINT USINC A$;D3,C3
1170 PRINT"EQUIPMENT",;:D4=D4-C4:C4=O:PRINT USING A$;D4,C4
1175 IF PR$="Y" THEN LPRINT "EQUIPMENT",;:LPRINT USINC A$;D4,C4
1180 PRINT"ACCTS RECVABL",;:D5=D5-C5:C5=O:PRINT USING A$;D5,C5
1185 IF PR$="Y" THEN LPRINT"ACCTS RECVABL",;:LPRINT USING A$;D5,C5
1190 PRINT"  TOTAL",;:DJ=D1+D2+D3+D4+D5:CJ=O:PRINT USING A$;DJ,CJ
1195 IF PR$="Y" THEN LPRINT"  TOTAL",;:LPRINT USING A$;DJ,CJ
1200 PRINT:PRINT"LIABILITIES FROM";JB;"THRU";JE:PRINT C$
1205 IF PR$="Y" THEN LPRINT:LPRINT"LIABILITIES FROM ";JB;" THRU
";JE:LPRINT  C$
1210 PRINT"ACCTS PAYBL",;:C6=C6-D6:D6=O:PRINT USINC A$;D6,C6
1215 IF PR$="Y" THEN LPRINT"ACCTS PAYBL",;:LPRINT USING A$;D6,C6
1220 PRINT"  TOTAL",;:PRINT USING A$;D6,C6
1225 IF PR$="Y"THEN LPRINT"  TOTAL",;:LPRINT USING A$;D6,C6
1230 PRINT:INPUT"PRESS ENTER TO CONTINUE";T$:CLS:IF PR$="Y"
THENLPRINT
1240 PRINT"PROPRIETOR FROM";JB;"THRU";JE:PRINT C$
1245 IF PR$="Y"THEN LPRINT"PROPRIETOR FROM ";JB;" THRU ";JE:LPRINT C$
1250 PRINT"CAPITAL",;:C7=C7-D7:D7=O:PRINT USINC A$;D7,C7
1255 IF PR$="Y" THEN LPRINT"CAPITAL",;:LPRINT USINC A$;D7,C7
1260 PRINT"DRAWING",;:PRINT USING A$;D8,C8
1265 IF PR$="Y"THEN LPRINT"DRAWINC",;:LPRINT USINC A$;D8,C8
1270 PRINT"  TOTAL",;:DL=D7+D8:CL=C7+C8:PRINT USING A$;DL,CL
1275 IF PR$="Y"THEN LPRINT"  TOTAL",;:LPRINT USINC A$;DL,CL
1280 PRINT:PRINT"INCOME FROM";JB;"THRU";JE:PRINT C$
1285 IF PR$="Y"THEN LPRINT:LPRINT"INCOME FROM ";JB;" THRU ";JE:LPRINT
C$
1290 PRINT"WHOLESALE",;:PRINT USINC A$;DA,CA
1295 IF PR$="Y"THEN LPRINT"WHOLESALE",;:LPRINT USINC A$;DA,CA
1300 PRINT"RETAIL",:PRINT USINC A$;DB,CB
1305 IF PR$="Y"THEN LPRINT"RETAIL",;:LPRINT USINC A$;DB,CB
1310 PRINT"BONUSES",;:PRINT USINC A$;DC,CC
1315 IF PR$="Y"THEN LPRINT"BONUSES",;:LPRINT USING A$;DC,CC
1320 PRINT"  TOTAL",;:DM=DA+DB+DC:CM=CA+CB+CC:PRINT USING A$;DM,CM
1325 IF PR$="Y"THEN LPRINT"  TOTAL",;:LPRINT USINC A$;DM,CM
1330 PRINT:INPUT"PRESS ENTER TO CONTINUE";T$:CLS:IF PR$="Y"THENLPRINT
1340 PRINT:PRINT"EXPENSES FROM";JB;"THRU";JE:PRINT C$
1345 IF PR$="Y"THEN LPRINT:LPRINT"EXPENSES FROM ";JB;" THRU
";JE:LPRINT C$
1350 PRINT"SUPPLIES EX",;:PRINT USING A$;DD,CD
1355 IF PR$="Y"THEN LPRINT"SUPPLIES EX",;:LPRINT USING A$;DD,CD
1360 PRINT"TRANSPORT EX",;:PRINT USING A$;DE,CE
1365 IF PR$="Y"THEN LPRINT"TRANSPORT EX",;:LPRINT USING A$;DE,CE
1370 PRINT"ADVERTISING",;:PRINT USINC A$;DF,CF
1375 IF PR$="Y"THEN LPRINT"ADVERTISING",;:LPRINT USINC A$;DF,CF
1380 PRINT"MISC EX",;:PRINT USINC A$;DG,CG
1385 IF PR$="Y"THEN LPRINT"MISC EX",;:LPRINT USING A$;DG,CC
1390 PRINT"EQUIP EX",;:PRINT USINC A$;DH,CH
1395 IF PR$="Y"THEN LPRINT"EQUIP EX",;:LPRINT USING A$;DH,CH
1400 PRINT"DISTRIB BON",;:PRINT USINC A$;DI,CI
1405 IF PR$="Y"THEN LPRINT"DISTRIB BON",;:LPRINT USINC A$;DI,CI
1410 PRINT"  TOTAL",;:DN=DD+DE+DF+DC+DH+DI:CN=CD+CE+CF+CG+CH+CI:PRINT
USINC A$;DN,CN
1415 IF PR$="Y"THEN LPRINT"  TOTAL",;:LPRINT USING A$;DN,CN
1420 PRINT:PRINT:INPUT"END OF BALANCE SHEET. PRESS ENTER FOR
MENU";T$:RETURN
1470 CLS:PRINT"* * * JOURNAL DATA TAPE CONSOLIDATION MODULE * * *"
1480 PRINT:PRINT"  THIS MODULE  WILL  CONSOLIDATE  THE DATA  ENTRIES
FROM SEVERALJOURNAL DATA TAPES INTO A SINCLE JOURNAL DATA TAPE.  IN
ORDER TO HANDLE A FAIRLY LARGE NUMBER OF TRANSACTIONS, AND SINCE THE"
1490 PRINT"FEATURES OF THE MAIN PROGRAM ARE NOT NEEDED FOR THIS
PURPOSE, ITIS INCLUDED AS AN UTILITY ROUTINE ON YOUR PROGRAM TAPE.
PREPAREYOUR RECORDER TO READ YOUR PROCRAM TAPE.  PRESS 'BREAK' AND
LOAD PROGRAM B."
1500 PRINT:INPUT"PRESS ENTER TO RETURN TO MENU";T$:RETURN
```

Journal Tape Consolidation Program

```
10 CLS:PRINT"* * * JOURNAL TAPE CONSOLIDATION PROCRAM * * *":PRINT
20 CLEAR 3000
30 B$="######  %              %   ###   ######.##  ######.##"
40 DEFDBL C,D
50 DEFINT A,I,K
60 DIM JD(50),D$(50),AC(50),DR(50),CR(50)
70 PRINT:PRINT" BEFORE  RUNNING  THIS ROUTINE, YOU SHOULD HAVE A TAPE
PREPARED WITH THE JOURNAL MODULE, CONTAINING ANY LECAL JULIAN DATE,
'LAST ENTRY' IN THE BRIEF DESCRIPTOR FIELD, AND '0' IN EACH OF THE"
80 PRINT"OTHER THREE FIELDS TO STOP INPUT IF NEEDED.":PRINT
90 PRINT:PRINT"  THIS MODULE WILL CONSOLIDATE THE DATA  ENTRIES  FROM
 SEVERAL JOURNAL DATA TAPES INTO A SINGLE JOURNAL DATA
TAPE.":PRINT:INPUT"   PRESS ENTER TO CONTINUE";T$:CLS
100 PRINT:PRINT:PRINT"  THE AMOUNT OF DATA YOU CONSOLIDATE ONTO A
SINGLE CASSETTE WILLDEPEND  UPON SEVERAL  FACTORS,  MOST OF WHICH YOU
WILL DISCOVER FROM EXPERIENCE.  HOWEVER, CONSIDER THESE FACTORS
BEFORE YOU "
110 PRINT"MAKE UP YOUR MIND.  TOO MUCH DATA ON A SINGLE CASSETTE
REQUIRES EXCESSIVE AMOUNTS OF DATA INPUT TIME.   TOO LITTLE DATA ON
EACH CASSETTE WILL TIE UP TOO MANY CASSETTES.  IF YOU ARE NEW IN THE
BUSINESS YOU MAY WANT TO KEEP THREE OR";
120 PRINT" FOUR MONTHS TRANSACTIONS ON A SINCLE CASSETTE.   AS YOUR
BUSINESS CROWS, YOU MAY WANT TO REDUCE THE TIME PERIOD UNTIL YOU WILL
PROBABLY MAINTAIN A CASSETTE FOR EACH MONTH."
130 PRINT:INPUT"PRESS ENTER TO CONTINUE";T$:CLS:PRINT
140 PRINT:PRINT"  FIFTY  ENTRIES WILL FIT ON A SINCLE SIDE OF A C10
TAPE WITH A FAIR MARCIN FOR VARIATION IN THE LENGTH OF THE
DISCRIPTION.  IF THE  CONSOLIDATION  ROUTINE  STOPS  IN THE MIDDLE OF
YOUR INPUT TAPE, REMOVE IT WITHOUT REWINDINC.   WHEN YOU";
150 PRINT" REPLACE IT IN THE RECORDER TO CONTINUE YOUR CONSOLIDATION,
BE SURE YOU REPLACE IT AS IT WAS."
160 PRINT:INPUT"   PRESS ENTER TO CONTINUE";T$:CLS
170 CLS:PRINT:PRINT"  AS THE DATA IS READ FROM  TAPE ,  IT WILL BE
DISPLAYED ON THE SCREEN SO THAT YOU MAY AUDIT WHAT IS BEINC
READ.":PRINT
173 INPUT"DO YOU WANT OUTPUT TO PRINTER (Y/N) ";PR$
180 INPUT"PREPARE RECORDER TO READ FIRST TAPE.  PRESS ENTER WHEN
READY";T$
190 CLS:PRINT"*  ONE MOMENT PLEASE  *"
200 FOR I=1TO50
210 INPUT#-1,JD(I),D$(I),AC(I),DR(I),CR(I)
```

```
220 IF D$(I)="LAST ENTRY" THEN 250
225 IF PR$ = "Y" THEN LPRINT USING B$;JD(I),D$(I),AC(I),DR(I),CR(I)
230 PRINT USING B$;JD(I),D$(I),AC(I),DR(I),CR(I)
240 NEXT I:PRINT"INPUT LIMIT REACHED.":GOTO 270
250 INPUT"LAST ENTRY SENSED.  DO YOU HAVE MORE DATA (Y/N)";T$
260 IF T$="Y" THEN INPUT"PREPARE RECORDER TO READ NEXT TAPE. PRESS
ENTER WHEN READY";T$:GOTO 210
270 I=I-1:INPUT"PREPARE RECORDER TO RECORD DATA.  PRESS ENTER WHEN
READY.";T$
280 FOR J=1TOI
290 PRINT#-1,JD(J),D$(J),AC(J),DR(J),CR(J)
300 NEXT J
310 JD=80000:D$="LAST ENTRY":AC=0:DR=0:CR=0:PRINT#-1,JD,D$,AC,DR,CR
320 PRINT"YOU HAVE CONSOLIDATED   ";J-1;"  ENTRIES ON THIS TAPE."
330 PRINT:INPUT"DO YOU HAVE MORE DATA TO CONSOLIDATE(Y/N)";T$
340 IF T$="N" THEN STOP
350 PRINT:PRINT"  IF THE  CONSOLIDATION  RUN  STOPPED  IN THE MIDDLE
OF A TAPE, REMOUNT THE TAPE WHERE IT STOPPED.   OTHERWISE, MOUNT NEXT
DATA TAPE AND PREPARE THE RECORDER TO READ."
360 PRINT:INPUT"PRESS ENTER WHEN READY";T$:GOTO 190
```



Home Construction Cost Estimate

John W. Essig

This program, written in TRS-80 Level II Basic, estimates the construction cost of a home based on the number of rooms, quality of construction, geographical area, and other factors. The programming is very straightforward, but you might want to improve it by adding yes/no answers or other factors. The sample run was printed using the screen-print feature of NEWDOS.

John W. Essig, 826 2nd St., PO Box 366, Webster City, Iowa 50595.

```
1 REM.  HOME CONSTRUCTION COST ESTIMATOR.
2 REM.  THIS PROGRAM WILL GIVE A REALISTIC ESTIMATE OF THE COST
        TO BUILD THE MAJORITY OF THE SINGLE-FAMILY HOMES SEEN
3 REM.  IN THE USA TODAY.  A USER WHO IS CONSIDERING BUILDING
        A HOUSE MAY REPEAT IT WITH DIFFERENT COMBINATIONS OF
4 REM.  VARIABLES AND VERY QUICKLY DETERMINE IF HIS PLANS WILL
5 REM.  INSURANCE AGENTS WILL FIND IT USEFUL IN HELPING A CLIENT
        DECIDE HOW MUCH COVERAGE TO BUY ON A GIVEN HOUSE.
6 REM.  REAL ESTATE AGENTS MAY USE IT TO SHOW PROSPECTS WHAT IT
        WOULD COST TO DUPLICATE A HOUSE THEY ARE CONSIDERING.
7 REM.  LENDING OFFICERS AND APPRAISERS MAY USE IT TO GET A
        REPLACEMENT BASIS FOR USE IN ESTABLISHING VALUE.
8 REM.  AUTHOR: J ESSIG, BOX 366, WEBSTER CITY, IA 50595
9 CLS:CLEAR:RESTORE
10 PRINT "    ***  HOME CONSTRUCTION COST ESTIMATOR   ***"
15 PRINT " "
20 PRINT "     THIS PROGRAM WILL ASK YOU QUESTIONS ABOUT THE"
30 PRINT "HOME TO BE ESTIMATED. ENTER YOUR ANSWERS BY TYPING"
40 PRINT "0 FOR NO, OR 1 FOR YES, AND THEN PRESS THE 'ENTER' KEY. "
55 PRINT " "
60 PRINT"**  REMEMBER  **   ZERO FOR NO  AND  1 FOR YES  **"
65 PRINT " "
70 PRINT "     THE PROGRAM ASSUMES THAT THE HOUSE HAS A KITCHEN,  A"
80 PRINT "LIVING ROOM, ONE FULL BATH AND TWO BEDROOMS PLUS THE OTHER"
90 PRINT "ROOMS IT WILL ASK YOU ABOUT.  IF IT IS SMALLER THAN THIS, "
100 PRINT "THE PROGRAM WILL BE UNABLE TO CALCULATE A COST FOR IT. "
115 PRINT " "
120 INPUT"IF YOU ARE READY TO GO, ENTER 1.  ARE YOU READY ";B$
125 IF B$ ="0" PRINT "END OF PROGRAM": END
130 CLS
140 PRINT "     1 = YES      0 = NO      "
145 PRINT " "
150 INPUT "DOES THE HOUSE HAVE A THIRD BEDROOM ";C
155 IF C = 0 GOTO 200
160 INPUT "A FOURTH BEDROOM ";D
```

Now take the ending location 833. The decimal equivalent of the low order byte 33 is 51(3*16+3). The high order byte 8 converts easily to 8 decimal.

```
190 POKE 60,0
200 POKE 61,8
210 POKE 62,51
220 POKE 63,8
```

Another way to compute the poke address is to take the decimal address and compute the low-order byte by using the MOD function with 256 as modulus and divide the decimal address by 256 to find the high order byte. So an alternative way is to POKE as follows:

```
170 POKE 60,2048 MOD 256
180 POKE 61, 2048/256
190 POKE 62, 2099 MOD 256
200 POKE 63, 2099/256
```

There are two ways of looking at the same thing. One is just as good as the other. If you look at the monitor, the first method is easier. If you count, the second way is easier.

The next thing we have to do is CALL the program that writes to the tape. First let's give ourselves a message to position our tape and put the program on hold until we're ready.

```
230 DIM A$(1)
240 PRINT "POSITION TAPE AND SET IN
    RECORD MODE"
250 INPUT "HIT RETURN WHEN READY",A$
260 CALL-307
270 PRINT "WRITE COMPLETED"
280 END
```

Now that wasn't too bad, was it?

The final thing we have to do now is write the program that will use the data created in program 1.

We'll do a simple program to read and print the data we read in. The most important thing to remember is that the create-data program and the use-data program must start out with the same dimension statement.

```
100 DIM A(10),B(10)
```

Now we POKE the read addresses we already computed when setting up the write program.

```
110 POKE 60,0
120 POKE 61,8
130 POKE 62,51
140 POKE 63,8
```

```
165 IF D = 0 GOTO 200
170 INPUT "A FIFTH BEDROOM ";E
175 IF E = 0 GOTO 200
180 INPUT "A SIXTH BEDROOM ";F
185 IF F = 0 GOTO 200
200 CLS
201 INPUT "DOES THE HOUSE HAVE A 2ND FULL (3-FIXTURE) BATH ";G
202 IF G = 0 GOTO 220
210 INPUT "A THIRD FULL BATH ";H
217 PRINT " "
220 INPUT "DOES THE HOUSE HAVE A HALF (2-FIXTURE) BATH ";T
222 IF T = 0 GOTO 226
225 INPUT "DOES IT HAVE A SECOND 2-FIXTURE BATH ";U
226 CLS
227 INPUT"IS THERE SPACE IN THE KITCHEN FOR A DINETTE ";V
228 PRINT " "
229 INPUT"DOES THE HOUSE HAVE A DINING ROOM ";L
230 PRINT " "
232 INPUT "DOES IT HAVE A FAMILY ROOM ";M
235 PRINT " "
240 INPUT "A STUDY ";K
250 PRINT " "
255 INPUT "DOES IT HAVE A DEN ";J
257 PRINT " "
260 INPUT "A RECREATION OR RUMPUS ROOM ";N
262 CLS
265 PRINT " "
268 INPUT"IS THERE AN ATTIC (BIG ENOUGH TO STAND UP IN ) ";BB
269 IF BB = 0 GOTO 275
270 INPUT "IS THE ATTIC FINISHED ";P
275 PRINT " "
276 INPUT "IS THERE A BASEMENT ";CC
277 IF CC = 0 GOTO 280
279 INPUT "IS THE BASEMENT FINISHED ";O
280 PRINT " "
282 INPUT "IS THERE A MAIN-FLOOR UTILITY ROOM ";S
283 PRINT " "
285 INPUT "IS THERE AN ATTACHED GARAGE ";AA
290 IF AA = 0 GOTO 300
295 INPUT "IS THE ATTACHED GARAGE A TWO-CAR ";Q
300 CLS
305 INPUT "IS THE HOUSE CENTRALLY AIR-CONDITIONED ";R
307 PRINT " "
345 INPUT "IS THERE A DECK ";W
350 INPUT "IS THERE AN ENCLOSED PORCH ";X
352 PRINT " "
355 INPUT "IS THERE A FIREPLACE ";Y
360 IF Y = 0 GOTO 367
365 INPUT "IS THERE A SECOND FIREPLACE ";Z
367 PRINT " "
369 INPUT "IS THERE ANY BRICK OR STONE-WORK ON THE EXTERIOR ";A
370 DD = (5+A+C+D+E+F+G+H+J+K+L+M+N+R)+.5*(O+P+Q+S+T+U+V+W+X+Y+Z+AA+BB+CC)
    CLS
372 PRINT "THE AREA OF THE COUNTRY WHERE THE HOUSE IS TO BE CONSTRUCTED IS
    ALSO A FACTOR IN COST ESTIMATION.":PRINT" "
373 INPUT "WILL THIS HOUSE BE BUILT IN THE MID-WEST";OO
374 IF OO = 0 GOTO 376
375 LL = .96 : GOSUB 405
376 INPUT "WILL IT BE BUILT SOUTH OF THE MASON-DIXON LINE ";EE
377 IF EE = 0 GOTO 382
378 INPUT "IN EITHER LOUISIANA OR KENTUCKY ";EE
379 IF EE = 0 GOTO 381
380 LL=.93 : GOSUB 405
381 LL = .83 : GOSUB 405
382 INPUT "IN THE GREAT SOUTHWEST ";GG
384 IF GG = 0 GOTO 389
385 INPUT "IN CALIFORNIA ";GG
386 IF GG = 0 GOTO 388
387 LL = 1.11 : GOSUB 405
388 LL = .95 : GOSUB 405
389 INPUT "IN THE NORTHWESTERN U S ";II
390 IF II = 0 GOTO 392
391 LL = .91 : GOSUB 405
392 INPUT "IN THE NEW ENGLAND STATES ";JJ
393 IF JJ = 0 GOTO 397
395 LL = .96 : GOSUB 405
397 LL = .93
405 MM=DD*LL :CLS
407 PRINT "NEXT, IT IS NECESSARY TO RATE THE DESIGN"
410 PRINT "AND CONSTRUCTION ON A SCALE OF 1 TO 4."
412 PRINT " "
415 PRINT "GRADE 1: A SIMPLE HOUSE BUILT FROM STOCK PLANS."
422 PRINT " "
425 PRINT "GRADE 2: BUILT FROM STOCK OR DESIGNER PLANS"
430 PRINT "WITH AVERAGE OR BETTER MATERIALS."
432 PRINT " "
435 PRINT "GRADE 3: CUSTOM BUILT FROM DESIGNER PLANS"
```

Then we set up to read in the arrays.

```
150 DIM A$(1)
160 PRINT "POSITION TAPE AN
    PLAY MODE"
170 INPUT "WHEN READY START TAPE AND
    HIT RETURN", A$
180 CALL-259
190 PRINT "READ COMPLETED"
```

Now we'll clear the screen and print the data, adding a delay so you can read the message in 190.

```
200 FOR J=1 TO 300: NEXT J
210 CALL-936
220 PRINT "B     A"
230 PRINT
240 FOR J = 1 TO 10
250 PRINT B(J);
260 PRINT "  "; A(J)
270 NEXT J
280 END
```

There you have it. A few points to remember: to compute the variable length be sure to add one byte for each letter in your variable name and the number of bytes includes subscript 0. For string variables each variable has one byte for each cell plus a termination byte (set to 1E) which marks the end of the string.

To review, then, for numeric variables the total number of bytes is:

L	letters in a name
+1	display byte
+2	address of next variable
+(N+1) * 2	two bytes for each cell (Dimension N plus 1).

For strings:

L	letters in name
+1	display byte
+2	address of next variable
+(N+1)	one byte for each cell (dimension N plus one)
+1	termination byte

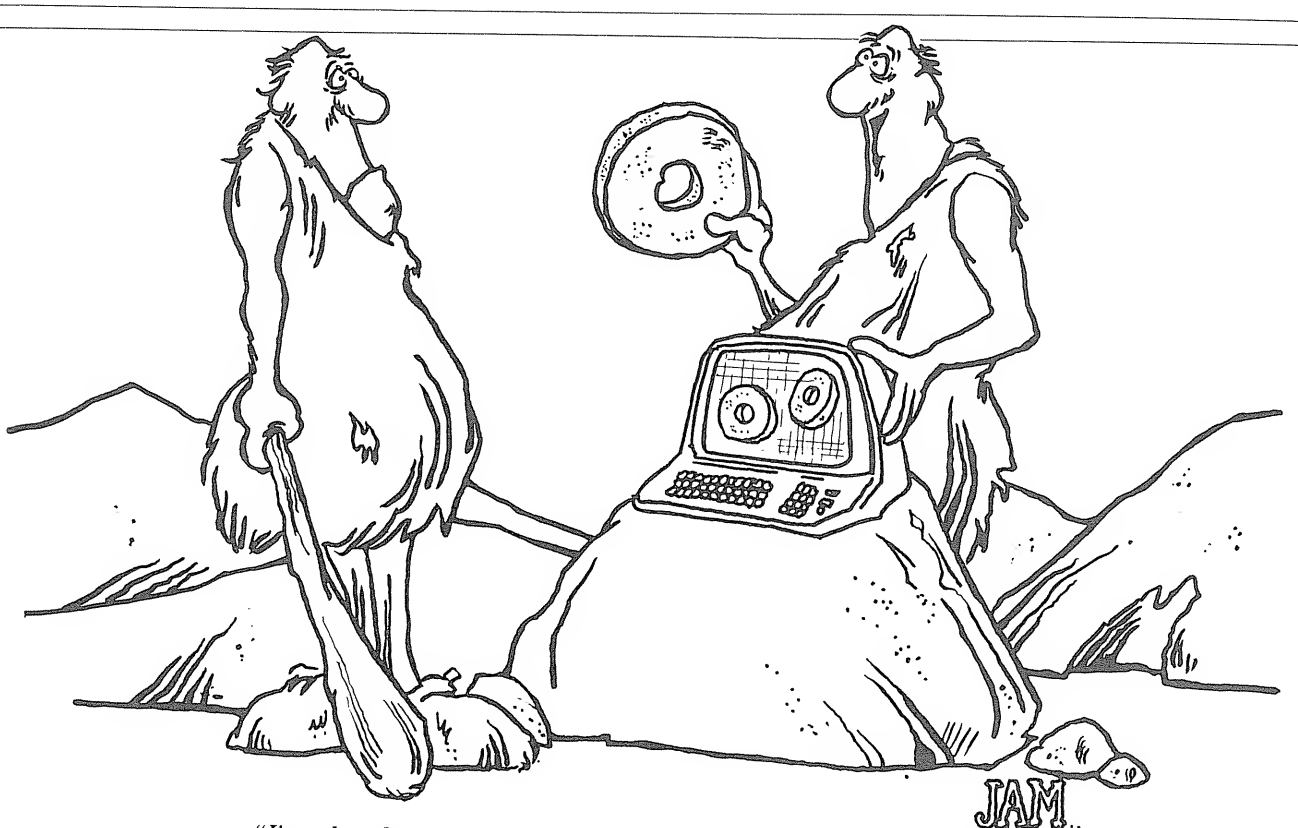
Also remember that both programs must start with the same dimension statements and have the same read and write addresses.

I think you'll find these routines useful. Now you can build your own Integer Basic programs to read and write data. □

```
440 PRINT "WITH MANY EXTRA FEATURES."
442 PRINT " "
445 PRINT "GRADE 4: ONE OF A KIND.  BUILT FOR AN"
450 PRINT "INDIVIDUAL UNDER AN ARCHITECT'S SUPERVISION."
452 PRINT " "
460 INPUT "PLEASE SPECIFY THE GRADE: 1,2,3 OR 4 ";FF
465 IF FF = 1 GOSUB 520
470 IF FF = 2 GOSUB 540
480 IF FF = 3 GOSUB 560
485 IF FF = 4 GOSUB 580
520 CLS
522 NN = MM * 3970
524 PRINT " "
525 PRINT "USING STOCK PLANS AND ECONOMY MATERIALS, WE "
527 PRINT "ESTIMATE THAT CONSTRUCTION COSTS FOR THIS HOME "
529 PRINT "WOULD TOTAL" USING "$$###,###.##";NN :GOSUB 600
540 CLS
542 NN = MM * 5024
545 PRINT "USING STOCK OR SIMPLE DESIGNER PLANS AND AVERAGE OR"
547 PRINT "BETTER MATERIALS AND WORKMANSHIP, WE ESTIMATE THAT"
551 PRINT "THE COST TO CONSTRUCT THE HOUSE YOU HAVE DESCRIBED"
554 PRINT "WOULD TOTAL" USING "$$####,###.##";NN :GOSUB 600
555 END
560 CLS : PRINT " "
562 NN = MM * 6164
565 PRINT "WE CALCULATE THAT THE CONSTRUCTION COST OF THIS"
567 PRINT "DESIGNER HOME (LOT & LANDSCAPING NOT INCLUDED)"
569 PRINT "WOULD BE" USING "$$####,###.##";NN
570 GOSUB 600
580 CLS: PRINT :PRINT"WE ESTIMATE THAT THE CONSTRUCTION COST OF THIS"
582 PRINT "UNIQUE HOME (LOT & LANDSCAPIING NOT INCLUDED)"
584 NN = MM * 7356
589 PRINT "WOULD BE" USING "$$####,###.##";NN :GOSUB 600
600 PRINT " "
602 PRINT "USING A CONSERVATIVE 6 % INFLATION FACTOR, THE"
606 PRINT "COST TO BUILD THIS HOUSE IN :"
610 PRINT "                              1981 WOULD BE"
      USING "$$####,###.##";1.06*NN
615 PRINT "                              1982 WOULD BE"
      USING "$$####,###.##";1.12*NN
620 PRINT "                              1983 WOULD BE"
      USING "$$####,###.##";1.19*NN
625 PRINT "                              1984 WOULD BE"
      USING "$$####,###.##";1.26*NN
630 PRINT "                              1985 WOULD BE"
      USING "$$####,###.##";1.34*NN
635 PRINT " "
640 PRINT " "
650 REM. AUTHOR:  J ESSIG, BOX 366, WEBSTER CITY, IA 50595
999 END
```

```
    ***  HOME CONSTRUCTION COST ESTIMATOR   ***

    THIS PROGRAM WILL ASK YOU QUESTIONS ABOUT THE
HOME TO BE ESTIMATED.  ENTER YOUR ANSWERS BY TYPING
0 FOR NO, OR 1 FOR YES, AND THEN PRESS THE 'ENTER' KEY.

**  REMEMBER  **   ZERO FOR NO  AND  1 FOR YES  **

    THE PROGRAM ASSUMES THAT THE HOUSE HAS A KITCHEN,  A
LIVING ROOM, ONE FULL BATH AND TWO BEDROOMS PLUS THE OTHER
ROOMS IT WILL ASK YOU ABOUT.  IF IT IS SMALLER THAN THIS,
THE PROGRAM WILL BE UNABLE TO CALCULATE A COST FOR IT.

IF YOU ARE READY TO GO, ENTER 1.  ARE YOU READY ? 1_


    1 = YES      0 = NO

DOES THE HOUSE HAVE A THIRD BEDROOM ? 1
A FOURTH BEDROOM ? 1
A FIFTH BEDROOM ? 1
A SIXTH BEDROOM ? 0_


DOES THE HOUSE HAVE A 2ND FULL (3-FIXTURE) BATH ? 1
A THIRD FULL BATH ? 0

DOES THE HOUSE HAVE A HALF (2-FIXTURE) BATH ? 1
DOES IT HAVE A SECOND 2-FIXTURE BATH ? 0_


IS THERE SPACE IN THE KITCHEN FOR A DINETTE ? 0

DOES THE HOUSE HAVE A DINING ROOM ? 1
```

```
DOES IT HAVE A FAMILY ROOM ? 0

A STUDY ? 1

DOES IT HAVE A DEN ? 0

A RECREATION OR RUMPUS ROOM ? 0_


   IS THERE AN ATTIC (BIG ENOUGH TO STAND UP IN ) ? 0

   IS THERE A BASEMENT ? 1
   IS THE BASEMENT FINISHED ? 1

   IS THERE A MAIN-FLOOR UTILITY ROOM ? 1

   IS THERE AN ATTACHED GARAGE ? 1
   IS THE ATTACHED GARAGE A TWO-CAR ? 1_


IS THE HOUSE CENTRALLY AIR-CONDITIONED ? 0

IS THERE A DECK ? 1
IS THERE AN ENCLOSED PORCH ? 0

IS THERE A FIREPLACE ? 1
IS THERE A SECOND FIREPLACE ? 1

IS THERE ANY BRICK OR STONE-WORK ON THE EXTERIOR ? 0_


THE AREA OF THE COUNTRY WHERE THE HOUSE IS TO BE

    CONSTRUCTED IS ALSO A FACTOR IN COST ESTIMATION.

WILL THIS HOUSE BE BUILT IN THE MID-WEST? 0
WILL IT BE BUILT SOUTH OF THE MASON-DIXON LINE ? 0
IN THE GREAT SOUTHWEST ? 0
IN THE NORTHWESTERN U S ? 0
IN THE NEW ENGLAND STATES ? 0_


NEXT, IT IS NECESSARY TO RATE THE DESIGN
AND CONSTRUCTION ON A SCALE OF 1 TO 4.

GRADE 1: A SIMPLE HOUSE BUILT FROM STOCK PLANS.

GRADE 2: BUILT FROM STOCK OR DESIGNER PLANS
WITH AVERAGE OR BETTER MATERIALS.

GRADE 3: CUSTOM BUILT FROM DESIGNER PLANS
WITH MANY EXTRA FEATURES.

GRADE 4: ONE OF A KIND.  BUILT FOR AN
INDIVIDUAL UNDER AN ARCHITECT'S SUPERVISION.

PLEASE SPECIFY THE GRADE: 1,2,3 OR 4 ? 3_


WE CALCULATE THAT THE CONSTRUCTION COST OF THIS
DESIGNER HOME (LOT & LANDSCAPING NOT INCLUDED)
WOULD BE  $88,854.10

USING A CONSERVATIVE 6 % INFLATION FACTOR, THE
COST TO BUILD THIS HOUSE IN :


1981 WOULD BE  $94,185.30
1982 WOULD BE  $99,516.60
1983 WOULD BE $105,736.00
1984 WOULD BE $111,956.00
1985 WOULD BE $119,064.00
```



"I've already programmed it on the CAD/CAM so theoretically it should roll."

Retracing Your Past

GENE: Retracing Your Past Through Genealogy

John W. Garson

People seem to be taking a new interest in their relatives, and not necessarily just those living. The TV program "Roots" helped us see what a fascinating thing it can be to try to trace back our family trees. But it isn't all that easy to keep track of a large genealogy. If you try to draw the tree on a giant piece of cardboard you run the risk of running out of room. So why not dust off that personal computer you have in the closet and use it to store data on your genealogy? Of course, you might need a little software to help do the job, so read on to find out more about GENE. GENE is written in TRS-80 Level II BASIC, but I have tried my best to use only those commands and statements that are common to most BASICs.

Session with GENE

To give you an idea of what GENE can do, let's imagine we have just loaded the program into a TRS-80 and typed RUN. After the introduction GENE asks:

```
DO YOU WANT TO LOAD A TAPE?
```

If we had previously run the program and had a tape with data on it to load, the answer would be YES, but since this is our first experience with GENE, we'll reply with NO (or N for short).

Since we don't have a tape, GENE takes out a sample family tree used for demonstration purposes. This tree contains names such as ME, DAD, MOM, AUNT, etc., instead of names of real people. When GENE is finished setting up this tree, it will let us see some more instructions if we desire. After that, GENE starts in on the main part of the program, and the screen looks like this:

```
MOTHER = MOM  FATHER = DAD
I SEE ME      :
```

GENE is now waiting for us to type a command after the colon. It is showing us where it is looking in the family tree. Now let's "take a walk" in this tree by typing FATHER (or just F for short). Then the screen will look like this:

```
MOTHER = DAD'S MOM
FATHER = DAD'S PA
I SEE DAD     :
```

What happened? GENE moved up the tree to the father of the person that it was looking at before (namely ME), so now it is looking at DAD and telling us the names of DAD's parents. If we type FATHER again, we get to look even further back in the tree:

```
MOTHER = GREATGRANDMA
FATHER = GREATGRAMPS
I SEE GREATGRANDMA  :
```

It looks as though there isn't any more information on the parents of great-grandma, so this line of the tree has come to an end. Let's go back down to ME, and trace up a different branch of the tree. What we do now is type SEE (or S), and the screen will look like this:

```
MOTHER = UNKNOWN
FATHER = UNKNOWN
I SEE GREATGRANDPA
TO GET OUT OF THIS HIT RETURN
:SEE
WHO WOULD YOU LIKE GENE
TO SEE?
```

Now we just type ME, and GENE gets us back to our starting point:

```
MOTHER = MOM  FATHER = DAD
I SEE ME      :
```

The message about hitting return points out a handy feature of GENE. Nobody is perfect! We're always going to make mistakes during a command, so we'll need a way to get out of any messes. Simply typing return at any point in a command will cancel the input and allow typing a new command at the colon on the right of the screen.

Making Lists

Now let's do something different. We type MOTHER to get up to MOM, and then type LIST (or just L). The screen displays:

```
MOTHER = MOM'S MOM
FATHER = MOM'S DAD
I SEE MOM     :
I CAN LIST ALL CHILDREN OR
SIBLINGS OF MOM
TO GET OUT OF THIS HIT RETURN
:LIST
WHAT RELATIONSHIP
DO YOU WANT?
```

Well, that's handy! I can now check up on something I always forget, the name of my mother's sister. All I have to do is type SIBLINGS (or just S), and I get a list of all mom's siblings:

```
SIBLINGS OF MOM
AUNT
```

How silly of me to forget! My mother's sister's name is AUNT. Now I know the name, so I can use SEE to get GENE to look at her, and trace back up her part of the tree.

But right now, we're going to do something else. Let's see what happens if we ask GENE for a list of my children. After typing LIST and CHILDREN, here is what we get:

```
CHILDREN OF MOM
ME
```

John W. Garson, University of Notre Dame, Notre Dame, IN 46556.

```
BRO
SIS
```

Oops, I wanted a list of my children, not mom's. I forgot that GENE was still looking at MOM. We have to type SEE,ME and then LIST CHILDREN. GENE types back:

```
CHILDREN OF ME
```

A blank list! Looks like I don't have any kids. But that is wrong; I happen to have a son. It looks as though I need to introduce him to GENE. To do that I type ADD (or just A) and the screen displays:

```
MOTHER = MOM FATHER = DAD
I SEE ME          :ADD
    I'M READY TO ADD DATA
        ON A NEW RELATIVE
TO GET OUT OF THIS HIT RETURN
NAME?
```

GENE is asking for the name of the person we're adding, so I type MYSON the next question is:

```
SEX?
```

and I answer MALE. GENE asks:

```
FATHER?
```

and I type ME. Finally GENE asks:

```
MOTHER?
```

and I type MYWIFE. GENE promptly responds:

```
I DON'T KNOW THAT PERSON
```

Looks like we haven't introduced GENE to MYWIFE either, but GENE goes ahead with the data received and shows us how things stand:

```
MOTHER = UNKNOWN
FATHER = ME
I SEE MYSON
```

You can see GENE does not know who MYSON's mother is. We can fix this up by adding MYWIFE to the program. We will have to use the ADD command to introduce MYWIFE. After that the SEE command is used to get back to MYSON and then type CHANGE (or C) to fix up his data:

```
MOTHER = UNKNOWN
FATHER = ME
I SEE MYSON       :CHANGE
READY TO CHANGE DATA ON MYSON
TO GET OUT OF THIS HIT RETURN
WHAT DATA DO I CHANGE?
```

We want to change the data on MYSON's mother, so we type MOTHER (or M). GENE responds with:

```
NEW MOTHER'S NAME?
```

MYWIFE is entered. GENE now knows all about my wife. A second later the screen changes to show that things have been fixed up:

```
MOTHER = MYWIFE
FATHER = ME
I SEE MYSON          :
```

Listing Descendants

Here is another of GENE's features. We type FATHER twice to get up to DAD, and then type DESCENDANTS (or just D). This gives us a list of all the descendants of my DAD.

```
DESCENDANTS OF DAD
GENERATION 1
CHILDREN OF DAD
ME
BRO
SIS
GENERATION 2
CHILDREN OF ME
MYSON
```

Sure enough, GENE knows enough to count MYSON as one of my DAD's descendants.

The END command terminates GENE. GENE then asks if you want to store the data on tape. After typing YES (or just Y) GENE requests that you load a new tape into the cassette recorder. We type return, the cassette is recorded, and GENE stops after printing a polite BYE! IT IS VERY IMPORTANT TO LOAD A NEW TAPE WHEN YOU ARE STORING DATA. If you forgot to do so, you may write over your copy of GENE. To avoid this, make sure the cassette you store GENE on has the little tabs punched out to prevent recording over it.

Entering Your Own Tree

There are two ways you can change the program so it works with your own family tree. Things are quite easy if you intend to store your data on tape. All you need do is type NEW (or N) to clear off the demonstration data. Then use the ADD command to put in new data on your relatives. You should put in your oldest relatives first so that GENE will already be introduced to fathers and mothers of people you add later on. Once you have built up a part of your family tree this way, you can store it on tape and simply load the tape anytime you use GENE again. You can then throw away lines 1500-1650 and 50-70, in case you want to save space. (I like to keep these lines, though, so I can show off how GENE works to people who don't happen to know anything about my relatives.)

One problem with the TRS-80 is that loading data from tape is a bit slow. It may be easier for you to simply replace the demonstration data in lines 2000 to 5000 with your own. You will have to add your data in by editing the program. The format for the data statement is:

```
DATA "name", "sex", M, F
```

The numbers of M and F should "point" to the data on the mother and father of a person. For example, the statement:

```
DATA "JAMES W. GARSON,"
"MALE," 6, 7
```

records data for me, and it says the data for my mother is in the 6th data statement and data on my father is in the 7th data statement. It is a good idea to number the DATA statements 2001, 2002, 2003, etc., so you can easily see which statement is the 1st, 2nd, 3rd and so on in your list of data. It is important to put DATA "UNKNOWN," "?," 1, 1 as your first data statement, and make sure that the last line (line 5000) reads DATA "", "", 1, 1.

GENE is really only a beginning. There are probably a number of features you want which I haven't thought about, or haven't had the time to put in. A program which did what everybody wants would probably be too large for a microcomputer, so it is better to let people who use GENE modify it to fit their needs. Here are a few guideposts to help you on your way.

Let's begin by explaining how to give GENE the ability to store other kinds of information such as birthdays, names of spouses, addresses, occupations or other notable facts. To store information on birthdays, for example, you need to set up a new array B$, and dimension it:

```
27 DIM B$(100)
```

Then you will need to add "B$(J)" at lines 76, 360 and 1520 to make sure the information stored in B$ will be read from the tape, stored on tape and read from the data in the program. For example, after the modification line 76 should read:

```
76 FOR J = 1 TO P: INPUT#-1,
N$(J), S$(J), M(J), F(J), B$(J):
NEXT J
```

You will also need to change the DATA items in lines 2000-5000 so they include data on birthdays. For example, line 2002 should now read:

```
2002 DATA "ME," "M," 5, 8
"JULY 26 1943."
```

If you are storing your data on tape, you will only need to change the first data statement this way, since the other data statement will be ignored.

You will also need to modify the ADD command to request information on birthdays, just add:

```
225 PRINT "BIRTHDAY";:
GOSUB 1000: B$(P + 1) = R$
```

The CHANGE command also needs a revision:

```
645 IF R$ = "B" THEN PRINT
"NEW BIRTHDAY";:
 GOSUB 1000: B$(K) = R$:
 GO TO 100
```

and the print statement at 658 needs an addition:

```
658 PRINT "ANSWER, NAME,
SEX, FATHER, MOTHER OR
BIRTHDAY":    GO TO 620
```

Another desired feature is to get GENE to print out data on birthdays, sex, or whatever else you stored. That is easy enough. For example, to get a command that prints the birthday of the person GENE is looking at, just add:

```
194 IF R$ = "B" THEN PRINT
"BDAY OF"; N$(W); " IS
"; B$(W): GOTO 1000!!
```

Another part of the program that needs development is the LIST command. You may want to include relationships other than siblings and children, such as brothers, sisters, aunts and uncles. The article, "How to Make a BASIC Tree" (Creative Computing, Nov. 1979) will explain the techniques in detail. Also, examine how the siblings and children feature is programmed in lines 580-599.

A Bit of Philosophy

I'm sure you can think of a lot of other things to add to GENE, but before you go off and experiment with it, I'd like to say a few things about the basic philosophy that went into the design of the program. There are two things I am proud of.

First, you are never stuck in a command. You can always get out by simply hitting return. Second, at any point in the program typing "?" will get you a helpful message telling you what your options are. This means that GENE has to have a number of subroutines to handle interaction with the user, but it is well worth it.

```
5 REM ************************************************
10 REM * GENE: A GENEALOGY PROGRAM
11 REM * (C) 1979 BY JIM GARSON
12 REM * DEPT OF PHILOSOPHY, U. OF NOTRE DAME
14 REM * NOTRE DAME, IND 46556
16 REM ************************************************
17 REM * IMPORTANT VARIABLES
18 REM * W   NUMBER OF PERSON GENE IS LOOKING AT
19 REM * P   NUMBER OF PEOPLE YOU HAVE DATA ON
20 REM * N$  ARRAY WITH NAMES IN IT
21 REM * S$  ARRAY WITH SEXES IN IT
22 REM * M   ARRAY WITH NUMBERS OF MOTHERS IN IT
23 REM * F   ARRAY WITH NUMBERS OF FATHERS IN IT
24 REM ************************************
25 CLEAR(2000)
28 DIM N$(100),S$(100), F(100),M(100)
29 DIM S(10),S1(10)
40 CLS
42 PRINT @400,"HI! I'M GENE!"
43 PRINT TAB(10);"YOUR FRIENDLY GENEALOGY PROGRAM"
44 PRINT:PRINT:PRINT"ANYTIME YOU DON'T KNOW WHAT TO DO"
45 PRINT "JUST TYPE QUESTION MARK (?) FOR HELP"
46 PRINT
50 PRINT "DO YOU WANT TO LOAD A TAPE";
60 R$="":INPUTR$:R$=LEFT$(R$,1)
65 IF R$="Y" THEN 72
66 IF R$="N" THEN 90
67 PRINT "TYPE YES OR NO"
68 PRINT "IF IN DOUBT TYPE NO":GOTO 50
70 PRINT "TYPE YES OR NO"
71 REM * LINES 72-78 LOAD A TAPE
72 PRINT "GET CASSETTE IN RECORDER AND PUSH RECORD BUTTON"
73  PRINT "HIT RETURN WHEN READY"
74  INPUT A$
75  INPUT#-1, P, W
76 FOR J=1 TO P:INPUT#-1, N$(J), S$(J),F(J),M(J):NEXTJ
78 GOTO 100
90 GOSUB 1500  REM * LOADS SAMPLE DATA
95 CLS
100 GOSUB 1200: REM *PRINT HEADING
110 PRINT @560, ":";
120 GOSUB1100  REM * PUT FIRST CHARACTER OF RESPONSE INTO R$
130 CLS : GOSUB 1200 : REM * PRINT HEADING
135 IF R$="E" THEN 300 : REM * END
140 IF R$="S" THEN 400 : REM * SEE
145 IF R$="A" THEN 200 : REM * ADD
150 IF R$="D" THEN 700 : REM * DESCENDENTS
155 IF R$="C" THEN 600 : REM * CHANGE
160 IF R$="L" THEN 500 : REM * LIST
170 IF R$="M" THEN W=M(W):GOTO 100 : REM * MOTHER
180 IF R$="F" THEN W=F(W):GOTO 100 : REM * FATHER
185 IF R$="N" THEN:P=1:PRINT@400,"READY FOR NEW DATA":GOTO 100
190 IF R$="?" THEN GOSUB910:GOTO 100: REM * SEE COMMANDS
195 PRINT @400, "WHAT??? TYPE ? FOR HELP":GOTO 100
200 REM * ADD INFO ON A NEW RELATIVE
205 PRINT @ 192,
207 REM * SUBROUTINE 1000 GETS RESPONSE AND PUTS IN R$
208 PRINT "I'M READY TO ADD DATA ON A NEW RELATIVE"
209 PRINT "IF YOU DON'T WANT TO DO THAT, HIT RETURN."
210 PRINT "NAME"; :GOSUB 1000:N$(P+1)=R$
220 PRINT "SEX";:GOSUB 1000:S$(P+1)=R$
230 GOSUB 1300 :REM * GETS NAMES OF MOTHER FATHER, UPDATES M, F
240 P=P+1 : REM * P NOW POINTS TO  FIRST EMPTY SPOT IN ARRAYS
250 W=P:GOTO 100
300 REM * END CHECK TO SEE IF THEY WANT TO STORE INFO ON TAPE
302 PRINT@ 528, "DO YOU WANT TO STORE WHAT YOU HAVE ON TAPE";
310 GOSUB 1100 : REM * PUTS FIRST CHARACTER TYPED IN R$
320 IF R$="N" THEN 390
325 IF R$="Y" THEN 330
327 PRINT "ANSWER YES OR NO"
328 PRINT "IF YOU AREN'T SURE, TYPE NO"
329 GOTO 302
330 PRINT "GET TAPE LOADED AND PRESS RECORD BUTTONS"
340 PRINT "TYPE RETURN WHEN READY"
```

One of the basic principles of personal programming should be to train the computer to follow some simple rules of conversational etiquette. These rules include letting people change the topic when they want, as well as helping them out when they get confused.

A final note: you may run out of memory if you put a lot of data into GENE. Not to worry. Just increase the values in the CLEAR and DIMENSION statements in lines 25 and 26. □

```
345 INPUT R$
350 PRINT@-1, P, W
360 FOR J=1 TO P:PRINT@-1,N$(J),S$(J),F(J),M(J):NEXT J
390 CLS:PRINT@256,"IF YOU WANT TO START ME OVER WITH THE SAME"
392 PRINT "DATA, YOU CAN TYPE CONT AFTER READY"
394 PRINT@400,"BYE!":STOP
395 GOTO 100
400 REM * MAKES GENE LOOK AT RELATIVE NAMED BY THE USER
401 PRINT @256, "TO GET OUT OF THIS HIT RETURN"
402 PRINT "WHO WOULD YOU LIKE GENE TO SEE";
405 GOSUB 1000 : REM * GET INPUT CHECK FOR ? AND EMPTY RESPONSE
410 GOSUB 1400 :REM * FIND NUMBER FOR NAME IN R$, PUT IN J
420 IF J=1 THEN GOTO 100
430 W=J
440 GOTO 100
500 REM * LISTS ALL RELATIVES WITH A GIVEN RELATIONSHIP
502 REM * TO THE PERSON GENE SEES.
504 REM * YOU MAY WANT TO ADD MORE RELATIONSHIPS TO THIS.
506 REM * TO LEARN HOW SEE "HOW TO BUILD A BASIC TREE"
508 PRINT @192,
509 PRINT "I CAN LIST ALL CHILDREN OR SIBLINGS OF ";N$(W)
510 PRINT "TO GET OUT OF THIS HIT RETURN"
515 PRINT "WHAT RELATIONSHIP DO YOU WANT";
520 GOSUB 1100: REM * PUTS FIRST CHARACTER TYPED IN R$
530 IF R$="C" THEN PRINT "CHILDREN OF ";N$(W): GOTO 580
535 IF R$="S" THEN PRINT "SIBLINGS OF ";N$(W): GOTO 590
540 PRINT "ANSWER C FOR CHILDREN OR S FOR SIBLINGS":GOTO 510
580 FOR J=1 TO P: REM * PRINT CHILDREN OF W
582 IF W=M(J) OR W=F(J) THEN PRINT N$(J)
584 NEXT J
586 GOTO 100
590 FOR J=1 TO P: REM * PRINT SIBLINGS OF W
591 IF W=J THEN 594 .REM * DON'T PRINT NAME OF PERSON SEEN
592 IF (F(W)=F(J) AND F(J)<>1) OR (M(W)=M(J) AND M(J)<>1) THEN  PRINT N$(J)
594 NEXT J
599 GOTO 100
600 REM * CHANGES DATA ON PERSON GENE SEES.
615 PRINT@256, "READY TO CHANGE DATA ON ";N$(W)
616 PRINT "TO GET OUT OF THIS HIT RETURN"
618 PRINT "WHAT DATA DO I CHANGE";
620 GOSUB 1100: REM * PUTS FIRST CHARACTER TYPED IN R$
630 IF R$="N"THEN PRINT"NEW NAME"; :GOSUB1000:N$(K)=R$:GOTO 100
640 IF R$="S" THEN PRINT"NEW SEX"; :GOSUB1000:S$(K)=R$:GOTO 100
650 IF R$="M" THEN GOTO 670
655 IF R$="F" THEN.GOTO 680
658 PRINT "ANSWER NAME, SEX, FATHER OR MOTHER":GOTO 620
670 INPUT "NEW MOTHER'S NAME"; R$: GOSUB 1400
672 M(W)=J: GOTO 100
680 INPUT "NEW FATHER'S NAME"; R$: GOSUB 1400
682 F(W)=J: GOTO 100
700 REM * PRINT DESCENDENTS
705 G=0
707 PRINT @ 192,
708 IF W=1 THEN PRINT "PERSON UNKNOWN":GOTO 100
710 PRINT "DESCENDENTS OF ";N$(W)
711 C=1:S(C)=W
712 G=G+1
715 PRINT "GENERATION "; G
720 F1=0
725 FOR K=1 TO C
730 PRINT "CHILDREN OF ";N$(S(K))
732 C1=0
735 FOR J=1 TO P
740 IF F(J)=S(K) OR M(J)=S(K) THEN C1=C1+1:S1(C1)=J:PRINT N$(J)
745 NEXT J
750 IF C1>0 THEN F1=1
755 NEXT K
760 IF F1=0 THEN INPUT"HIT RETURN TO CONTINUE";R$:GOTO 100
765 C=C1:FOR K=1 TO C1:S(K)=S1(K):NEXT K: GOTO 712
910 CLS:
912 PRINT "COMMANDS CAN ALL BE SHORTENED TO THEIR FIRST LETTER"
914 PRINT "  END      GIVES YOU THE OPPORTUNITY TO STORE DATA"
916 PRINT "           ON TAPE, AND THEN STOPS"
918 PRINT "  SEE      GENE WILL LOOK AT A PERSON YOU NAME"
920 PRINT "  MOTHER   GENE WILL LOOK AT THE MOTHER OF THE"
930 PRINT "           PERSON SEEN"
940 PRINT "  FATHER   GENE WILL LOOK AT FATHER OF PERSON SEEN"
945 PRINT "  ADD      ADD DATA ON A NEW PERSON"
950 PRINT "  CHANGE   CHANGE DATA ON PERSON SEEN BY GENE"
955 PRINT "  LIST     LISTS ALL PEOPLE WITH A GIVEN RELATION-"
960 PRINT "           SHIP TO PERSON SEEN BY GENE"
970 PRINT " DESC    LISTS ALL DESCENDENTS "
980 PRINT "           OF PERSON GENE SEES"
982 PRINT " NEW      CLEARS OFF OLD TREE SO YOU CAN START OVER"
990 PRINT " ?        TO SEE THIS LIST OR FOR HELP"
995 INPUT "HIT RETURN TO CONTINUE"; R$
999 CLS:RETURN
1000 REM*PUTS USER'S RESPONSE IN R$ GOES TO 100 IF RETURN HIT
1010 R$="":INPUTR$:IF R$="" THEN  100
1020 RETURN
1100 REM * PUTS FIRST CHARACTER IF USER'S RESPONSE IN R$
1110 REM * IF RETURN WAS HIT GOES TO 100
1120 R$="":INPUT R$:R$=LEFT$(R$,1):IF R$="" THEN 100
1130 RETURN
1200  REM * PRINTS HEADING
1210 PRINT @0,
1220 PRINT @0,TAB(4);"MOTHER = ";N$(M(W)),TAB(38);"FATHER = ";N$(F(W));"
1230 PRINT TAB(20);  "I SEE ";N$(W)
1240 RETURN
1300 REM * THIS SUBROUTINE TAKES MOTHERS AND FATHER NAMES
1310 REM * AND STORES THEIR CORRESPONDING NUMBERS AT
1320 REM * LOCATION P+1 IN ARRAYS M AND F
1330  PRINT "MOTHER";:GOSUB1000: REM * PUT RESPONSE IN R$
1340 GOSUB 1400 : REM * GET NUMBER FOR NAME IN R$
1350 M(P+1)=J
1360 PRINT "FATHER";:GOSUB1000: REM * PUT RESPONSE IN R$
1370 GOSUB 1400
1380 F(P+1)=J
1390 RETURN
1400 REM * TAKES NAME IN R$ AND PUTS ITS NUMBER IN J
1410 FOR J=1 TO P
1420 IF N$(J)=R$ THEN 1460
1430 NEXT J
1440 PRINT "I DON'T KNOW THAT PERSON": J=1
1460 RETURN
1500 REM * THIS LOADS DEMONSTRATION DATA
1502 REM * TO BUILD YOUR OWN FAMILY TREE
1504 REM * USE THE NEW COMMAND TO ERASE DEMONSTRATION DATA.
1506 REM * THEN USE THE ADD COMMAND TO PUT IN YOUR TREE
1508 REM * THIS WORKS IF YOU ARE STORING DATA ON TAPE
1510 REM * IF YOU WANT TO STORE DATA IN THE PROGRAM
1512 REM * YOU WILL NEED TO EDIT LINES 2000-5000
1514 REM * EACH DATA STATEMENT HAS THE FORM
1516 REM * DATA "NAME","SEX",M,F
1518 REM * M IS THE NUMBER OF THE DATA STATEMENT
1520 REM * WHERE DATA THE MOTHER OF THIS PERSON IS STORED
1522 REM * AND F IS THE PLACE WHERE DATA ON THE FATHER IS.
1540 W=2 : REM * W IS PERSON GENE IS LOOKING AT
1550 P=0
1555 P=P+1
1560 READ N$(P),S$(P),M(P),F(P)
1570 IF N$(P)="" THEN 1600
1580 GOTO 1555
1600 CLS.PRINT "I JUST LOADED A SAMPLE FAMILY TREE SO THAT
1610 PRINT "YOU CAN GET AN IDEA OF WHAT I CAN DO."
1620 PRINT "YOU WILL SEE NAMES LIKE ME, MOM, DAD,AUNT, ETC."
1630 PRINT "INSTEAD OF NAMES OF ACTUAL PEOPLE."
1640 PRINT "I WILL START BY PRINTING THE NAME OF THE PERSON
1650 PRINT "I AM LOOKING AT IN THE TREE, AND THE NAMES OF THIS"
1660 PRINT "PERSON'S PARENTS.  I SUGGEST YOU TRY OUT THE"
1670 PRINT "MOTHER AND FATHER COMMANDS FIRST."
1680 PRINT "PLAY AROUND 'TIL YOU ARE USED TO ALL THE COMMANDS"
1690 PRINT "TO PUT IN YOUR OWN FAMILY TREE USE THE NEW COMMAND"
1695 PRINT "TO CLEAR OFF THE DEMONSTRATION TREE."
1700 PRINT "THEN USE THE ADD COMMAND TO PUT IN DATA ON"
1710 PRINT "YOUR REALTIVES."
1720 PRINT "TO SEE A LIST OF THE COMMANDS OR FOR HELP TYPE ?"
1730 R$="":INPUTR$:R$=LEFT$(R$,1)
1735 IF R$="?" THEN GOSUB910
1740 CLS:RETURN
2001 DATA "UNKNOWN", "?", 1,1
2002 DATA "ME", "M",5,8
2003 DATA "BRO", "M", 5,8
2004 DATA "SIS", "F", 5,8
2005 DATA "MOM", "F",6,7
2006 DATA "MOM'S MOM", "F",1,1
2007 DATA "MOM'S DAD", "M",1,1
2008 DATA "DAD", "M",9,10
2009 DATA "DAD'S MOM", "F",1,1
2010 DATA "DAD'S PA","M",12,11
2011 DATA "GREATGRAMPS", "M",1,1
2012 DATA "GREATGRANDMA", "F",1,1
2013 DATA "AUNT", "F",6,7
2014 DATA "COUSIN", "M", 13,1
5000 DATA "","",1,1
```

Examine Your Family Tree

How to Make a Basic Tree

James W. Garson

A while back a **Creative Computing** reader asked how to store information on his/her genealogy with a microcomputer. This article describes some of the principles of tree construction in Basic and explains how to apply them in constructing a genealogy program. We'll be making reference to the GENE program which appeared in last month's Creative Computing ("GENE: Retracing Your Past Through Genealogy"-Feb 80). It is written in TRS-80 Level II Basic, but can easily be adapted to other versions of Basic which allow string arrays.

Initial Housekeeping

The main problem in working out a genealogy program is to find a way to represent your family tree so it can be modified easily. Here is how. First, set up a string array N$ which will contain your relative's names:

```
DIM N$(100)
```

We will put "UNKNOWN" in N$(1), and then store names of 99 relatives in N$(2), N$(3), . . . , N$(100). It really won't matter what order you put the names in.

Now we set up two more arrays of the same size as N$:

```
DIM M(100), F(100)
```

These arrays contain numbers that point to the places in N$ where the names of the mother and father of a given relative can be found. For example, suppose N$(4) contains:

```
MORTIMER SNERT
```

Then M(4) should contain a number (say it's 21) which tells us where the name of Mort's mom can be found in N$. To put it another way, N$(21), that is, N$(M(4)) will contain the name of Mort's mom. Similarly, N$(F(4)) will contain the name of Mort's father. If it turns out that nobody knows who Mort's father was, we just let F(4) = 1, so that F(4) "points to" what is in N$(1), namely "UNKNOWN" (You should set M(1) = 1 and F(1) = 1, since mothers and fathers of unknown people are unknown too.)

James W. Garson, University of Notre Dame, Notre Dame, IN 46556.

Working With Your Family Tree

Writing programs to change and to move around in your family tree is fairly easy. If you want to add new names, just write them in the first unoccupied spots in N$. To keep track of where this is, you need to keep a number in a variable (call it P) that points to the last occupied spot in N$. To add a new relative named Samantha Snoggert, we just add 1 to P, and store her name in N$(P). We will also need to update M and F so that they point to the places where the names of her parents are found. Normally we won't remember the numbers where these names are found, but it is easy enough to write a program that takes, for example, the name of Samantha's mother, and then puts the right number in M(P). Here is a program that does just that:

```
10  INPUT "MOTHER'S NAME"; M$
20  FOR J=1 TO P
30  IF N$(J) = M$ THEN GOTO 60
40  NEXT J
50  J=1         (Name wasn't found so
                 person is unknown)
60  M(P)=J      (J is the number for
                 name in M$)
```

Here are a couple of programs that are handy for "moving around" in a family tree of this kind. To print the name of Mort's mother, for example, we just write

```
100  X=M(4)
       (Remember, Mort's number is
       4, so X is his mother's number).
110  PRINT N$(X)
```

To get the name of his grandmother, just change line 100 to:

```
100  X=M(M(4))
       (X is the mother of the mother
       of Mort.)
```

Getting the names of Mort's children takes some searching:

```
500  FOR J=1 TO P
       (P is the number of people we
       have stored.)
510  IF 4=F(J)  THEN PRINT N$(J)
       (If Mort is J's father, print J's
       name.)
520  NEXT J
```

This program is a bit inefficient since it must look at all the names to find those of Mort's kids. The alternative is to set up more pointers for children. Since a relative can have many children, this pointer must be a two dimensional array. For example, C(4, 1) would contain the number for Mort's first child, C(4, 2) the number for his second child, and so on. You will have to decide whether you want to add "forward" pointers like C along with M and F. Doing so would speed up getting data on children and children's children, etc., but the TRS-80 is fast enough that you don't notice the time it takes to do the search. Given that, it isn't worth the bother of setting up the new array C and writing a subroutine to update it every time you add new data.

To get names of Mort's grandchildren, we just change line 510 of the previous program:

```
510  IF 4=F(F(J)) THEN PRINT N$(J)
       (If Mort is father of the father
       of J, then print J's name.)
```

Now this program will only print out names of Mort's grandchildren on the male line; it does not print out names of the children of his daughters. To get all his grandchildren we must write:

```
510  IF 4=F(F(J)) OR 4=F(M(J))
       THEN PRINT N$(J)
```

Here the name for J is printed if Mort is the father of the father of J or the father of the mother of J.

To get a list of Mort's siblings (his brothers and sisters), we just change line 510 to

```
510  IF F(4)=F(J) THEN PRINT N$(J)
       (If Mort's father and J's father
       are the same, print J's name.
```

This program will print out the names of all people who have the same father as Mort (including Mort himself). That means it will list any of Mort's half sisters or half brothers by his father. To get only Mort's full brothers and sisters we write:

```
510  IF F(4)=F(J) AND M(4)=M(J)
       THEN PRINT N $(J)
```

which prints the name of only those people who have the same father and mother as Mort. To get a list that includes Mort's half brothers and sisters, just put OR instead of AND in 510. If we want a list of Mort's aunts and uncles on his father's side, we simply change line 510 again:

```
510  IF F(F(4))=F(J)
       THEN PRINT N$(J)
       (If father of father of Mort is
father of J, then print.)
```

Further Expansion

Now let's get sex into the picture. To get names of brothers but not sisters, or uncles, but not aunts, we need to be able to tell who is male and who is female. It is easy to store sex data in a new array S$. Then S$(4) can contain "MALE," since that is what Mort is, and S$(21) will contain "FEMALE," since Mort's mother is female.(You may want to savememory by using a numerical array S, and code numbers for the two sexes.) Now that this information is available, we can change our program so that it prints a list of Mort's brothers (by his father):

```
510  IF F(4)=F(J) AND S$(J)="MALE"
       THEN PRINT N$(J)
```

This prints J's name only if J has the same father as Mort and is male. We won't go into all the ramifications on half-uncles and 25th cousins, but it should be clear that if you can state clearly which of Mort's relatives you want, you can write the correct condition into 510 to get a list of their names. In the program GENE (see lines 500-599) you can only get the siblings and children of a given relative listed, but once you understand how to rewrite the condition in lines like 510, you can easily expand this part of the program to handle any other relationships you like.

Getting an entire list of Mort's descendents is a bit complicated. For example, we would have to write

```
510  IF 4=F(F(F(J))) OR 4=F(F(M(J)))
       OR 4=F(M(F(J))) OR
       4=F(M(M(J))) . . .
```

just to get a full list of Mort's great-grandchildren. Maybe that explains why genealogists only trace the male line. It is just too complicated to deal with the whole picture. You could write a program that puts the numbers of Mort's children in an array, and then Mort's mother's descendents on her female line for 5 generations:

```
700  FOR G=1 TO 5
710  PRINT "GENERATION"; G
720  FOR J=1 TO P
730  Z=J
740  FOR L=1 TO G
       (Lines 740-760
       set Z=M (...M(J))
            G times
750  Z=M(Z)
       So Z is the mother of
       (G times) J.)
760  NEXT L
770  IF 21=Z THEN PRINT N$(J)
       (Mort's mom's number is 21;
       Z is the mother of (G times) J.)
780  NEXT J
790  NEXT G
```

You male chauvinists can replace line 750 with Z=F(Z).

adds the numbers of their children, and so on, and then prints the names for all the numbers stored. Writing this program isn't too difficult. However, since most genealogists don't keep information on the female lines, you may never need to do this. Since I disagree about which side of the family is most important, here is a simple program that prints out the names of all

Conclusion

There is probably a lot more information you want to store on your relatives than just their names and sexes. It is easy to set up new arrays to store as many different kinds of information as you like. If you want to keep track of birthdays, set up an array D$, and store Mort's birthday in D$(4). GENE doesn't have any extra arrays like this, but you can add them. The advantage of setting up our tree the way we did is so you can easily change your mind about what kinds of data to store without messing up the information that represents your family tree. This may help you solve the inevitable problem of inadequate memory. Any reasonably large genealogy will gobble up 16K. But when you are hunting around for relatives, the only arrays that really have to reside in memory are M, F, N$ and S$. (You can even do without N$, if you can remember relatives by number.) All other information can be on tape, or even in notebooks. During a search you can keep numbers of relatives you want data on in a list, which can later be used to print out or look up specific information. □

Making Budgeting and Planning Easier

Expense Management Package

Bethany Prendergast

Have you ever wished that you could make your little microcomputer act like one of the "Big Boys" or do you just hanker for that one little application that could make even the biggest skeptic go "Ohhhh!" and "Ahhhh!"? Well, for you TRS-80 owners, this might be just the program you are looking for.

You can run your computer in a large scale manner, convince the skeptics in your house and at the same time produce usable outputs for both amateurs and small businessmen alike. Can you give me: 1) the amounts in dollars and cents, 2) the mean, the standard deviation and future projections, and 3) either a video screen graph or a hard copy graph of your electric bill for the past two years? I can and, in addition, I can give it to you (or IRS) in many more areas than my electric bills — all in a matter of seconds.

This article will deal with 3 programs: EXPENSES, TRANSFER/EXP, and GRAPH/EXP. The programs have been written for the TRS-80 Level II, 32K (or 48K), with a disk and line printer. The line printer used is the IDS-225, but the programs will run with any 80 column printer after only slight modification (such as removal of expanded print commands).

First, some general remarks that apply to the package as a whole. You will note some program steps in the listings that appear to be needless and serve no purpose. I assure you they are there for a reason. In my attempts to go "Bigtime" I experienced many unexplainable errors — not explainable from the TRS-80 books, that is. Just as I learned from experimenting that the best cure for keybounce was to insert Tunerlube Jelly into the key after removing it, so I had to start experimenting to find out the causes of my erratic machine behavior. What was my trouble? You name it and I was having it! Incomplete reads/writes, all kinds of disk errors, along with much unexplained garbage. After much wailing and gnashing of teeth, I determined the following to be necessary for the operation of my system:

1. removal of the power packs from the expansion interface, placing them to the side;
2. aluminum foil between the expansion interface and the video screen;
3. a fan on the system when working;
4. false loops to slow down the disk during rapid read/writes, as reflected in the program listings;
5. loading of blanks to the disk buffer between read/writes.

Now, I'm pleased to report that my TRS-80 is operating right at 100%, 4 to 6 hours per day, all of the time.

The programs are long. For those of you who will take the time to contact me first, I'll try to arrange to make a disk/tape copy provided you will furnish the tape/disk and reimburse me for the postage and the gas to get to the post office. I will accept prepaid calls before 10 PM in the evening to answer any question that you may have. If you want a copy made, you'd best call first because I am not set up for any large scale reproduction. For those of you who live in a different time zone, I would appreciate your checking the time here in Florida before calling because I put my recorder on at 10 PM.

Expenses Program

EXPENSES is a general purpose TRS-80 program that stores date and amount information into 20 different accounts. The information is stored to the accounts using the random file technique. The information can be recalled for the purpose of 1) display of the entries in each account; 2) graphing of the entries in each account to the video screen and/or line printer; and 3) display of statistical data for each account to date.

I have named the accounts for my own use. However, the program keeps track of the accounts by using variables E1-E20, so you may change the name of the accounts to any name(s) you wish by merely changing the PRINT and the LPRINT statements starting at line number 2000. The number of accounts may also be altered to more or less than 20 by altering the value of "Z" where appropriate. Altering the number of the accounts would be difficult at best and I do not recommend trying this until you feel certain you understand the program thoroughly. Instead of altering the numbers of accounts, I suggest that you carry the 20 and use only the ones you need, keeping the unused ones as dummy accounts with a zero beginning entry and carrying them that way until you want to use them.

The printer I have is an IDS-225, and I rate it at A+. The LPRINT CHR$(n) statements in the program control the print size on my printer. You may delete them from your program and/or substitute the codes for your printer. All printing is upper case.

Entries are made to each account in response to screen prompts. When entering information for the date, you may exercise some degree of individual preference. The only restriction is that the date entry cannot exceed 9 characters nor be shorter than 8 characters. I have elected to use a three letter month followed by a comma and then the year (e.g., JUN,1979 or SEP,1979). After you play with the program for awhile, you might prefer to use the actual day's date. Be sure to stick to one format, whatever you decide. Option #3, for example, searches the account for a match which is why consistency is important.

When the program is first run, it will display 5 options and ask you to

Bethany Prendergast, 10129 Leisure Lane North, Deerwood, Jacksonville, Florida 32216.

select one. On your first run, select option #1 and make at least one date and amount entry for each of the 20 accounts. If you have no entry to an account or do not choose to use an account, enter the current month, year and zero for the amount. The program will accept and calculate zero amounts for the purposes of graphing and computing statistics. There is one exception to this. Because the TRANSFER/EXP program initializes the accounts to a zero as the first entry, the statistics routine checks first to see if the first entry is a zero. If it finds a zero as the first entry it passes it by and selects the second entry to start computation. After you have used option #1 to initialize everything you probably won't need to use it again. The rest of the options are self-explanatory. If you have any questions as to which one to use, fall back on option #2. It calls out the entire file and tells you what's on it. It also should be used to call an account for graphing or statistical purposes.

The EXPENSES program doesn't begin to pay off for you until you have accumulated some small amounts of data. Its value lies in two areas; to store and retrieve and to display trends. For example, I can tell (after accumulating 6 months data) just how much of a gasoline savings I realized when I switched over to a Diesel Olds.

In addition to being able to recall in seconds, individual expenditures (such as the last time I paid my dentist) I can give you a pictorial analysis, in the form of a graph, on my video screen in a flash. You will be surprised at just what this program will show up about things that you never had the means to examine before — especially if you are in business. Once again, it is necessary to enter a fair amount of history initially, or wait a few months for it to accumulate.

I generally batch enter my entries from a small book I carry around with me (and which serves as an audit trail if a disk were to 'bomb'). Every three or four days, I enter all my expenditures onto a disk labeled my "DAILY DISK." At the completion of each entry session, I get a printout of the account totals to date. Accordingly, if my system were to go down, I have both my original entry book and a copy of the totals from the last input session to permit me to recover. At the end of the month, I transfer my account totals (using the TRANSFER/EXP program) to a "MONTHLY DISK" and reinitialize the "DAILY DISK" for a new month. I keep a backup disk for the "MONTHLY DISK," incidentally, so I feel that I have a sufficient audit trail in the event of some calamity.

The EXPENSES program is composed of fairly complex routines. An in-depth understanding of the program would require a good foundation in Basic programming to include a thorough working knowledge in the areas of string manipulation and the use of random files to pack data. Unfortunately, it is hard to find a good elementary treatise on these subsections of Basic. It should be possible for you to study the program in conjunction with your TRS-80 book, however, and, taking a command at a time, slowly piece together just what is happening. To help you, the following is a line-by-line breakdown of the program:

LINES #90 - 590:

This portion of the program is the graph routine to the video screen. It graphs the data that has been stored under A(X) by the expense parts of the program.

The data is graphed on an X-Y axis basis and is scaled down (if necessary) to fit the screen. Some of the accuracy is lost by taking the integer function of the Y value in line 200, and by the scaling, but its purpose is to merely show a relationship of one entry to another and display trends.

LINES #600-950:

This portion of the program is the statistical routine. A prompt appears on the screen after the graphing routine or if the graph option was bypassed asking if you want statistical data. The routine gives the mean, the standard deviation and the coefficient of correlation for the account you are dealing with at the time. Future projections can be requested at this time.

LINES #960-1040:

This is the beginning of the program and gives a general explanation of what the program does along with the different options that are available for running. The month and year are carried under Y$(X) and the amounts are carried under A(X).

LINES #1100 - 1140:

This portion of the program gives the five options permitting the entry and/or retrieval of data. Option #1 is to be used for the initializing of accounts the first time through. Options #2, #3, #4 and #5 are self-explanatory. Option #5 permits the batching of your entries to all accounts and permits you to enter large amounts of data in a relatively short period of time. This option allows you the luxury of not having to post your entries everyday; instead, you may post to the accounts as seldom as once a week.

LINES #1170 - 1260:

Option #1 begins at #1170 and jumps to #2140 - 2260 to list the 20 different accounts by name and number (E1-E20). Variables E1-E20 identify each account and are then carried as F$ throughout the remainder of the program. The entry of data for option #1 takes place in #1200-1240, with a print of the data to the video in #1430. The saving of the data takes place at lines #1700 - 1740.

LINE #1280:

Start of option #2.

LINES #1310 - 1340:

Check to make sure that file exists on the disk.

LINES #1350 - 1520:

Opens file F$ and brings the file into RAM. File is then printed using 0$ format concluding with a total.

LINES #1530 - 1800:

Permit corrections or additions to files. Note that all files are stored using the random file designators and are packed 16 date and amount entries to a sector. Each file manipulation must unpack and/or pack before moving on to a save or other handling of the data. A loop is set up to load each date and its corresponding amount into the buffer up to 16 times, dependent on the number of entries. (See lines #1700). You are cautioned to remember as you step through this that if you have less than 16 entries you must stop at your last entry. Correspondingly, your loop must take into account that there may be more than 16 entries. An alteration would have to be made at this point if your memory did not permit all data for an account to be brought into RAM. I did not provide for this true 'random handling' because I cannot see any single account having that many entries before I retire that disk.

LINES #1810 - 1840:

This routine returns the dollar amount entry for the given date.

LINE #1850:

Goto the print subroutine.

LINES #1860 - 1870:

Enter account identifier (E1-E20) and the date.

LINE #1940:

Jump to error routine if you are attempting a read from a file that has not been established yet.

LINES #1960 - 2060:

Returns file requested in line #1870 by T$.

LINES #2090 - 2120:

Prints file name, date and amount to line printer.

LINES #2130 - 2700:

Contain recurring subroutines to

print out file names.
LINES #2710 - 2780:
Prints current date heading on line printer.
LINES #2800 - 3150:
Brings in all files one at a time printing file totals to video and/or line printer.
LINES #3160 - 3170:
Beginning of option #5 for batch entries to all accounts.
LINES #3190 - 3210:
Input of data.
LINE #3220:
Total your entries for display on video.
LINE #3280:
Permits positing to any one (or all) accounts.
LINES #3320 - 3340:
Zero out Y$(X) and A(X) because total to that account has been saved under CO for amount and Q$ for date.
LINE 3440:
Prints the name of the account being handled.
LINES #3450 - 3480:
Checks to see if file is on disk.
LINES #3510 - 3720:
Bring in each file, add new information and write it back to the disk.
LINE #3730:
Check that all files have been counted.
LINES #3750 - 3800:
If last sector was full, write information to a new sector.
LINES #3820 - 3950:
Create a new file.

Transfer Expenses Program

TRANSFER/EXP is the second program of the EXPENSES PROGRAM PACKAGE. It presumes some knowledge of the EXPENSE program, at least to the extent of understanding what it does.

TRANSFER/EXP is the easiest of the three programs to understand in that it is, for the most part, operator independent. Once you start to run it, it sort of takes off by itself. Operator intervention is called for by video prompts, but is limited. You are given the option of specifying some hard copy printout only. The result of the operating time is devoted to changing disks when requested.

TRANSFER/EXP is a special purpose utility program. It is intended to deal only with those random files created by EXPENSES. The "DAILY DISK" is totaled by each individual account and the totals are retained in RAM. The "DAILY DISK" is then exchanged for the "MONTHLY DISK" and the totals are written to the monthly accumulation for each account. At the completion of the transfer of the totals, you are asked if you want to reinitialize the daily accounts. A "Yes" answer will cause the screen to prompt for the reinsertion of the "DAILY DISK" and the old account entries will be killed. The program goes on to make a new first entry for each of the 20 accounts of the month and the year with the insertion of a zero for the amount. Accordingly, each account now contains only one entry on the "DAILY DISK"; namely, month, year and $0.00.

As I stated previously, there is a backup in the event of a system failure with your daily accounts. Specifically, you have original documents and the hard copy printout after each entry session. Unfortunately, there is always the possibility that you could have a system catastrophy resulting in destruction of both the "DAILY DISK" and "MONTHLY DISK." I agree that such a thing is highly unlikely, but it is better to plan for the worst. This is why the first part of this program gives you a hard copy printout of each account on the daily disk. If you retain this printout and the original documents, you can reconstruct and/or build new disks with very little effort. I also go one step further and back up my monthly disk each month.

If you have studied the EXPENSES program in any detail, you will find that the same logic is carried forward into the routines for this program. The following is a summation of the program by line numbers:

LINE #150 - 190:
A blinking prompt routine using the INKEY$ statement.
LINES #320 - 490:
The daily files are opened and the entries totaled for each account. The account totals are carried under T(Z). If you asked for hard copy in line #80 it is printed out in #490.
LINES #540 - 1040:
#550 jumps back to another blink routine starting at line #230 and, upon completion, the totals are transferred to the "MONTHLY DISK." The new date entry is carried as Q$. Each file is opened and read. The new date, Q$, and the appropriate total under T(Z) is added on to the Y$(X) and A(X) columns. The new information is written out to the disk starting at E1 to E20. If you attempt a write to an account that has not been opened on the monthly disk, use option #1 of EXPENSES to open it. The packing and the unpacking of the file buffer is the same as with EXPENSES.
LINES #1110 - 1250:
Each account is killed off the "DAILY DISK." Line #1260 requests the new month and year to serve as the new first entry to each account as it is reinitialized.
LINES #1370 - 1420:
New entry is made as each account is reinitialized.
LINES #1420 - :
Program concludes.

Graph of Expenses Program

GRAPH/EXP is the last program of the EXPENSES PROGRAM PACKAGE. The GRAPH/EXP program is a special purpose program that will take information from the files, established by EXPENSES and TRANSFER/EXP, and graph that information to your line printer. The information is graphed to the line printer on the X-Y axis with the entries represented on the Y axis and the sequence of the entry shown on the X axis. Accordingly, the first entry is graphed to the 1st line of X, and so on. Some accuracy is sacrificed by virtue of converting the entries to integers and by scaling the entries down to not exceed 90 (the number of print columns).

Some of my students have called this program "tricky." I'm not sure that is an appropriate way to refer to any program, but in any event, I truly did not intend it to be so. I wrote the program because I found the video screen graph to be difficult to study. Also, the video graph is, at best, temporary and I find that I often want to examine where my accounts are headed in the convenience of my armchair. All I am looking for is a hard copy printout that will allow me to carefully compare my entries. If I spot something that looks out of line, I'll go to the account itself for a more accurate examination.

The crux of the program is in understanding the way I convert the numeric entries to strings and how I then locate the print symbol on that string. The whole print line is assembled in memory before it is printed. The technique used to accomplish this is really quite simple. For example, let's examine how I convert $15.00 to a string: 1) enter the number you want to convert as the upper limit on a loop, FOR X = 1 TO 15 2) Select a string variable that has been set to zero and add any character to it as it goes through the loop, A$ = A$ + "." 3) Print the string variable at completion and it will contain the number of characters equal to the original number.

```
10  A$ = ""
20  FOR X = 1 TO 15
30  A$ = A$ + "."
```

```
40  NEXT X
50  PRINT A$
```

Your video will now display:

. . . . . .

That's all there is to it! I actually have the strings display periods while the program runs so you can see something happening instead of watching a blank screen.

I have elected to use a 90 column print format. You may alter this program to run on any line printer by changing the loops to correspond to the number of print columns you want to use. The LPRINTCHR$(30) is peculiar to the IDS and may be deleted.

A breakdown of the program by line numbers is as follows:

LINE #80:

The number of lines in the X axis (the length).

LINES #90 - 100:

Enter the file to be read.

LINES #110 - 170:

Open the file and bring in the entries to RAM.

LINES #200 - 240:

Convert the entries to integers.

LINES #260 - 400:

If the entries are > 90 scale them down.

LINES #410 - 460:

Set up a loop that converts the numeric values into strings of periods stored under A$(I).

LINE #480:

Display the strings of periods on the video screen.

LINE #520:

Print the Y axis increments.

LINES #530 - 620:

Print the Y axis.

LINES #630 - 830:

Set up each line for a print and locate the print symbol on that line (#750). Repeat until all entries are finished.

LINES #880 - 970:

If X lines have been called for to extend the graph beyond the last entry graphed they are now printed. □

```
10 '-PROGRAM LISTING FOR CREATIVE COMPUTING
20 '-.....EXPENSES.....
30 '-PROGRAM BY BETHANY PRENDERGAST 10129 LEISURE LANE N.
40 '-DEERWOOD- JACKSONVILLE, FLORIDA 32216 ( 904-642-1902 )
50 '-WRITTEN FOR TRS80, LEVEL II, 32 OR 48 K WITH (1) DISK
60 '-GOOD LUCK....AUTHOR WILL ACCEPT PREPAID CALLS BETWEEN
70 '-7-10 IN THE EVENING IF YOU HAVE PROBLEMS!!!
80 '-PROGRAM BEGINS AT 960
90 '-GRAPH ROUTINE
100 CLS: GOTO960
110 GOTO120
120 CLS:GOSUB 320
130 REM-THIS IS A GRAPHING PROGRAM , IT WILL GRAPH ITEMS
140 REM-TAKEN FROM DATA STORED UNDER A(X)...
150 CLS
160 PRINT@ 66,"Y";
170 PRINT@ 1023,"X";
180 IF A(1) =0 THEN FF7=2 ELSE FF7=1
190 FOR F=FF7 TO Z
200 Y=A(F): Y= INT ((47-(Y/G))+.5)
210 FOR S%= 1 TO 47
220 SET(F,Y)
230 Y=Y+1: IF Y>47 THEN 250
240 NEXT S%
250 NEXT F
260 Y= 47
270 FOR X = 0 TO 127 : SET (X,Y) : NEXTX : X=0
280 FOR Y = 0 TO 47  : SET(X,Y) : NEXT Y
290 FOR WW=1 TO 1000: NEXT WW
300 GOTO 1780
310 '-GRAPH BY B. PRENDERGAST 11/1/78
320 G=0:GG=0
330 FOR S= 1 TO Z
340 IF A(S)<47 THEN G=1:GOTO560
350 IF A(S)<94 THEN G=2:GOTO560
360 IF A(S)<141 THEN G=3:GOTO560
370 IF A(S)<188 THEN G=4:GOTO560
380 IF A(S)<235 THEN G=5:GOTO560
390 IF A(S)<282 THEN G=6:GOTO560
400 IF A(S)<329 THEN G=7:GOTO560
410 IF A(S)<376 THEN G=8:GOTO560
420 IF A(S)<423 THEN G=9:GOTO560
430 IF A(S)<470 THEN G=10:GOTO560
440 IF A(S)<517 THEN G=11:GOTO560
450 IF A(S)<564 THEN G=12:GOTO560
460 IF A(S)<614 THEN G=13:GOTO560
470 IF A(S)<658 THEN G=14:GOTO560
480 IF A(S)<705 THEN G=15:GOTO560
490 IF A(S)<752 THEN G=16:GOTO560
500 IF A(S)<799 THEN G=17:GOTO560
510 IF A(S)<846 THEN G=18:GOTO560
520 IF A(S)<893 THEN G=19:GOTO560
530 IF A(S)<940 THEN G=20:GOTO560
540 IF A(S)<987 THEN G=21:GOTO560
550 IF A(S)=>987 THEN PRINT"Y POINTS TOO LARGE TO BE GRAPHED"
560 IF G=>GG THEN GG=G                                    :GOTO1780
```

```
570 NEXT S
580 G=GG
590 RETURN
600 '-STATISTICS ROUTINE
610 R=0:M=0:P=0:O$="$$####.##.##"
620 IF A(1)=0 THEN II7=2 ELSE II7=1
630 FOR I= II7 TO N
640 D=A(I):P=P+D:M=M+D[2
650 NEXT I
660 IF II7=2 THEN N=N-1
670 R=P/N
680 V=(M-N*R[2)/N
690 SD=SQR(V)
700 CLS:PRINT"MEAN = ";R
710 PRINT"STANDARD DEVIATION = ";SD
720 J=0:K=0:L=0:M=0:R2=0
730 FOR I= 1 TO N
740 X=I:Y=A(I)
750 J=J+X:K=K+Y:L=L+X[2:M=M+Y[2
760 R2=R2+X*Y
770 NEXT I
780 B=(N*R2-K*J)/(N*L-J[2)
790 A=(K-B*J)/N
800 J=B*(R2-J*K/N)
810 M=M-K[2/N
820 K=M-J
830 R2=J/M
840 PRINT"COEFF.OF CORRELATION (1 IS PERFECT) = ";SQR(R2)
850 QQ$=""
860 PRINT:PRINT"DO YOU WISH TO PREDICT A FUTURE VALUE
    (YES/NO) ?"
870 INPUT QQ$
880 IF QQ$="" THEN 860
890 IF QQ$="NO" THEN 950
900 IF QQ$="YES" THEN CLS
910 INPUT"ENTER MONTH NO.(1,2,...X) YOU WANT ESTIMATE
    FOR";X
920 Y=A+B*X
930 PRINT"PROJECTION ESTIMATED AS";USING O$;Y
940 PRINT"ANOTHER ESTIMATE (YES/NO) ?": GOTO 870
950 CLS:PRINT"PROGRAM CONCLUDED":END
960 CLS:PRINT"THIS PROGRAM PERMITS THE MANIPULATION OF 120"
970 PRINT"MONTHS OF DATA TO/FROM RANDOM FILES. DATA MUST BE"
980 PRINT"ENTERED AS DATE ( SEE REMARKS LINES ) FOLLOWED BY"
990 PRINT"'AMOUNT'. THE AMOUNT IS MANIPULATED AS SINGLE
    PRECISION."
1000 PRINT"THIS PROGRAM WILL ALSO GIVE A VIDEO DISPLAY"
1010 PRINT"WITH A GRAPH AND WILL  PROVIDE STATISTICS ON DATA"
1020 PRINT"IF DESIRED. OPTION #2 FOR ADD/DELETE/CORRECT ITEMS.'
1030 PRINT"OPTION #3 ALLOWS ENTRY OF MONTH,YEAR AND PROGRAM THE
1040 PRINT"RETRIEVES CORRESPONDING AMOUNT."
1050 PRINT
1060 '-PROGRAM BY BETH PRENDERGAST 11/20/78
1070 '- Y$(X) = MONTH,YEAR
1080 '- A(X) = AMOUNT
1090 CLEAR(3500): DIM Y$(150): DIM A(150):DIMB(150):DIM T(150)
1100 PRINT "ENTER: 1 - TO OPEN FILE FOR FIRST TIME"
1110 PRINT "       2 - BRING OUT WHOLE FILE FOR ADD/CORRECT/
     LOOKSEE
1120 PRINT "       3 - RETRIEVE A MONTH'S ENTRIES FROM A FILE
1130 PRINT "       4 - PRINT TOTAL OF EACH FILE TO DATE
1140 PRINT "       5 - BATCH ENTER TO 1 OR MORE ACCOUNTS
1150 INPUT Q
1160 ON Q GOTO 1170 ,1280 ,1810 ,2710 ,3150
1170 CLS:GOSUB 2130
1180 INPUT"ENTER FILE I.D. (IN QUOTES) DATA IS TO BE SAVED
     UNDER";F$
1190 CLS
1200 INPUT"NO. OF ENTRIES (MT,YR & AMT = 1 ENTRY)";N
1210 FOR X=1 TO N:PRINT"ENTRY NO.";X
1220 LINEINPUT"ENTER MONTH,YEAR (YOU MAY USE COMMA) ?";Y$(X)
1230 INPUT"ENTER AMOUNT";A(X):CLS
1240 NEXT X
1250 Z=N
1260 '-MOVE TO PRINT ROUTINE
1270 GOTO1430
1280 CLS:GOSUB2130
1290 INPUT"ENTER FILE (IN QUOTES) TO BE READ";F$
1300 PRINT"FILE NAME IS  ";:GOSUB 2300
1310 ON ERROR GOTO 2280
```

```
1320 OPEN"I",1,F$
1330 CLOSE
1340 ON ERROR GOTO 0
1350 OPEN"R",1,F$:X=1
1360 FOR I=1 TO LOF(1):GET1,I:FOR E=0TO16
1370 FIELD1,(E*15)AS DUM$,11AS Y$,4AS A$
1380 Y$(X)=Y$: A(X)=CVS(A$)
1390 U$="              ":IF Y$(X)=U$ THEN 1410
1400 X=X+1:NEXT E: NEXT I
1410 CLOSE1 : N=X-1
1420 '-MOVE TO PRINT ROUTINE
1430 C=0:CLS:INPUT"PAUSE AFTER EACH ENTRY
     PRINTS(YES/NO)";QQ$
1440 CLS:O$="$$###,###.##"
1450 FOR X=1 TO N
1460 PRINT"ENTRY NO. ";X;":     ";Y$(X);"-";USING O$;A(X)
1470 C=C+A(X)
1480 IF QQ$="YES" INPUT"PRESS ENTER";Q:CLS
1490 IF QQ$="NO" THEN FOR Y=1 TO 600:NEXT Y
1500 NEXT X
1510 PRINT:PRINT"TOTAL TO DATE = ";USING O$;C
1520 Z=X-1:N=X-1
1530 PRINT"ANY (MORE) CORRECTIONS/ADDITIONS (YES/NO)?":QQ$
1540 INPUT QQ$: IF QQ$="NO" THEN 1640
1550 CLS:PRINT"LAST ENTRY NO. IS  ";X-1;" ON FILE NAMED ";
1560 QQ$="":GOSUB2300
1570 INPUT"ENTER RECORD NO. FOR CORRECTION/ADD";X
1580 PRINT"ADD/RE-ENTER RECORD NO.";X
1590 LINEINPUT"MONTH,YEAR (YOU MAY USE COMMA)"; Y$(X)
1600 INPUT"AMOUNT";A(X)
1610 IF X>N THEN N=X:Z=N
1620 PRINT"FILE WILL NOW PRINT AGAIN"
1630 INPUT"PRESS ENTER";Q: GOTO 1430
1640 CLS:PRINT"DATA IS FROM FILE  ";F$;" - ";
1650 GOSUB 2300
1660 QQ$="":INPUT"DO YOU WANT TO SAVE THIS DATA
     (YES/NO)";QQ$
1670 IF QQ$="YES" THEN 1700 ELSE 1760
1680 CLS:F$="":GOSUB2130
1690 INPUT"ENTER FILE NAME (IN QUOTES) FOR SAVE";F$
1700 OPEN"R",1,F$:X=1:AA=INT(N/17)+1:FOR I=1 TO AA
1710 FOR E=0 TO 16
1720 FIELD1,(E*15)AS DUM$,11AS Y$,4AS A$
1730 LSET Y$=Y$(X) : RSET A$=MKS$(A(X)): X=X+1
1740 NEXT E: PUT1,I: NEXT I: CLOSE 1
1750 CLS:PRINT"FILE HAS BEEN SAVED  AS  ";F$:QQ$=""
1760 INPUT"DO YOU WANT A  GRAPH (YES/NO)";QQ$
1770 IF QQ$="YES" THEN 110  ELSE QQ$=""
1780 INPUT"DO YOU WANT STATISTICS (YES/NO)";QQ$
1790 IF QQ$="YES" THEN 610  ELSE QQ$="":CLS
1800 CLS:PRINT"PROGRAM CONCLUDED":END
1810 C1=0:CLS:GOTO 1820
1820 PRINT"THIS ROUTINE GETS AM'T WHEN GIVEN THE DATE...."
1830 PRINT"ENTER THE DATE AS IT IS IN THE FILE..."
1840 FOR Y=1 TO 600:NEXT Y
1850 CLS:GOSUB 2130
1860 INPUT"ENTER FILE (IN QUOTES) TO BE READ";F$
1870 LINEINPUT"ENTER DATE (YOU MAY USE A COMMA)  ?";T$
1880 '-BB$=2 & CC$=3 BLANKS TO MAKE T$= 11
1890 B7=LEN(T$) : BB$="  ": CC$="   "
1900 IF B7<8 THEN PRINT"YOU HAVE NOT FOLLOWED INSTRUCTIONS":END
1910 IF B7>9 THEN PRINT"YOU HAVE NOT FOLLOWED INSTRUCTIONS":END
1920 IF B7 = 8 THEN T$=T$+CC$ : GOTO 1940
1930 IF B7 = 9 THEN T$=T$+BB$ : GOTO 1940
1940 ON ERROR GOTO 2280
1950 QQ$="":CLS:INPUT"PRINTOUT OF TOTAL (YES/NO) ";QQ$
1960 OPEN"I",1,F$:CLOSE
1970 OPEN"R",1,F$ : X=1
1980 CLS
1990 FOR I=1 TO LOF(1) : GET 1,I: FOR E= 0 TO 16
2000 FIELD 1,(E*15) AS DUM$,11 AS Y$,4 AS A$
2010 Y$(X)=Y$ : A(X) = CVS(A$)
2020 O$="$$###,###.##"
2030 IF Y$(X) = T$  THEN 2040 ELSE 2060
2040 PRINTY$(X);"-";USINGO$;A(X)
2050 C1=C1+A(X)
2060 X=X+1: NEXT E: NEXT I: CLOSE 1
2070 IF QQ$= "NO" THEN 2120
2080 LPRINTCHR$(30)
2090 LPRINT F$;" - ";:GOSUB2300
2100 LPRINT T$;" - ";USINGO$;C1
2110 LPRINT "FILE SEARCH COMPLETE":LPRINTCHR$(29)
2120 PRINT"FILE SEARCH COMPLETE":PRINT:PRINT"
     ROUTINE CONCLUDED":END
```

```
2130 GOTO2140
2140 PRINT"E1-HOUSE (T/A/S&W/M)","E13-MEDICAL"
2150 PRINT"E2-MOTELS",,"E14-CLUB"
2160 PRINT"E3-MARKET",,"E15-SCHOOL"
2170 PRINT"E4-ENTERTAIN",,"E16-TRASH"
2180 PRINT"E5-DRUGSTORE",,"E17-MISC"
2190 PRINT"E6-GAS",,"E18-TRIPS(TOTAL)"
2200 PRINT"E7-INSR",,"E19-PHONE"
2210 PRINT"E8-CAR",,"E20-ELEC"
2220 PRINT"E9-CLOTHES/D"
2230 PRINT"E10-CLOTHES/M"
2240 PRINT"E11-CLOTHES/B"
2250 PRINT"E12-RESTAURANTS"
2260 PRINT
2270 RETURN
2280 PRINT"YOU ARE ATTEMPTING TO READ FROM A NONEXISTENT FILE
2290 PRINT"PROGRAM CONCLUDED":END
2300 IF F$="E1" AND QQ$="YES" THEN LPRINTTAB(15) "HOUSE"
2310 IF F$="E1" THEN PRINT "HOUSE"
2320 IF F$="E2" AND QQ$="YES" THEN LPRINTTAB(15) "MOTELS"
2330 IF F$="E2" THEN PRINT "MOTELS"
2340 IF F$="E3"AND QQ$="YES" THEN LPRINTTAB(15)"SUPER MARKET"
2350 IF F$="E3" THEN PRINT "SUPER MARKET"
2360 IF F$="E4"AND QQ$="YES" THEN LPRINTTAB(15)"ENTERTAIN"
2370 IF F$="E4" THEN PRINT "ENTERTAIN"
2380 IF F$="E5"AND QQ$="YES" THEN LPRINTTAB(15)"DRUGSTORE"
2390 IF F$="E5" THEN PRINT "DRUGSTORE"
2400 IF F$="E6"AND QQ$="YES" THEN LPRINTTAB(15)"GAS"
2410 IF F$="E6" THEN PRINT "GAS"
2420 IF F$="E7"AND QQ$="YES" THEN LPRINTTAB(15)"INSUR."
2430 IF F$="E7" THEN PRINT "INSUR."
2440 IF F$="E8"AND QQ$="YES" THEN LPRINTTAB(15)"CAR"
2450 IF F$="E8" THEN PRINT "CAR"
2460 IF F$="E9"AND QQ$="YES" THEN LPRINTTAB(15)"CLOTHES/D"
2470 IF F$="E9" THEN PRINT "CLOTHES/D"
2480 IF F$="E10"AND QQ$="YES" THEN LPRINTTAB(15)"CLOTHES/M"
2490 IF F$="E10" THEN PRINT "CLOTHES/M"
2500 IF F$="E11"AND QQ$="YES" THEN LPRINTTAB(15)"CLOTHES/B"
2510 IF F$="E11" THEN PRINT "CLOTHES/B"
2520 IF F$="E12"AND QQ$="YES" THEN LPRINTTAB(15)"RESTAURANTS"
2530 IF F$="E12" THEN PRINT "RESTAURANTS"
2540 IF F$="E13"AND QQ$="YES" THEN LPRINTTAB(15)"MEDICAL"
2550 IF F$="E13" THEN PRINT "MEDICAL"
2560 IF F$="E14"AND QQ$="YES" THEN LPRINTTAB(15)"P.V.CLUB"
2570 IF F$="E14" THEN PRINT "P.V.CLUB"
2580 IF F$="E15"AND QQ$="YES" THEN LPRINTTAB(15)"SCHOOL"
2590 IF F$="E15" THEN PRINT "SCHOOL"
2600 IF F$="E16"AND QQ$="YES" THEN LPRINTTAB(15)"TRASH"
2610 IF F$="E16" THEN PRINT "TRASH"
2620 IF F$="E17"AND QQ$="YES" THEN LPRINTTAB(15)"MISC"
2630 IF F$="E17" THEN PRINT "MISC"
2640 IF F$="E18"AND QQ$="YES" THEN LPRINTTAB(15)"TRIPS(TOTAL)"
2650 IF F$="E18" THEN PRINT "TRIPS (TOTAL)"
2660 IF F$="E19"AND QQ$="YES" THEN LPRINTTAB(15)"PHONE"
2670 IF F$="E19" THEN PRINT "PHONE"
2680 IF F$="E20"AND QQ$="YES" THEN LPRINTTAB(15)"ELECTRIC"
2690 IF F$="E20" THEN PRINT "ELECTRIC""
2700 RETURN
2710 CLS
2720 INPUT"HARD COPY (YES/NO)";QQ$
2730 IF QQ$="" THEN 2720  ELSE 2740
2740 IF QQ$="YES" THEN 2750  ELSE 2790
2750 INPUT"DATE (XX/XX/XX)";DATE$
2760 LPRINTCHR$(01):LPRINTCHR$(30)
2770 LPRINTTAB(12)".........";DATE$;"........."
2780 LPRINT
2790 GOTO 2880
2800 REM- BLINK ROUTINE FOR DAILY
2810 CLS
2820 PRINT@400,"INSERT DISK TO BE TOTALED (ENTER)"
2830 FOR Y=1 TO 100:NEXT Y
2840 B$=INKEY$
2850 IF B$="" THEN 2860  ELSE 2870
2860 CLS: GOTO 2820
2870 RETURN
2880 T$="E":GOSUB 2800
2890 Z=1:SUM=0
2900 C$=STR$(Z)
2910 IF LEN(C$)=2 THEN N=1
2920 IF LEN(C$)=3 THEN N=2
2930 D$= RIGHT$(C$,N)
2940 F$=T$+D$:C1=0
2950 OPEN"R",1,F$:X=1
2960 FOR I=1 TO LOF(1): GET 1,I:FOR E=0 TO 16
```

```
2970 FIELD 1,(E*15) AS DUM$, 11 AS Y$, 4 AS A$
2980 U$="           "
2990 R$= Y$: IF R$ = U$ THEN CLOSE 1: GOTO3030
3000 Y$(X)=Y$:A(X)=CVS(A$)
3010 C1=C1+A(X)
3020 X=X+1: NEXT E: NEXT I: CLOSE 1
3030 T(Z)=C1: O$="$$###,##.##":SUM=SUM+T(Z)
3040 IF QQ$="YES" THEN 3050  ELSE 3060
3050 GOSUB2300 :LPRINTTAB(20)F$;"-";USINGO$;T(Z):
3060 PRINTF$;"-";USINGO$;T(Z)
3070 IF Z=20 THEN 3100
3080 Z=Z+1
3090 GOTO 2900
3100 IF QQ$="YES" THEN 3110  ELSE 3120
3110 LPRINT:LPRINTTAB(10)"GRAND TOTAL TO DATE IS";USINGO$;
     SUM
3120 PRINT:PRINT"GRAND TOTAL TO DATE IS  ";USINGO$;SUM
3130 LPRINTCHR$(02):LPRINTCHR$(28)
3140 END
3150 GOSUB 2130
3160 INPUT"ENTER FILE I.D.(IN QUOTES) DATA IS TO BE SAVED
     UNDER";F$
3170 CLS: PRINT"ENTER AMOUNTS(ONE AT A TIME) ENDING WITH
     0 FOR FILE  "; :GOSUB2300
3180 CO=0
3190 FOR X=1 TO 100
3200 INPUT A(X): IF A(X)=0 THEN 3220
3210 CO=CO+A(X):NEXT X
3220 PRINT"TOTAL IS: ";CO
3230 LL9=LEN (F$)
3240 IF LL9=3 THEN VV=2
3250 IF LL9=2 THEN VV=1
3260 Z=VAL(MID$(F$,2,VV))
3270 T(Z)=CO: QP$=""
3280 INPUT"DO YOU WANT TO MAKE ENTRIES FOR OTHER ACCOUNTS
     (YES/NO)";QP$
3290 IF QP$="" THEN 3280
3300 IF QP$="YES" THEN 3150  ELSE 3310
3310 LINE INPUT "ENTER MT./YR. FOR SAVE?";Q$
3320 FOR X= 1 TO 150
3330 A(X)=0:Y$(X)=""
3340 NEXT X
3350 CLS:PRINT"SAVING TOTALS"
3360 T$="E": Z=1
3370 IF T(Z)>0 THEN 3380  ELSE Z=Z+1:IFZ>20THEN END ELSE3370
3380 FOR Y= 1TO 50:NEXT Y:C$=STR$(Z)
3390 CLOSE 1
3400 IF LEN (C$)=2 THEN N=1
3410 IF LEN (C$)=3 THEN N=2
3420 D$= RIGHT$(C$,N)
3430 F$=T$+D$
3440 PRINTF$,
3450 '-CHECK TO SEE IF FILE IS ON THE DISK
3460 ON ERROR GOTO 3820
3470 OPEN"I",1,F$:CLOSE 1
3480 ON ERROR GOTO 0
3490 '-DD$=255 BLANKS TO CLEAR BUFFER
3500 DD$="
                                                        "
3510 OPEN"R",1,F$:FIELD 1,255 AS Z$
3520 LSET Z$=DD$:CLOSE 1: FOR Y=1 TO 50 :NEXT Y
3530 '-FILE EXISTS & MUST BE READ BEFORE WRITING TOTAL
3540 OPEN"R",1,F$:X=1:L=LOF(1)
3550 GET1,L:FOR E=0 TO 16
3560 FIELD 1,(E*15) AS DUM$,11 AS Y$,4 AS A$
3570 U$="           "
3580 '-SET FAKE VARIABLE TO SEE IF DATA FIELD IS EMPTY
3590 R$=Y$:IF R$=U$ THEN 3620
3600 Y$(X)=Y$:A(X)=CVS(A$)
3610 X=X+1:NEXT E: GOTO 3750
3620 Y$(X)=Q$: A(X)=T(Z)
3630 CLOSE:FOR Y=1 TO 50:NEXT Y:OPEN"R",1,F$
3640 FIELD 1,255 AS Z$:LSET Z$=DD$ :CLOSE:FOR Y=1 TO 10:NEXT
3650 OPEN"R",1,F$
3660 M=1
3670 FOR E=0 TO 16
3680 FIELD 1,(E*15) AS DUM$, 11 AS Y$, 4 AS A$
3690 IF Y$(M)="" THEN PUT 1,L:GOTO3720
3700 LSET Y$=Y$(M): RSET A$=MKS$(A(M))
3710 M=M+1:NEXT E: PUT 1,L
3720 CLOSE 1:FORX=1TO150:Y$(X)="":A(X)=0:NEXTX
3730 IF Z=20 THEN END
3740 Z=Z+1: GOTO 3370
3750 '-START A NEW SECTOR-LAST READ SHOWED SECTOR FULL-CLEAR
```

```
3760 CLOSE 1:FOR Y=1 TO 50: NEXT Y: OPEN"R",1,F$
3770 FIELD 1,255AS Z$:LSETZ$=DD$:CLOSE1:FOR Y=1 TO 10:
     NEXT Y
3780 OPEN"R",1,F$:FIELD 1,11 AS Y$,4 AS A$,240 AS DUM$
3790 LSET Y$=Q$: RSET A$=MKS$(T(Z))
3800 L=L+1: PUT 1,L
3810 GOTO 3720
3820 '-FILE MUST BE CREATED & TOTAL WRITTEN
3830 '-DD$=255 BLANKS TO CLEAR BUFFER
3840 CLOSE1:OPEN"R",1,F$
3850 FIELD 1,255 AS Z$
3860 DD$="


3870 LSET Z$=DD$
3880 CLOSE 1
3890 OPEN"R",1,F$
3900 FIELD 1,11 AS Y$, 4 AS A$, 240 AS DUM$
3910 LSET Y$=Q$: RSET A$=MKS$(T(Z))
3920 PUT 1,1:CLOSE 1
3930 IF Z=20 THEN END
3940 Z=Z+1
3950 RESUME 3370
```

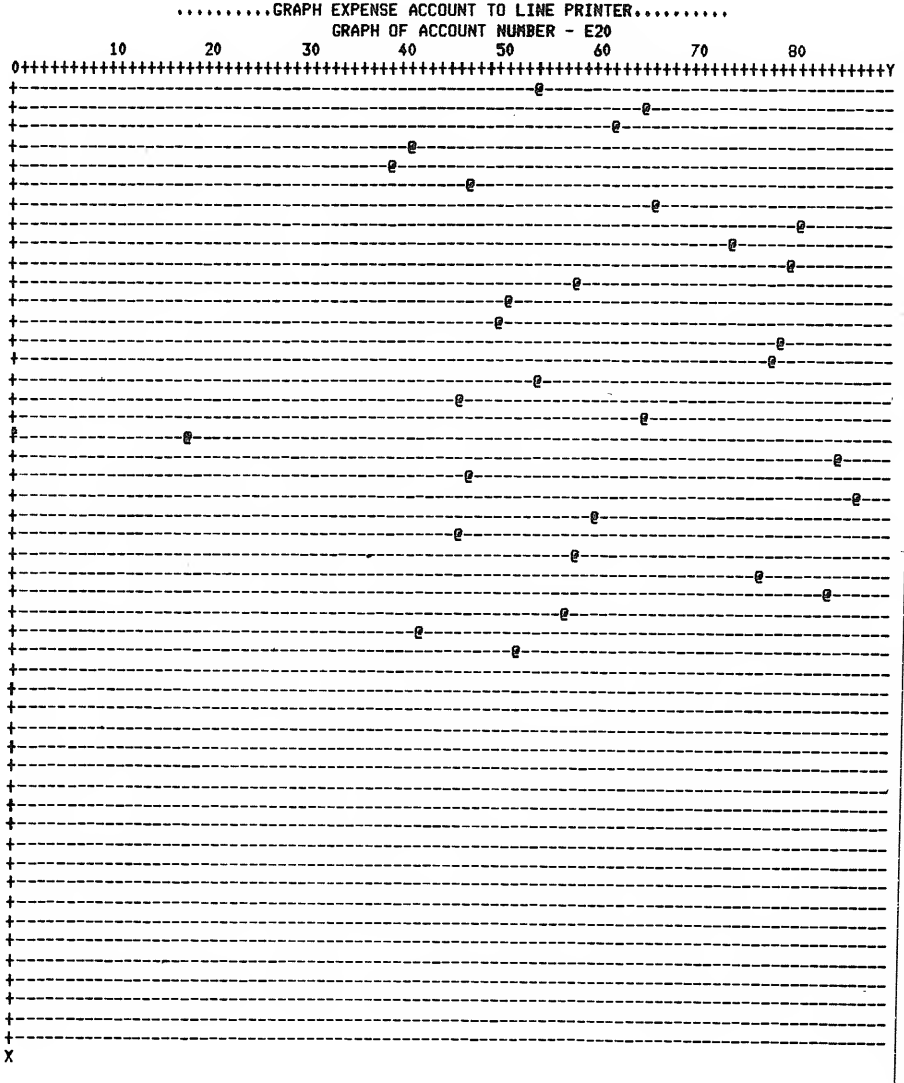
SAMPLE OUTPUT FROM EXPENSES

```
.........07/22/79.........

     HOUSE
          E1-    $1,216.77
     MOTELS
          E2-      $272.30
     SUPER MARKET
          E3-       $79.96
     ENTERTAIN
          E4-      $133.32
     DRUGSTORE
          E5-       $46.66
     GAS
          E6-      $379.25
     INSUR.
          E7-       $68.00
     CAR
          E8-       $89.10
     CLOTHES/D
          E9-      $100.00
     CLOTHES/M
          E10-     $250.00
     CLOTHES/B
          E11-     $300.00
     RESTAURANTS
          E12-     $300.00
     MEDICAL
          E13-       $0.00
     P.V.CLUB
          E14-       $0.00
     SCHOOL
          E15-       $0.00
     TRASH
          E16-       $0.00
     MISC
          E17-       $0.00
     TRIPS(TOTAL)
          E18-     $420.00
     PHONE
          E19-     $134.38
     ELECTRIC
          E20-     $738.12

GRAND TOTAL TO DATE IS   $4,527.86
```

```
10 '-PROGRAM BY BETHANY PRENDERGAST 10129 LEISURE LANE N.
20 '-JAX.,FLA.904-642-1902. WRITTEN FOR TRS-80 LEVEL II,
30 '-48K DISK SYSTEM WITH LINE PRINTER ATTACHED. TO BE RUN
40 '-IN CONJUNCTION WITH EXPENSES PROGRAM.
50 CLEAR(1500):DIMY$(200):DIMA(200):DIMA$(200):DIMT(50)
60 CLS
70 PRINT@400,"TRANSFER/EXP"
80 PRINT:INPUT"WANT HARD COPY OF TOTALS TO DATE";HH$
90 '-PROGRAM BY BETHANY PRENDERGAST 10129 LEISURE LANE N.
100 '-JAX.,FLA.904-642-1902. THIS PROGRAM IS A UTILITY
110 '-ROUTINE TO BE RUN WITH "EXPENSES" FOR THE PURPOSE
120 '-OF INITIALIZING DAILY TRANSACTION DISK AFTER
130 '-TRANSFER OF TOTALS TO MONTHLY DISK
```

```
140 GOTO 310
150 REM- BLINK ROUTINE FOR DAILY
160 CLS
170 PRINT@400,"INSERT DAILY DISK(ENTER)"
180 FOR Y=1 TO 100:NEXT Y
190 B$=INKEY$
200 IF B$="" THEN 210   ELSE 220
210 CLS: GOTO 170
220 RETURN
230 REM- BLINK ROUTINE FOR MONTHLY
240 CLS
250 PRINT@400,"INSERT MONTHLY DISK (ENTER)"
260 FOR Y-1 TO 100: NEXT Y
270 B$=INKEY$
280 IF B$="" THEN290   ELSE 300
290 CLS: GOTO250
300 RETURN
310 T$="E":GOSUB 150
320 Z=1
330 C$=STR$(Z)
340 IF LEN(C$)=2 THEN N=1
350 IF LEN(C$)=3 THEN N=2
360 D$= RIGHT$(C$,N)
370 F$=T$+D$:C1=0
380 FOR Y= 1 TO 50:NEXT Y
390 OPEN"R",1,F$:X=1
400 FOR I=1 TO LOF(1): GET 1,I:FOR E=0 TO 16
410 FIELD 1,(E*15) AS DUM$, 11 AS Y$, 4 AS A$
420 U$="              "
430 R$= Y$: IF R$ = U$ THEN CLOSE 1: GOTO470
440 Y$(X)=Y$:A(X)=CVS(A$)
450 C1=C1+A(X)
460 X=X+1: NEXT E: NEXT I: CLOSE 1
470 T(Z)=C1
480 IF HH$="YES" THEN 490   ELSE 500
490 LPRINTCHR$(30);LPRINTF$;"-";T(Z);LPRINTCHR$(29)
500 PRINTF$;"-";T(Z),
510 IF Z=20 THEN GOTO 540
520 Z=Z+1
530 GOTO 330
540 FOR Y=1 TO 50:NEXT Y:GOSUB 230
550 LINE INPUT "ENTER MT./YR. FOR SAVE?";Q$
560 FOR X= 1 TO 150
570 A(X)=0:Y$(X)=""
580 NEXT X
590 CLS:PRINT"SAVING TOTALS"
600 T$="E": Z=1
610 FOR Y= 1TO 50:NEXT Y:C$=STR$(Z)
620 CLOSE 1
630 IF LEN (C$)=2 THEN N=1
640 IF LEN (C$)=3 THEN N=2
650 D$= RIGHT$(C$,N)
660 F$=T$+D$
670 PRINTF$,
680 '-CHECK TO SEE IF FILE IS ON THE DISK
690 ON ERROR GOTO 1480
700 OPEN"I",1,F$:CLOSE 1
710 ON ERROR GOTO 0
720 '-DD$=255 BLANKS TO CLEAR BUFFER
730 DD$="
740 OPEN"R",1,F$:FIELD 1,255 AS Z$
750 LSET Z$=DD$:CLOSE 1: FOR Y=1 TO 50 :NEXT Y
760 '-FILE EXISTS & MUST BE READ BEFORE WRITING TOTAL
770 OPEN"R",1,F$:X=1:L=LOF(1)
780 GET1,L:FOR E=0 TO 16
790 FIELD 1,(E*15) AS DUM$,11 AS Y$,4 AS A$
800 U$="              "
810 '-SET FAKE VARIABLE TO SEE IF DATA FIELD IS EMPTY
820 R$=Y$:IF R$=U$ THEN 850
830 Y$(X)=Y$:A(X)=CVS(A$)
840 X=X+1:NEXT E: GOTO 980
850 Y$(X)=Q$: A(X)=T(Z)
860 CLOSE:FOR Y=1 TO 50:NEXT Y:OPEN"R",1,F$
870 FIELD 1,255 AS Z$:LSET Z$=DD$ :CLOSE:FOR Y=1 TO 10:
880 OPEN"R",1,F$
890 M=1
900 FOR E=0 TO 16
910 FIELD 1,(E*15) AS DUM$, 11 AS Y$, 4 AS A$
920 IF Y$(M)="" THEN PUT 1,L:GOTO950
930 LSET Y$=Y$(M): RSET A$=MKS$(A(M))
940 M=M+1:NEXT E: PUT 1,L
950 CLOSE 1:FORX=1TO150:Y$(X)="":A(X)=0:NEXTX
960 IF Z=20 THEN 1050
970 Z=Z+1: GOTO 610
980 '-START A NEW SECTOR BECAUSE READ SHOWS LAST SECTOR
      FULL - CLEAR BUFFER!
```

```
990 CLOSE 1:FOR Y=1 TO 50: NEXT Y: OPEN"R",1,F$
1000 FIELD 1,255AS Z$:LSETZ$=DD$:CLOSE1:FOR Y=1 TO 10:
     NEXT Y
1010 OPEN"R",1,F$:FIELD 1,11 AS Y$,4 AS A$,240 AS DUM$
1020 LSET Y$=Q$: RSET A$=MKS$(T(Z))
1030 L=L+1: PUT 1,L
1040 GOTO 950
1050 CLS:PRINT"DO YOU WISH TO KILL AND REINITIALIZE OLD
     DAILY FILES (YES/NO)"
1060 INPUT QQ$
1070 IF QQ$="YES" THEN 1100
1080 IF QQ$="NO" THEN PRINT"PROGRAM CONCLUDED":END
1090 IF QQ$="" THEN 1050
1100 GOSUB 170
1110 CLS:PRINT@400," KILLING FILES"
1120 T$="E": Z=1
1130 C$=STR$(Z)
1140 IF LEN (C$)=2 THEN N=1
1150 IF LEN (C$)=3 THEN N=2
1160 D$= RIGHT$(C$,N)
1170 F$=T$+D$
1180 PRINTF$,
1190 KILL F$
1200 FOR Y=1 TO 50 : NEXT Y
1210 IF Z= 20 THEN 1250
1220 Z=Z+1
1230 GOTO 1130
1240 '-REINITIALIZE FILES FOR NEW MONTH
1250 FOR Y= 1 TO 50 : NEXT Y
1260 CLS: LINE INPUT"ENTER MT./YR.(USE COMMA)
     FOR FIRST ENTRY ON NEW FILES?";Y$(X)
1270 CLS: PRINT@400," INITIALIZING FILES"
1280 T$="E":Z=1:A(X)=0
1290 C$=STR$(Z)
1300 IF LEN(C$)=2 THEN N=1
1310 IF LEN (C$)=3 THEN N=2
1320 D$=RIGHT$(C$,N)
1330 F$=T$+D$
1340 PRINTF$,
1350 '-FILLER FOR BUFFER
1360 V$="
1370 OPEN"R",1,F$
1380 FIELD 1,11 AS Y$,4 AS A$,240 AS DUM$
1390 LSET Y$=Y$(X): RSET A$= MKS$(A(X))
1400 LSET DUM$=V$
1410 PUT 1,1: CLOSE1
1420 FOR Y=1 TO 50: NEXT Y
1430 IF Z= 20 THEN 1460
1440 Z=Z+1
1450 GOTO 1290
1460 CLS:PRINT" FILES HAVE BEEN INITIALIZED AS  ";Y$(X)
1470 PRINT:PRINT"PROGRAM CONCLUDED": END
1480 '-FILE MUST BE CREATED & TOTAL WRITTEN
1490 '-DD$=255 BLANKS TO CLEAR BUFFER
1500 CLOSE1:OPEN"R",1,F$
1510 FIELD 1,255 AS Z$
1520 DD$="
       "
1530 LSET Z$=DD$
1540 CLOSE 1
1550 OPEN"R",1,F$
1560 FIELD 1,11 AS Y$, 4 AS A$, 240 AS DUM$
1570 LSET Y$=Q$: RSET A$=MKS$(T(Z))
1580 PUT 1,1:CLOSE 1
1590 IF Z=20 THEN 1050
1600 Z=Z+1
1610 RESUME 610
```




```
10 '-PROGRAM LISTING FOR CREATIVE COMPUTING - 07/15/79
20 '-PROGRAM BY BETHANY PRENDERGAST 904/642-1902 TO BE
     RUN IN
30 '-CONJUNCTION WITH EXPENSES PROGRAM.
40 CLS:CLEAR(5000):DIM Y$(200):DIM A(120): DIM A$(120)
50 LPRINTCHR$(30)
60 LPRINT"
   .....GRAPH EXPENSE ACCOUNT TO LINE PRINTER....."



70 PRINT"            GRAPH/EXP PROGRAM"
80 INPUT"ENTER NO.OF LINES ON X-AXIS DESIRED (54 TO A PAGE)";LL
90 INPUT"ENTER FILE (IN QUOTES) TO BE READ";F$
100 LPRINT"  GRAPH OF ACCOUNT NUMBER - ";F$
110 OPEN"R",1,F$:X=1
120 FOR I=1 TO LOF(1):GET 1,I:FOR E=0 TO 16
130 FIELD1, (E*15)AS DUM$,11AS Y$,4AS A$
140 Y$(X)=Y$:A(X)=CVS(A$)
150 U$="            ":IF Y$(X)=U$ THEN 170
160 X=X+1:NEXT E:NEXT I
170 CLOSE1:N=X-1
180 PRINT"WORKING"
190 Z=N
200 '-CHANGE Y VALUES TO INTEGERS
210 FOR X= 1 TO Z
220 A(X)=INT(A(X)+.5)
230 PRINTA(X);
240 NEXT X
250 '-CHECK THAT VALUES OF Y NOT > 90 FOR LPRINT
260 FOR X = 1 TO Z
270 IF A(X)>90 THEN 290  ELSE NEXT X
280 GOTO410
290 '-SCALE DOWN Y TO PERMIT LPRINT
300 IF A(X)> 90 THEN D=2
310 IF A(X)>180 THEN D=4
320 IF A(X)>360 THEN D=8
330 IF A(X)>720 THEN D=10
340 IF A(X)>900 THEN D=20
350 IF A(X)>1800 THEN 860
360 IF D>0 THEN LPRINT
    ( Y AXIS VALUES HAVE BEEN DIVIDED BY ";D;")"
```

```
370 FOR X= 1 TO Z
380 A(X)=INT((A(X)+.5)/D)
390 '- GO BACK AND CHECK ALL NUMBERS AGAIN FOR SCALE
400 NEXT X : GOTO 260
410 '-DETERMINE WHERE ON LINE Y FALLS
420 FOR I= 1 TO Z
430 '- SET UP A$(I) TO EQUAL NO. OF BLANKS = TO A(I)
440 FOR L=1 TO A(I)
450 S$=S$+","
460 NEXT L
470 A$(I)=S$
480 PRINTA$(I)
490 S$=""
500 NEXT I
510 CLS:PRINT"COMPUTING GRAPH FOR LINE PRINTER"
520 LPRINTCHR$(30);"     10        20        30        40
    50       60        7(       80"
530 '-A$(I) NOW HAS NO OF BLANK'S CORRESPONDING TO INTEGER
    VALUE OF A(X)
540 '-THIS PROGRAM GRAPHS NUMERIC DATA TO THE PRINTER
550 '-SET STRING OF +"+" SEPARATED BY BLANKS
560 T$="0+++"
570 FOR X= 1 TO 21
580 T$=T$+"++++"
590 NEXT X
600 '-SET UP Y AXIS FOR PRINT
610 T$=T$+"++Y"
620 LPRINTT$
630 '-SET STRING OF "-"'S AND THEN PUT "@"
    WHERE IT BELONGS ON LINE
640 FOR X = 1 TO Z
650 L$="+---"
660 FOR Y= 1 TO 22
670 L$=L$+"----"
680 NEXT Y
690 P=LEN(A$(X))
700 IF P=0 THEN  790
710 A$=LEFT$(L$,(P))
720 P1=91-(P+1)
730 B$=RIGHT$(L$,P1)
740 P1=0
750 L$=A$+"@"+B$
760 LPRINT L$
770 L$="":A$="":B$=""
780 GOTO820
790 B$=RIGHT$(L$,83)
800 L$="@"+B$
810 GOTO 760
820 NEXT X
830 IF Z<60 THEN GOTO 880
840 LPRINT"X":LPRINTCHR$(28)
850 END
860 CLS:PRINT"Y POINTS ARE> 1800 ,
    PLEASE SCALE DOWN & RERUN !"
870 END
880 FOR X= 1 TO (LL-Z)
890 W$="+"
900 FOR Q= 1 TO 9
910 W$=W$+"----------"
920 NEXT Q
930 LPRINT W$
940 NEXT X
950 LPRINT"X"
960 LPRINTCHR$(28)
970 END
```

Maxi C.R.A.S.

Check Register and Accounting System, Too?

C.A. Johnson

Good grief! What is this we find on our pages? A review of a checkbook balancing program—the very thing we have so cynically pooh-poohed for so many years. We never thought we'd see the day, but here it is. We have published a review of a check register program that made us want to buy one. If you are still waiting for the day, read on.

I have seen several programs that provide a record of checkbook activity. Generally, they provide a subroutine for balancing your checkbook against the monthly bank statement, and not much more. For me, they have been more trouble than they were worth.

Perhaps, I did not learn how to use them properly, but it appeared to me that they more than doubled the work of maintaining my checkbook. When you have to keep a manual record so that you will know the status of your bank account any time you write a check, entering the data again into the computer just to reconcile your records with the bank statement hardly seems worth the effort and takes longer than balancing with a pocket calculator.

So, I had given up on computerizing my checkbook records—until *Maxi C.R.A.S.*

Maxi C.R.A.S. is not only a check register system, but an excellent double entry accounting system, as well. It is a conversational, menu-driven program. It is so user-oriented that even the newest computer tyro can use it without difficulty, yet it provides a sophisticated set of reports which are designed to aid in money management and ease the pain of completing the Internal Revenue Service forms.

The documentation is well written, clear, and to the point. All features of the program appear to have been dealt with in an excellent manner. Some hints on advanced uses of the program are provided, but I am convinced that there are many additional applications and variations of this program which will be developed by those who make serious use of it.

The Account Structure

Naturally, as with any accounting system, the first chore is to define the account structure. In *Maxi C.R.A.S.*, you may establish as many as 223 accounts, divided in any way you choose between expense and income accounts. Once the account structure has been established and the transactions entered, the account structure cannot be altered nor can any of the transactions be edited. Of course, you can always start over, define a new account structure and reenter the data, if you wish.

To allow for oversights in defining the accounts, I recommend that you include several "undefined" accounts in both the expense and income account lists. *Maxi C.R.A.S.* allows you to edit the account titles later. If you reserve a larger number of accounts than you initially think you need and enter the extra accounts with blank titles, you may add the actual titles later. The accounts will be there to use as you discover your need for them.

The program does not restrict you to a one check-one account entry. If you write a check for a group of items which span several accounts, you may enter the total amount of the check for the check register and then break out the amounts to be entered into the accounting system according to the accounts to which they belong.

It is a standard practice in accounting that nothing is ever erased once it is entered into the books. Errors are generally corrected by entering new transactions, properly identified as corrections, to offset the errors. *Maxi C.R.A.S.* conforms to this practice.

After entering the data for a transaction into the computer, you are given a last chance to review it and either cancel it or add it to the disk file. You may include a note as part of the transaction if an explanation is desired. Once the transaction has been written to disk, however, it cannot be changed.

By including a CASH account, you can handle cash expenditures without affecting your checkbook balance. This is done by entering a 0.00 for the check amount and offsetting the purchase with a negative CASH amount and a positive amount for the other accounts.

SOFTWARE PROFILE

Name: Maxi C.R.A.S.
Type: Checkbook Processor
Systems: TRS-80 Model I or III, 48K, 2 Disk Drives
Format: TRS-80 Disk (Both Model I and III included)
Language: Basic
Summary: Excellent menu-driven double entry accounting system
Price: $99.95
Manufacturer:
Adventure International
P.O. Box 3435
Longwood, FL 32750

C.A. Johnson, 3619 Sugarhill Dr., San Antonio, TX 78230.

Figure 1.

09/26/82 Check Register for September, 1982 Page 1

A. B. Yourname 123-345-6 AnyBank USA

TXN #	Day	C:*	Check #	Payee/Payor	In Payment Of	Amount Paid	Amount Received	Balance	N:*	TXN #
Opening balances								500.00		
1:	2:			Magazine A	Stop Smoking	0.00	250.00	750.00	*	1:
2:	4:		501:	Local Office Supply Store	5 Reams Paper	42.56	0.00	707.44		2:
3:	4:		502:	Cash	Petty Cash	100.00	0.00	607.44		3:
4:	5:			Post Office	Postage	0.00	0.00	607.44		4:
5:	7:		503:	Youbuyit Office Supply	Typist Chair	78.00	0.00	529.44		5:
6:	8:			Youreadit Publishers	Second year royalties	0.00	2,000.00	2,529.44	*	6:
7:	15:			Self	Transportation	0.00	0.00	2,529.44	*	7:
8:	20:		505:	Local Office Supplies	Envelopes	20.00	0.00	2,509.44		8:
Closing balances for September, 1982						240.56	2,250.00	2,509.44		

Figure 2.

09/26/82 Income & Expense Account Subtotals for September, 1982 Page 1

A. B. Yourname 123-345-6 AnyBank USA

No.	Account Title	Opening Balance	Monthly Subtotal	Closing Balance
1:	Office Supplies	0.00	62.56	62.56
2:	Postage	0.00	16.00	16.00
3:	Office Furniture	0.00	78.00	78.00
4:	Telephone	0.00	0.00	0.00
5:	Transportation	0.00	18.00	18.00
6:	Cash	0.00	66.00	66.00
7:		0.00	0.00	0.00
8:		0.00	0.00	0.00
9:		0.00	0.00	0.00
10:		0.00	0.00	0.00
11:	Article Sales	0.00	-250.00	-250.00
12:	Book Royalties	0.00	-2,000.00	-2,000.00
13:	Newspaper Features	0.00	0.00	0.00
14:	Ghosting	0.00	0.00	0.00
15:	Other	0.00	0.00	0.00
16:		0.00	0.00	0.00
17:		0.00	0.00	0.00
18:		0.00	0.00	0.00
19:		0.00	0.00	0.00
20:		0.00	0.00	0.00

Printing Checks

Maxi C.R.A.S. provides for printing checks as well as reports. At the time the transactions are entered, you are asked if you wish to print the check. If you reply "Yes," a notation is made into a special file. When you are ready to print checks, you select the proper item from the menu and indicate whether you are using continuous form checks.

If you answer N you may print checks singly. The program will *not* allow you to print a check more than once. If you wish to use window envelopes to mail your checks, you may build an address file and the program will pick up the address and include it on the check at printing time.

Several useful reports are provided. The Check Register Statement (Figure 1) is a two-page report (too wide for a single page printing) showing all of the transactions during the month with a running balance.

Other reports include check Register Notes, Income and Expense Account Subtotals (Figure 2), selected Account Statement (Figure 3), Checkbook Reconciliation, and Account Distribution Statement (Figure 4).

Pros and Cons

One of the outstanding features of *Maxi C.R.A.S.* is its ability to interface with VisiCalc. The format for the file set up for VisiCalc is as follows: Account names are in the first column, opening balances for each account in the second column, and monthly balances in the remaining columns. Once you get the data into VisiCalc, you can manipulate it in many ways. If you are a VisiCalc

Figure 3.

```
09/26/82          Statement of Account No. 1 -- Office Supplies          Page 1

                     For September, 1982 through September, 1982

                     A. B. Yourname       123-345-6       AnyBank USA

TXN#:MO:DA:   Payee/Payor    : In Payment Of  :CHK #:   Amount  :   Subtotal  :N
-----:--:--:-----------------:----------------:------:----------:------------:-
Opening balance                                      :          :       0.00:
   2: 9: 4:Local Office Supp:5 Reams Paper    :  501:     42.56:      42.56:
   8: 9:20:Local Office Supp:Envelopes        :  505:     20.00:      62.56:
```

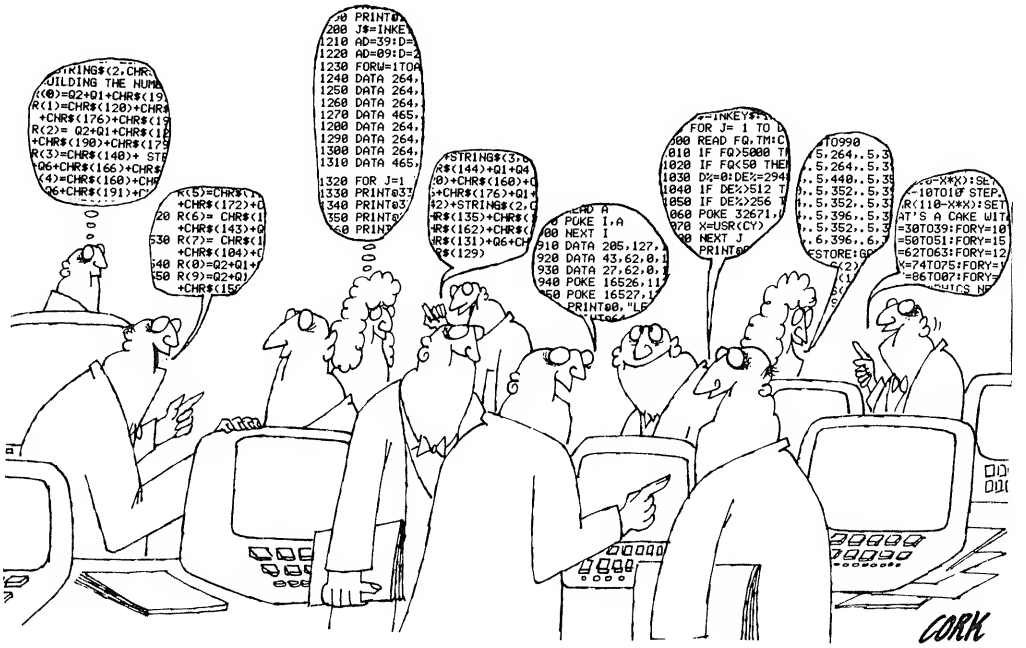
Figure 4.

```
09/26/82        Account Distribution Statement For September, 1982       Page 1

                   A. B. Yourname       123-345-6       AnyBank USA

TXN #:          Payee/Payor           :ACT #:    Amount    :    Account Name
------:-----------------------------:------:-------------:--------------------
    1:Magazine A                     :   11 :     -250.00:Article Sales
    2:Local Office Supply Store      :    1 :       42.56:Office Supplies
    3:Cash                           :    6 :      100.00:Cash
    4:Post Office                    :    2 :       16.00:Postage
                                     :    6 :      -16.00:Cash
    5:Youbuyit Office Supply         :    3 :       78.00:Office Furniture
    6:Youreadit Publishers           :   12 :   -2,000.00:Book Royalties
    7:Self                           :    5 :       18.00:Transportation
                                     :    6 :      -18.00:Cash
    8:Local Office Supplies          :    1 :       20.00:Office Supplies
```

devotee as I am, you will agree that this feature alone is worth the price of the program.

Of all the many things that *Maxi C.R.A.S.* does, there is only one that I am not happy with. The report program is designed to be used with continuous form paper. There is no provision for pausing at the end of each page to insert a new sheet of paper. I attempted to provide the Figures for this article with my daisy wheel printer and gave up in utter frustration. I could not insert the new pages quickly enough. It seems to me that it would have been a simple matter to have provided the capability to select either continuous feed or single sheet reporting.

Written specifically for the TRS-80, *Maxi C.R.A.S.* is supplied with both Model I and Model III versions. The Model I version comes on two disks, and the Model III is supplied on a single disk. Both versions are delivered on disks with TDOS, a stripped down version of DOSPLUS. If you wish to use a different operating system, the instructions for conversion are provided. □



Chapter V
Programming Tips

How Not to Be Out of Sorts

Part I — Insertion Sort

Albert Nijenhuis

Why sort?

Sooner or later — always sooner than expected — you will have to do some sorting. Your list to be sorted may be a symbol table in a compiler, which has to be alphabetized, or you may want the largest 100 items, in order, in a list of 10,000. Perhaps you are a student who wants to update the mailing list for his school paper. Or, if you have a business, you may want to put in order the recently active accounts before updating the entire file.

Anyhow, computers spend a large amount of their time just sorting one list or another. Finding the right method for each job is therefore of great importance.

A Few Methods.

In this article and in two more parts to follow we will discuss just a few of the many available sorting methods, and indicate situations in which they are particularly useful.

Before you read on, it is useful to ask yourself how you would sort a list of, say, six numbers in increasing order. Chances are you would end up with something similar to the insertion sort which we discuss in this article, or the "bubble sort," which is very similar, and is often used as a programming exercise.

The **Insertion Sort** is extremely simple in concept, and is very easy to program. It is very suitable when, for example, you have to sort lists of length up to about 10. When the lists are longer, the number of comparisons it makes become very large, and the method is definitely not recommended. For short lists it is nevertheless very fast, because there is a minimum of book-keeping involved.

Heapsort is a method which greatly economizes on the number of comparisons that are performed, by using the general principle that if "a < b" and "b < c" have been established, it is no longer necessary to compare a and c. Since the per-step bookkeeping is a bit more involved, it works a bit more slowly on short lists than the insertion sort.

Both insertion sort and heapsort use no array space except that in which the input data is stored. This feature makes these methods simple to use as a part of bigger programs.

The third method is a **linked merge-sort,** and will be discussed in the third article. It is particularly useful in situations where the items to be sorted are bulky and hard to displace, or where the input data already has so much of the desired order that not using this order would be wasteful. This sorting method does require some working storage, and its program is longer than the other two.

Insertion Sort.

We shall assume that the numbers to be sorted are stored in an array with n locations a(1), . . ., a(n), and the sorting consists of switching the contents of these locations so these contents are listed in increasing order.

The following way of looking at the problem is extremely important in understanding algorithms: we assume that the first j elements a(1) . . . a(j) are already in order. On first sight this may seem like magic — it may suggest that we have already solved the problem that we wanted to solve. In fact, it does nothing of the kind: whatever magic there is, is in the replacement of the fixed given subscript n by the subscript j which may, in principle, take any of the values 1, . . ., n. The statement "a(1), . . ., a(j) are already in increasing order" will be true for certain values of j and not for others. If it is indeed true for j=n, then we are finished, but that is only seldom the case. All we are allowed to assume is its truth for j=1, because any list with only one member is automatically sorted. What we shall do is design a method which, assuming the statement is true for any particular j, will cause it to be true for the next value of j. This method will thus lead us, step by step, from j=1 to j=n.

For example, suppose we had the list consisting of 1, 2, 6, 8, 12, 5. Then the statement is true for j=5, but not for j=6. To make it true for j=6, we compare 5 and 12, and interchange the two because they are out of order. This yields the list 1, 2, 6, 8, 5, 12. Still focusing on 5, we now compare it with 8, and interchange. Next is the comparison with 6 in the new list 1, 2, 6, 5, 8, 12, which again leads to an interchange. The next comparison, with 2, does not lead to an interchange, and we have just achieved the truth of the statement for j=6.

The story with general values of j is

Albert Nijenhuis, University of Pennsylvania, Dept. of Mathematics, Philadelphia, PA 19104.

remarkably the same, though we rephrase it slightly. We first store a(j+1) in b to simplify the procedure. (So, in the previous paragraph we set b=5.) Now, compare b with a(j), a(j-1), etc.; or to put it even better: compare b with a(i), where i successively takes the values j, j-1, . . . until termination is called for. If the comparison of b and a(i) yields that b is the smaller one, then move a(i) to a(i+1), and repeat the comparison, with i replaced by i-1. If, however, b is greater than, or equal to, a(i), then insert b into a(i+1). Finally, if i reaches O, then insert b into a(1). The insertion of b terminates the process, and the contents of a(1), . . ., a(j+1) are now in order.

The Basic program in Figure 1 performs the above insertion (lines 1020 to 1080) in a loop or J=1 to n-1. Figure 2 shows the sorting of a list, with the j-value above each column. The input list is in the left-most column.

```
1000 'SUBR: INSERTION SORT OF  A
     OF DIM  N
1010 FOR J=1 TO N-1
1020 LET B=A(J+1)
1030 FOR I=J TO 1 STEP -1
1040 IF B>=A(I) GOTO 1080
1050 LET A(I+1)=A(I)
1060 NEXT I
1070 LET I=0
1080 LET A(I+1)=B
1090 NEXT J
1100 RETURN
```

Figure 1

J=	1	2	3	4	5	6	7	8	9
3	3	3	2	1	1	1	1	1	1
7	7	4	3	2	2	2	2	2	2
4	4	7	4	3	3	3	3	3	3
2	2	2	7	4	4	4	4	4	4
1	1	1	1	7	7	7	6	6	5
9	9	9	9	9	9	8	7	7	6
8	8	8	8	8	8	9	8	8	7
6	6	6	6	6	6	6	9	9	8
10	10	10	10	10	10	10	10	10	9
5	5	5	5	5	5	5	5	5	10

Figure 2

Sources On Sorting.

The literature on sorting is very extensive. Fortunately, a virtual encyclopedia of what happened before 1973 can be found in Vol. 3 of D.E. Knuth's *The Art of Computer Programming,* where some 350 pages are devoted to this one subject. Meanwhile, developments continue to be published in several journals, such as those of the Association for Computing Machinery. Look for them in your public or school library, or the library of a nearby college with an engineering program. □

How Not to Be Out of Sorts

Part II — Heap Sort

Albert Nijenhuis

The first part of this series of three articles discussed the insertion sort as an excellent method to sort short lists. In this part we discuss the amazing Heapsort. Its program is short, no auxiliary storage is required, and its method is intriguingly clever, though initially not simple. It was discovered by Williams and Floyd, in 1964. Our description closely follows that in "Combinatorial Algorithms" by the author and H.S. Wilf (Acad. Press), where the reader can also find a compact program in Fortran.

Albert Nijenhuis, University of Pennsylvania, Dept. of Mathematics, Philadelphia, PA 19104.

What Is A Heap?

For the purposes of heapsort the best way to visualize the array a(1) . . . a(n) to be sorted is shown in Figure 3. In the first row one box represents a(1), in the next row two boxes represents a(2) and a(3), etc.; each row contains twice as many array members as the previous one, until we run out (n=26 in Figure 3).

Each box ("father") is connected to two boxes (his "sons"), as long as the supply lasts. (The terminology precedes Women's Lib!) Figure 3 is an example of a so-called BINARY TREE, ROOTED at box 1. It is useful to observe that each box is the root of a smaller binary tree, e.g., box 5 is the root of a binary tree consisting of boxes 5, 10, 20, 21, 22, 23, while box 14 is a binary tree all by itself.

It is easy to determine the structure of a binary tree with n boxes without a picture: box i is the father of boxes 2i and 2i+1, so long as these do not exceed n.

A binary tree is called a *heap* if the elements stored in the boxes have the property that the value stored at each father is greater than, or equal to, that at each of the sons: if 2i<n this means $a(i) \geq a(2i)$ and $a(i) \geq a(2i+1)$, while if 2i=n it means only the first of these two conditions — if 2i>n there is no

condition on the (son-less) fatherhood at box i. For example, in Figure 3 the subtrees rooted at 4 and 5 are heaps, while the tree rooted at 6 is not a heap.

Creating A Heap.

The first phase of heapsort consists in converting the binary tree into a heap. This is accomplished by a carefully designed sequence of interchanges of entries between father and son. In analogy to Part I we assume that at any time part of the desired heap structure is already present, and we shall extend it, until the whole binary tree is a heap. For example, if an interchange between a father and son is contemplated, the binary trees rooted at the sons (there may be 0, 1 or 2 of them) are already heaps. Initially, all the subtrees rooted at "son-less" fathers are heaps.

So, suppose that the strategy calls for a possible interchange at father i; see Figure 4. Then we first compare

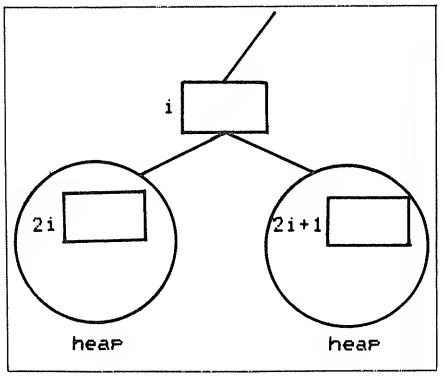


Figure 4

the sons 2i and 2i+1 to find the larger one of the two; let j=2i or j=2i+1 as the case may be (if 2i=n, we always have j=2i). Next, compare a(i) an a(j). If a(i) ≥ a(j), the heap condition holds at box i, since the value at the father is greater or equal to that at the sons. Otherwise, we interchange a(i) and a(j), thereby insuring that the new values satisfy the heap condition at box i. Note, however, that the heap condition may no longer hold at box j, since the new value is less than the old one. We may, therefore, have to readjust the situation at box j. In view of the fact that the binary trees rooted at the sons of box j are heaps, we have the same problem again, though of a smaller size. It may be hard to believe, but this one operation of father-son interchanges is the building block of the whole heapsort!

Referring to Figure 3, let's make a heap. First of all, the binary trees rooted at boxes 13 through 26 are (trivially) heaps. Now, first make a

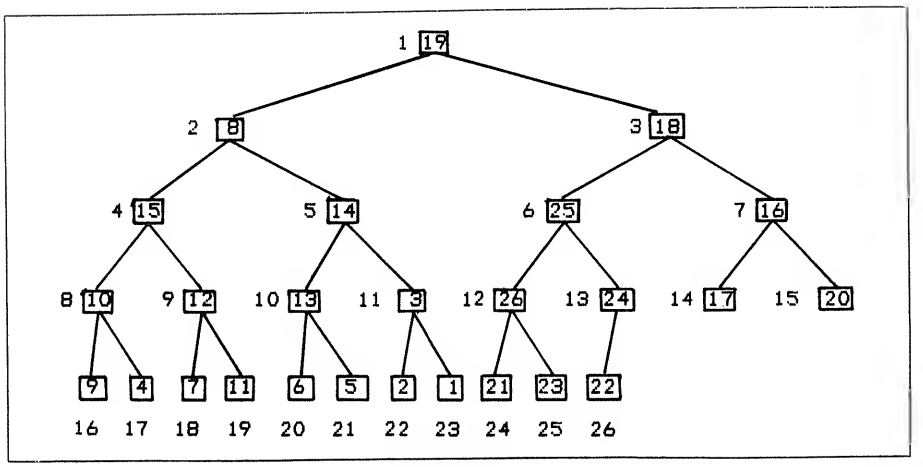


Figure 3

```
1000 'SUBR: HEAPSORT OF  A(1),
     ...,A(N)
1010 'PHASE 1
1020 LET M=N
1030 FOR L=INT(N/2) TO  1 STEP -1
1040 LET B=A(L)
1050 GOSUB  1150
1060 NEXT L
1070 'PHASE 2
1080 LET L=1
1090 FOR M=N-1 TO 1 STEP -1
1100 LET B=A(M+1)
1110 LET A(M+1)=A(1)
1120 GOSUB 1150
1130 NEXT M
1140 RETURN
1150 'TOHEAP
1160 LET I=L
1170 LET J=I+I
1180 IF J>M GOTO 1250
1190 IF J=M GOTO 1210
1200 IF A(J+1)>A(J) THEN LET J=J+1
1210 IF B>=A(J) GOTO 1250
1220 LET A(I)=A(J)
1230 LET I=J
1240 GOTO 1170
1250 LET A(I)=B
1260 RETURN
```

Figure 5

heap out of the subtree rooted at 13, then 12, etc., through 8. After this, continue with the binary rooted at 7, 6, etc., all the way to 1. In this last group an interchange at the root may lead to further interchanges below it, of course.

Let's follow in detail what happens at box 6. Observe that the binary trees rooted at 12 and 13 are heaps. Now a(12) > a(13), so compare a(6) and a(12), which calls for an interchange. Now compare a(24) and a(25) to find the larger one. So, compare the new a(12)=25 with a(25). Since the latter is smaller, no further interchanges are needed; otherwise, we would have interchanged the new a(12) and a(25).

Algorithm TOHEAP.

We assume that the binary tree rooted at box I has to be made into a heap, while the binary trees rooted at the sons (if any) of box I are heaps. The highest-numbered box is not to exceed m (this was n in section 6; we will see later why).

Step 1 (Initialize) Set b ← a(I); set i ← I1

Step 2 (Find a son, if any) Set j ← 2i; if j > m goto step 5

Step 3 (If there is a second son, let j be the larger one) If j=m, goto step 4, else, if a(j+1) > a(j) set j ← j+1

Step 4 (Compare b with a(j)) If b ≤ a(j) goto step 5, else (move a(j) up) set a(i) ← a(j) and i ← j, then goto step 2

Step 5 (Insert b) Set a(i) ← b, EXIT

Phase 1 (creating a heap) is now accomplished by performing TOHEAP with m=n, and with 1 running backward from [n/2] to 1. (We use [x] to denote the largest integer in x; e.g., [7.5] = 7, [6] = 6.)

Heapsort, Phase 2.

So — now that we have a heap: what next? We wanted to SORT the list, didn't we? And what the heap gives us is a pile, rather strangely organized, with the largest element on top. It is not even clear where the second-largest element lives; and as to the third-largest . . . Nevertheless, in just a few lines we can describe how it all ends.

Take a(1), the largest element, and interchange with a(n). Then the largest element is where it belongs (don't touch it again!), and we now have a binary tree consisting of the boxes 1 through n-1, which is a heap, except at the root. One application of TOHEAP (1, n-1) restores the heap structure, so the (next) largest element is now in a(1). Interchange a(1) and a(n-1), and apply TOHEAP again, et. That's all!

```
Phase 1. For I = [n/2] to 1 in steps —
    1 do TOHEAP (I, n); next I
Phase 2. For m = n-1 to 1 in steps —
    1 interchange a(m+1), a(1), do
    TOHEAP (1, m), next m; EXIT
```

In Figure 5 we give a Basic program for Heapsort. The subroutine TOHEAP starts at 1150. It differs from the algorithms in section 7 only in that b(=B) is defined just prior to entering the subroutine, in instructions 1040 and 1100. The loop 1030-1060 is Phase 1, while the loop 1090-1130 is Phase 2.

Final Comments.

A careful examination of TOHEAP shows that each application of this subroutine is in fact an insertion. However, the lists on which this insertion is performed are extremely short. Other methods, e.g., merge sort methods (see the next article) require even fewer comparisons, but demand their own price, e.g., additional working storage. Also, some other methods make use of any pre-existing order in the input data. In this last respect, even the insertion sort is superior.

All with all, Heapsort is a very desirable candidate for a quick sorting, without fuss, of lists of most any length, particularly when the input is (usually) in a state of considerable disorder. □

How Not to Be Out of Sorts

Part III — Linked Merge Sort

Albert Nijenhuis

The previous two parts of this series have dealt with the insertion sort, a very short program suitable for very short lists only, and heapsort, still a short program, suitable for lists of any length. Neither method requires any working array space, and both move the items around a number of times in the given array space.

As a last method we discuss a version of merge sort. The idea is one of the oldest around, and still is among the most efficient. The version which we discuss takes full advantage of any pre-existing order in the input data. Another plus is that the records to be sorted are not displaced at all, so bulky records or records of varying sizes can be sorted by this method. To achieve all this, we store along with each record one extra word, plus a few additional words of working storage. These words will be referred to as LINKS or POINTERS, respectively.

Albert Nijenhuis, University of Pennsylvania, Dept. of Mathematics, Philadelphia, PA 19104.

Linked Lists.

The items of a linked list consist each of two parts, the LINK and the RECORD. A record is a piece of data that needs to be sorted, such as a customer's account, or the text of an address label in a mailing list. The record will contain a KEY, the part on which to sort, such as an account number, a postal (zip) code, or a name to be placed in alphabetical order. The link associated with a record is a single word, and is used in the sorting. It is used in two ways: first, the location of the link should tell the programmer how to find the key in the record (e.g., the fifth word after the link, or the item pointed to by the third item in the directory at the beginning of the record). We shall make no specific assumptions about how the key is obtained, and will simply assume that if the link is stored in location 1, the key is denoted K(1), and K(1) may be anything from the output of a subroutine K to the 1-th member of an array K. (Our Basic program will assume the latter, to avoid unneeded clutter.)

As to the content of the link, each link 1 contains the location L(1) of the link of another record, in such a way as to "thread together" all the records. A special HEAD h points to the first link L(h), and the last link point to the "null" link, some symbol that is recognized as not being the location of a link. As we start at the head, passing from link to link, we traverse the whole list. Figure 6 shows two examples of linked lists; Figure 7 shows an example of something that isn't a linked list.

Sorting a linked list means shuffling the contents of the links in such a manner that a traversal of the list yields the records in sorted order of keys.

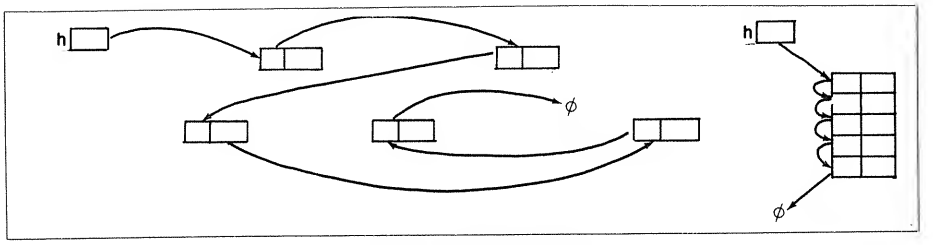


Figure 6. Two linked lists.

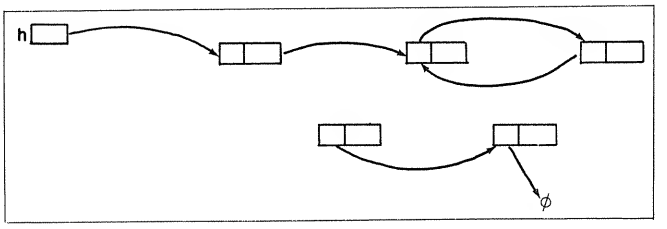


Figure 7. Not a linked list.

Merge Sorting.

If one has two sorted linked lists, they may be MERGED into one list. Examine the smallest elements of the two lists. Remove the smaller one of these two from its list and place it at the tail of the merged list (initially, the merged list is empty). This process is repeated until one list is empty, at which time the (remaining) other list is appended to the tail of the merged list.

Any linked list, when submitted for sorting, consists of shorter lists that are already sorted. Some of these sublists may be long, while others may have a length as small as 1. However, as long as there is more than one sorted sublist, these sublists may be merged in pairs, and when just one list is left, we are finished.

Note how again (as in Parts I and II) we are dealing with a structure (a family of sorted sublists) which fits the input data and of which the desired result is a special case, while throughout the sorting process we remain within the structure.

As an aside, it is possible to merge more than two lists simultaneously, but usually there is little advantage in doing so. However, this is an interesting area of experimentation. (At one time the author has merged as many as 19 lists in one pass!)

In order to find the pre-existing order in a list, we require an algorithm to locate sorted sublists of maximal length. It takes as input a linked list with head h, links L and keys K, and finds the first sorted sublist, which will be output as a linked list with head h' and the "null" link (we use the number 0) as the final link. The length of the sorted sublist is returned in m, and the remaining portion of the input list is again headed by h. The variable t points to the "tail" of the sorted sublist, and is used when new members are attached to the list.

```
ALGORITHM FIND(h,h',m)
m ← 0
h' ← h
DO
  m ← m+1
  t ← h
  h ← L(h)
Repeat while h ≠ 0 and K(t) ≤ K(h)
L(t) ← 0
EXIT
```

Test the algorithm on the list in Figure 8! It occupies lines 1370-1480 in the Basic program in Figure 10.

The second essential ingredient in the merge sort is an algorithm which merges two sorted lists, headed by h' and h", into one list headed by h'. If K(h') <= K(h"), the first item of the list headed by h' is moved to the merged list; else we first interchange h' and h". The head of the merged list is temporarily held in L(0) (any other unused location in L will do equally well).

```
ALGORITHM MERGE(h',h")
t ← 0
DO
  If K(h') > K(h") then interchange
      h' ← h"
  L(t) ← h'
  t ← h'
  h' ← L(h')
Repeat while h' ≠ 0
L(t) ← h"
h' ← L(0)
EXIT
```

Test the algorithm on the lists in Figure 9. It occupies lines 1090-1230 in the Basic program in Figure 10.

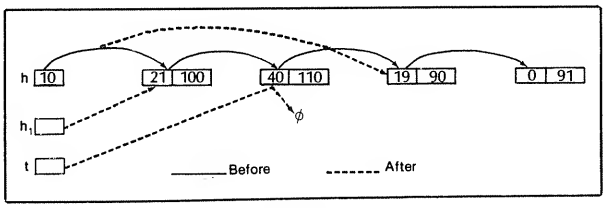


Figure 8.

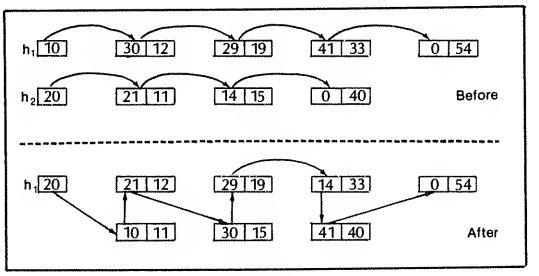


Figure 9.

Merging Strategies.

It is clear that the repeated merging of sorted lists will eventually lead to just one sorted list, which is our aim. The order in which the sorted sublists are merged can have a significant effect on the total required effort. As a rough approximation, the effort involved in merging two lists is proportional to the combined lengths of the lists. The merging of a very long list with a very short one thus accomplishes a little for a lot of work: this type of situation is to be avoided. The more equal in length two candidates for merging, the better off we are. If we did not intend to use the pre-existing order, we could use some rather rigid scheme to minimize the number of comparisons, see, e.g., the "mouse and spider" article by R. Hart in *Creative Computing*, Vol. 4, Issue 1, p. 96. Using the pre-existing order, as we have chosen to do, we are stuck with the initial lengths of the sorted sublists, and have to make the best of it from there on. Our merging strategy will have to depend on these initial lengths, yet to be efficient, will have to be simple. (We don't want to spend all our time computing strategies!)

At this point there are many options. You, the reader, may have some very useful ideas. Please develop them, and TEST them!

Here is the option we have chosen. When at all possible we shall merge two lists only if one is less than twice as long as the other. (This may not always work: suppose that initially there are three sorted sublists, of lengths 1, 10 and 100.) To accomplish this, we have an auxiliary array D, the "directory," which holds head pointers of sorted sublists (or 0 to indicate the absence of a pointer). If a sorted sublist has length 1, its head is stored in D(1), if the length is 2 or 3, we use D(2) to store its head, if the length is between 4 and 7 we use D(3), etc., moving up by a factor 2 for the minimal length each time.

The sorting begins with the search for the first sorted sublist. As it is identified, its length is obtained, and this determines the location in D in which its head is to be stored. The same is done for the next sorted sublist, etc. This continues until (pretty soon, probably) a "collision" occurs: there are two lists whose head pointers belong in the same place, say D(i). When that happens, the conflict is resolved at once: the two lists are merged. We set D(i) to 0 because the length of the new list is now such that its head deserves to be stored in D(i+1). If this location is free, the entry is made; if there is another collision there will be another merging, etc. This process continues until all sorted sublists of the input list have been exhausted.

At this point there will be a few lists left, whose head pointers are stored in some of the locations in D. We now search D for head pointers, merging as we search. And that finishes the job.

The size of the directory is easily determined: the largest list whose head pointer will ever be entered is that of the full list. Let N be its length: write it to base 2; e.g., if N=1000 we have

1000 (base ten) = 1111101000 (base 2)

and each bit requires a location in D: so in this case 10 locations in D will be required. As a quick, safe guess, dimension D to the number of decimal digits is N times 3⅓, rounded upward. Although that is generous for N=1000, it is exact for N=999.

Merge-Sort.

Here now comes the complete algorithm. To summarize, we are given a linked list of records, with links L, headed by h, and with keys K. The algorithm rearranges the links so the output list, again headed by h, is in sorted order. A directory D, of length d' is required; d' must be no less than the bit length of N, the number of records to be sorted. (One may take d' to be 3⅓ times the number of decimal digits in N, rounded upward.)

Figure 10.

```
1000 'SUBR: MERGE / LINK SORT
1010 'LINKS IN  L  HEADED BY  H
1020 'KEYS IN  K
1030 '
1040 'INITIALIZE DIRECTORY: SIZE D1
1050 FOR I=1 TO D1
1060 LET D(I)=0
1070 NEXT I
1080 '
1090 'MERGING STRATEGY
1100 IF H=0 GOTO 1240   'INPUT LIST EXHAUSTED
1110 GOTO 1370   'NEXT SORTED SUBLIST?
1120 FOR I=1 TO D1
1130 LET M=INT(M/2)
1140 IF M=0 GOTO 1160
1150 NEXT I
1160 IF D(I)<>0 GOTO 1190   'COLLISION
1170 LET D(I)=H1
1180 GOTO 1100   'GET NEXT SORTED SUBLIST
1190 LET H2=D(I)
1200 LET D(I)=0
1210 GOSUB 1490   'MERGE SUBLISTS
1220 LET I=I+1
1230 GOTO 1160   'RE-ATTEMPT INSERTION IN  D
1240 'ALL SORTED SUBLISTS ARE NOW IN  D
1250 FOR I=1 TO D1
1260 IF D(I)<>0 GOTO 1280
1270 NEXT I
1280 LET H1=D(I)
1290 IF I=D1 GOTO 1350
1300 FOR L=I+1 TO D1
1310 LET H2=D(L)
1320 IF H2=0 GOTO 1340
```

```
ALGORITHM MERGSORT
For i=1 to d' DO D(i) ← 0 Next i
While h>0 DO
  FIND(h,h',m)
  i ← 0
  While m>0 DO
    i ← i+1
    m ← [m/2]
  Endwhile
  While D(i) ≠ 0 DO
    h" ← D(i)
    D(i) ← 0
    MERGE(h',h")
    i ← i+1
  Endwhile
  D(i) ← h'
Endwhile
i ← min(j such that D(j)>0)
h' ← D(i)
For I = i+1 to d' DO
  h" ← D(I)
  If h">0 then MERGE (h',h")
Next I
h ← h'
EXIT                                  □
```

```
1330 GOSUB 1490    'MERGE SUBLISTS
1340 NEXT L
1350 LET H=H1
1360 RETURN
1370 'FIND NEXT SORTED SUBLIST IN INPUT
1380 LET M=0
1390 LET H1=H
1400 LET T=H
1410 LET M=M+1
1420 LET H=L(H)
1430 IF H=0 GOTO 1120
1440 IF K(T) <= K(H) GOTO 1400
1470 LET L(T)=0
1480 GOTO 1120
1490 'MERGE LISTS HEADED BY  H1  AND  H2
1500 LET T=0
1510 IF K(H1)<=K(H2) GOTO 1550
1520 LET H0=H1
1530 LET H1=H2
1540 LET H2=H0
1550 LET L(T)=H1
1560 LET T=H1
1570 LET H1=L(H1)
1580 IF H1<>0 GOTO 1510
1590 LET L(T)=H2
1600 LET H1=L(0)
1610 RETURN
```

Upper and Lower Case for Your Model I

David Yon

You've just received your TRS-80 with its new lower-case modification. You decided to go for it and get the good one from Radio Shack with the true descenders. You plug everything back in and decide to experiment with your new capability before doing anything serious.

Just load the lower-case driver in (what a strange filename), and you're in business. Look at those letters in lower case. How did you ever get along without it? All is well, correct? Maybe not.

David Yon, RD #1-168, Richmond, VT 05477.

While the lower-case kit on the whole is excellent (though a bit over-priced), the driver that Radio Shack supplies has some inherent problems. First, it eats up several hundred bytes of memory which means that if you want to type in an adventure or other long program using lower case, there may not be room.

Also, what about shifting between all-caps and the upper/lower case mode? I find it extremely awkward to hit the shift and "0" keys at the same time. These are but a few of the discoveries that you make the first time you load this little utility.

You shrug off the disadvantages and decide to type in a program from *Creative Computing*. So you load the driver, then TSHORT, and get down to business. Wait a minute. Why don't the keywords appear when you type a shift-letter, even when you're in the all-caps mode? This new driver seems to be shutting off TSHORT. So you load TSHORT without the driver.

But wait. You wanted this program in lower case. So you just have to type in the

program without using TSHORT, but then you've wasted money on an extremely helpful utility that you can't use any more.

A close look reveals that the driver has also shut off the control characters available with the "new-ROM" Level-IIs (shift-down arrow-C enters a control-C from the keyboard.)

Well, now you can forget about that uncooperative driver. The driver that frustration forced me to write eliminates all of those problems.

First, it takes no memory (that's right, it resides in low memory below your program space), which means that you have given up none of your precious memory space, and also means that it will run on any size system, won't clash with any high-memory printer drivers, etc., and won't interfere with other programs such as TSHORT.

Second, it toggles the shift-lock with the shift-right arrow, which is much easier on the fingers and prevents accidentally going into double width. (However, it does mean you will lose the TAB keyword from TSHORT). Also you can still run TSHORT, and turn it off and on with a single keystroke. Lastly, any other special feature that your particular keyboard has will still be there.

To use the driver, merely type and assemble the source code, and put it on tape. Now, everytime you want to power-up in Basic, simply make it a habit to load the lower-case driver first.

All you have to do is type SYSTEM, then the filename that you specified when you assembled the driver. When the driver is loaded, simply hit Break to enter Basic. When you first load the driver, you are locked into all-caps (actually, all-caps is the normal mode for the TRS-80 keyboard, where you press shift to get lower-case). To shift between all-caps and upper/lower (typewriter) keyboard, hit shift-right arrow. Notice that you don't enter double-width, a feature I consider more of a blessing than a fault. Notice, too, that if you have the new ROM, you will still be able to enter control characters.

I seriously doubt that the driver will work on disk as-is, you will probably have to relocate it somewhere else if you want to use it with a disk system.

Things become a bit more complicated if you want to use TSHORT with the driver, however. To make it easier, I put a copy of TSHORT directly after a copy of the driver (an example of a legal use for System Copy).

To power-up with lower case and TSHORT, just type SYSTEM and the filename of the driver, and let the driver load. After it loads, type TS and left TSHORT load. Now start TSHORT as before.

Notice that TSHORT functions exactly as before when first loaded. However, when you hit the shift-right arrow, you enter the lower-case keyboard mode, and TSHORT is effectively shut off. To return to the TSHORT keyboard, hit the toggling keystroke again; you can have lower case and TSHORT runninbg at the same time, and toggle between them with a single keystroke. Note that you must load the driver first, or TSHORT will not work. As a matter of fact, if you load anything that taps into the keyboard or video routines in the ROM, you should load the driver first if you plan to use it.

The Program

Referring to the listing, lines 120-140 contain the pointers to direct Basic to our routine, they load directly into memory with the rest of the program.

Lines 160-280 contain the video routine which is almost identical to the one found in ROM, except that it doesn't test for and screen out lower-case characters, as it displays any size letter.

The remaining lines function as follows:

Line 290 calls the ROM keyboard routine.

Line 300 tests for no keystroke.

Line 320 loads HL with the location of the flag byte (0=shift lock, 255=typewriter).

Line 330 tests for the shift-right arrow, and activates the shift-lock.

Line 360 gets the flag byte, tests for shift-lock, and returns the keystroke "as-is."

If the program gets as far as lines 400 to 430 these lines check to see if we are dealing with a letter.

If we have a letter, line 440 will change an upper-case letter to lower-case and vice versa.

The routine beginning with line 460, the shift-lock routine, toggles the flag byte between 0 (all-caps) and 255 (upper/lower case).

There are several possibilities for indirect control over my routine. First, you can change the shift-lock keystroke by changing the value in line 330 to the value of your new keystroke. Or, to allow program control over the mode the keyboard is in, change memory location 16408 to 0 for upper case only (for answering "Y" or "N" for example), and to 255 for the typewriter mode (for typing in a person's name, etc.).

```
                00100 ;LOWER CASE DRIVER 2.0
4016            00110          ORG     4016H
4016 5C40       00120          DEFW    KYBD        ;KEYBOARD ROUTINE
0006            00130          DEFS    6
401E 3E40       00140          DEFW    VIDEO       ;VIDEO ROUTINE
403E            00150          ORG     403EH
403E DD6E03     00160 VIDEO    LD      L,(IX+3)    ;VIDEO DRIVER*****
4041 DD6604     00170          LD      H,(IX+4)    ;CURSOR POS TO HL
4044 DA9A04     00180          JP      C,49AH      ;HOUSEKEEP
4047 DD7E05     00190          LD      A,(IX+5)    ;GET CURSOR CHAR
404A B7         00200          OR      A           ;TEST FOR NULL
404B 2801       00210          JR      Z,V1        ;GO IF NULL
404D 77         00220          LD      (HL),A      ;PUT CHAR IN TI'S PLACE
404E 79         00230 V1       LD      A,C         ;GET NEXT CHAR
404F FE20       00240          CP      20H         ;CNTRL?
4051 DA0605     00250          JP      C,506H      ;CNTRL ROUTINE
4054 FEC0       00260          CP      0C0H        ;SPACE COMPRESSION?
4056 D2AA04     00270          JP      NC,4AAH     ;IF SO.....
4059 C37D04     00280          JP      47DH        ;PRINT IT
405C CDE303     00290 KYBD     CALL    03E3H       ;KEYBOARD DRIVER*****
405F B7         00300          OR      A           ;TEST FOR NULL
4060 C8         00310          RET     Z           ;NO KEYSTROKE
4061 211040     00320          LD      HL,4010H    ;FLAG BYTE POINTER
4064 FE19       00330          CP      25          ;SHIFT LOCK?
4066 280F       00340          JR      Z,K1        ;LOCK ROUTINE
4068 47         00350          LD      B,A         ;SAVE
4069 7E         00360          LD      A,(HL)      ;LOWERCASE FLAG BYTE
406A B7         00370          OR      A           ;TEST FOR LOWERCASE
406B 78         00380          LD      A,B         ;RESTORE BYTE
406C C8         00390          RET     Z           ;LEAVE BYTE ALONE
406D FE60       00400          CP      96          ;LOWERCASE?
406F 3003       00410          JR      NC,K2       ;YES.....
4071 FE40       00420          CP      64          ;LETTER?
4073 D8         00430          RET     C           ;RETURN IF NO)
4074 EE20       00440 K2       XOR     20H         ;COMPENSATE
4076 C9         00450          RET                 ;RETURN
4077 3EFF       00460 K1       LD      A,255       ;INVERTER BYTE
4079 AE         00470          XOR     (HL)        ;INVERT IT
407A 77         00480          LD      (HL),A      ;PUT BACK
407B AF         00490          XOR     A           ;ZERO
407C C9         00500          RET                 ;RETURN
06CC            00510          END     06CCH
00000 TOTAL ERRORS
K2      4074
K1      4077
V1      404E
VIDEO   403E
KYBD    405C
```

To lock the keyboard into either mode, set 16408 (as described above), then set 16485 to 0. To allow the user to control the mode again, return 16485 to 25. As a final short-cut, to lock the computer into lower case without changing 16485 and doing two POKEs, set 16408 to any number between 254 and 1. □

Unlistable Lines for the TRS-80

Paul Tiernan

The TRS-80 is a marvelous machine which is capable of much more than the Level II Reference Manual would have you believe. Without leaving Basic, it is possible to write a program which will do much more than its listing suggests.

For example: A program when LISTed gives:

```
10 PRINT "JUST A SIMPLE ONE LINER."
```

But when it's RUN

```
THIS IS DEFINITELY NOT
JUST A SIMPLE ONE LINER
```

Or another program LISTs as:

```
10 PRINT "I AM THE WALRUS"
20 PRINT "GOO GOO G'JOOB"
```

and runs as

```
I AM THE WALRUS
GOO GOO G'JOOB
GOO GOO G'JOOB
GOO GOO G'JOOB
GOO GOO G'JOOB
```

As you have probably deduced, there are lines in both runs which do not appear in a listing but are still included in the run. In the first case it is a print statement and in the second it is a FOR-NEXT loop which is "unlistable." In fact almost any line under 115 characters, not including line numbers, can be made unlistable.

Paul Tiernan, 117 Military Rd., Henley South, South Australia 5022.

But first let's try an example. Type exactly what is written—don't insert or delete spaces or change spellings of words or anything—there will be plenty of time to be creative later.

Type:

```
10 PRINT "HOW IS THIS PRINTED?": REM
THESE WORDS ARE GOING TO DIS-APPEAR
```

and ENTER it

Now type: Edit 10 and ENTER. Type 32 then the space bar, then 36C and keep pressing the backspace key until the cursor does not move any further. Now press ENTER. RUN it. LIST it. Amazing isn't it. When you list your program you can't see anything, but when you run it

```
HOW IS THIS PRINTED
```

is displayed. You may have noticed that when you type LIST the line flashes on and then off. Don't worry; this is not noticeable in a long program listing and not, in any case, to the uninitiated!

To generate your own invisible line:

1) Turn the computer on and make sure it is running properly by typing ?MEM and ENTER. The computer should return with 3284 (for 4K) or 15572 (for 16K).

2) Type in and ENTER the line that you want to make unlistable, (it must have fewer than 115 characters).

3) LIST the line. Make sure that there is nothing that you may want to change (If there is, do it and return here afterwards).

4) Count the number of characters in the line (including spaces, line numbers, etc.).

5) Type EDIT followed by the number of your line, and ENTER. This puts you into the Insert mode.

6) Press X (and don't ENTER). This brings the cursor to the end of the line and puts you into the Insert mode.

7) Type:

```
:REM
```

Then type the same number of characters as there are characters in the line, plus five. (It doesn't matter what they are; they're going to disappear anyway.)

8) Press the SHIFT and up arrow keys simultaneously.

9) Type in the number corresponding to the previous number of characters in the line plus five, followed by a backspace. (For example, if you had counted 31 characters in step 4, you would type 36 and backspace). This should place the

cursor directly after the M in REM. (If it doesn't, type X and go back to step 8).

10) Again type the number that you counted in step 4, plus five, this time followed by a C (Do not ENTER).

11) Keep pressing the backspace key until the cursor doesn't move any further.

12) Now press ENTER. You can RUN or LIST your program as much as you like.

How It Works

The principle is very simple. When you RUN the program, the computer executes the first part of the line, and when it reaches the REM statement it disregards the rest—which consists of backspaces. When you LIST it the entire line is displayed, but, because of the backspaces after the REM statement, the line is printed over. This happens so quickly that it appears that there is no line there.

There are quite a few things that this idea could be used for, apart from puzzling your friends who understand Basic. One thought is that answers in a quiz program could be hidden, so that cheating would become almost impossible. (If you choose to use this idea, use line numbers ending in 3, 7 or 9 to prevent accidental EDITing by unwitting students). Another use is that in programs for publication, you can sneak your name in, to stop pirating of your ideas. This is by no means a comprehensive list of uses—just a few ideas to stimulate your imagination. □

In TRS-80 Basic . . .

Fast Changing Screen Display

Michael Flanagan

When your Basic program uses the same display several times, you can waste a lot of time waiting for Basic's slow graphic routines to redraw the display. Here is a machine language subroutine that allows you to store your display in high memory on a 16K TRS-80 and almost instantly transfer it to the screen with a USR call.

There are two distinct methods available. You can use the Editor/Assembler, or you can POKE in the machine language from Basic. To use the Editor/Assembler:

1. Load EDITOR/ASSEMBLER.
2. Type in assembler program as in Listing 1.
3. Assemble program and create object tape.
4. Turn computer off then on again using 30975 for memory size.
5. Type SYSTEM and load the object tape.
6. Load the program to be modified and add subroutines 3 and 4.

To POKE the routine in from Basic:

1. Turn on the computer using 30975 for memory size.
2. Load the program to be modified and add subroutines 2, 3 and 4.

In either case subroutine 3 is executed just once, when the screen to be saved is visible on the monitor. The screen takes less than a minute to move, and once it's done, subroutine 4 may be called as often as necessary to redisplay that screen "instantly."

The first method assumes you have an editor/assembler, a desire to learn, and much patience. There is one advantage to offset the extra steps involved: this is the ability to make changes in the machine language subroutine.

The second method is faster, both initially and upon every loading of the program thereafter, as the object program does not have to be loaded from tape, since it is created by the Basic program itself (Listing 2).

With either method you must POKE 16526,0 and POKE 16527,121 before calling subroutine 4. This simply tells the computer where to branch to (in decimal) when it encounters the USR statement.

The subroutines are written for the TRS-80 Level II, 16K, and only work in a tape-based machine. For disk machines, you must relocate the routine to the top of 32K or 48K and use DEFUSR instead of POKE 16526,0 and POKE 16527,121. Other Z-80 machines can use similar block move routines, with the address changed and different patches to Basic.

Michael Flanagan, 8700 Outlook Dr., Brooklyn, OH 44144.

Listing 1.

```
100             ORG   7900H
110   VIDEO     EQU   3C00H
120   MAP       EQU   7A00H
130             LD    HL,MAP
140             LD    DE,VIDEO
150             LD    BC,400H
160             LDIR
170             RET
180             END
```

Listing 2.

```
1100  REM  CREATE MACHINE LANGUAGE
      SUBROUTINE
1110  POKE 30976,33: POKE 30977,0:
      POKE 30978,122: POKE 30979,1
     7
1120  POKE 30980,0: POKE 30981,60:
      POKE 30982,1: POKE 03983,0
1130  POKE 30984,4: POKE 30985,237
     : POKE 30986,176: POKE 30987,
     201
1140  RETURN
```

Listing 3.

```
1200 REM  SAVE SCREEN
1210 I1=15872
1220 FOR I=15360 TO 16383
1230 P= PEEK (I)
1240 POKE I+I1,P
1250 NEXT I
1260 RETURN
```

Listing 4.

```
>PR#0
>LIST
 1300 REM  REDRAW SAVED SCREEN
 1310 N=0:X=USR(N)
 1320 RETURN
```

Secrets of Where and How Variables Are Stored

Inside the TRS-80

Curtis F. Gerald

One important question asked by programmers and experimenters is how variables are stored in a system. For example, IBM 370 computers gain their power in part from the rich variety of storage formats afforded the user. Numerical quantities can be handled in any of seven different forms and there are specialized machine instructions for most of them. In a computer system based on microprocessors much less sophistication is built into the hardware, but the software can provide the needed flexibility. The TRS-80 system is typical.

Curtis Gerald, 980 West St., San Luis, Obispo, CA 93401.

In Radio Shack's computer there are four types of variables: integers, single precision, double precision and character strings. The user designates which type of variable is to be employed by appending a type designator to the one or two characters that form the name as shown in Table 1.

When BASIC accesses the numeric variables named in the statements of the program, it searches for them in a list that includes the name, the value and a numeric code that indicates the type. The numeric type indicators also show how many bytes are used to store the values. For strings, the indicator does not give the same information. In this case, the number of bytes stored with the name is still equal to the type code three, but the actual value is stored elsewhere. The three bytes stored with the name comprise a two byte address pointer that shows where in memory the characters are located and one byte that gives the number of characters in the string.

Because the BASIC interpreter always scans the list of variables from its head in order to access any of the values or to update a value that is

stored, the speed of execution can be increased by causing the more frequently used variables to be stored near the head of the list. This is not hard to do; it can be done by merely using the variables in statements that are executed early. It may be an advantage then to use these variables, even in an artificial way, so they get placed at the beginning of the list.

Arrays (subscripted variables) are stored in a separate list of variables. All the values that share a common name are stored together. There is some economy of memory space and a possible speed advantage when quantities are stored in an array. All four types of variables may be assigned to arrays in TRS-80 Level II.

Radio Shack's Level II manual gives some information about the storage formats for variables but it is incomplete. One purpose of this article is to supplement that information and also to disclose how pointers are maintained to locate the lists of variables. A second purpose is to provide a program that will list all the variables used by a program in the order in which they are stored.

Integer Variables

The storage format for integer variables is easiest to describe. A total of five bytes is used, as illustrated in Figure 1a. The first byte is always 02, the numeric code that designates an integer variable. The next two bytes give the ASCII equivalent of the characters of the name with the second character preceding the first. If the name is only a single letter a zero byte is inserted where the second character would normally appear. The last two bytes give the hexadecimal digits of the value with the less significant byte preceding the more significant. If the quantity is negative the value is stored as two's compliment. Because only two bytes are provided to store the value, the range of values for integer variables is confined to -32,768 to +32,767.

Two's compliment representation is used almost universally in computers to store negative integers. It offers the major advantage that subtraction can be performed in the same electronic circuitry that is used for addition. To form the two's compliment of a number, its magnitude in 16-bit binary form is first complimented (this gives the one's compliment) then one is added. For example, to represent the decimal value -23, we proceed as follows:

1. Write 23 as a binary number: 0000 0000 0001 01111 (00 17 hex)
2. Compliment each bit: 1111 1111 1110 1000 (FF E8 hex)
3. Add one: 1111 1111 1110 1001 (FF E9 hex)

If the example in Figure 1a were for K% = -23, the fourth and fifth bytes would be (in hexadecimal) E9 FF, or (in decimal) 233 255. (One needs to consider the decimal values in addition to the hexadecimal because the response to the PEEK command in TRS-80 displays the memory contents in decimal form).

The TRS-80 affords the user a special command to locate where variables are stored in memory. The instruction PRINT VARPTR(K%) will display the decimal value of the memory address where the value is stored. Note that to find the name of the variable one must PEEK in the two bytes preceding VARPTR(K%), and, to find the code that designates the type, one PEEKs in the third byte preceding.

Type	Type Designator	Example of Name	Bytes for Value	Numeric Type Indicator
Integer	%	P%	2	02
Single Precision*	!	XJ!	4	04
Double Precision	#	K3#	8	08
String	$	AA$	0 to 255	03

* If no designator is specified, the type is single precision by default.

Table 1

Floating Point Variables

Figures 1b and 1c show the storage formats for single and double precision variables. These quantities are in floating point representation, a form closely related to scientific notation in which the value is represented as a fraction (often called the mantissa) that is to be multiplied by a scale factor. The scale factor is some base value raised to an integer power. In computers, two is the most common base value, but sometimes other bases are used. In particular, IBM 370 systems use sixteen for the base. Since the base value for the scale factor is always two, only the exponent needs to be stored.

Floating point numbers need, in addition to the fraction part and the exponent, two additional pieces of information. The sign of the number must be represented and also the sign of the exponent. The TRS-80 system uses special ways to record these two signs. For the exponent a biasing scheme is used. The exponent part of a floating point number is stored in one byte, affording eight bits for the value. If only positive exponents were involved, a range from 0 to 255 could be accomodated but, because negative values are also required, this is reduced to 0 to 127 (or 0 to 7F in hexadecimal). To represent both positive and negative numbers a bias value of 128 (equal to 80 in hexadecimal) is added to the binary value of the exponent. The result is that decimal values of the biased number from 0 to 128 (0 to 80 in hexadecimal) represent negative exponents (for the base of 2) from -128 to 0, while decimal values from 128 to 255 (80 to FF in hexadecimal) represent positive exponents from 0 to 127.

The sign of the number (which we may think of as the sign of the fraction) is represented in the TRS-80 in a very ingenious way. Normally one

Byte	Contents Hexadecimal	Decimal	Significance	
1	02	2	Numeric type indicator	
2	00	0	ASCII for name	
3	4A	74		
4	17	23	Least significant	Value bytes
5	00	0	Most significant	

(a) Storage format for an integer variable, corresponding to J% = 23.

1	04	4	Numeric type indicator	
2	33	51	Second character	of name
3	58	88	First character	
4	00	0	Fraction, least to	
5	00	0	most significant bytes	
6	20	32		
7	82	130	Exponent, biased by 128 (80 hex)	

(b) Storage format for a single precision variable, corresponding to X3 = 2.5 (binary 0.1010×2^2)

1	08	8	Numeric type indicator	
2	5A	90	Second character	of name
3	5A	90	First character	
4	00	0		
5	00	0		
6	00	0	Fraction, least to	
7	00	0	most significant bytes	
8	00	0		
9	00	0		
10	C8	200		
11	83	131	Exponent, biased by 128	

(c) Storage format for a double precision variable, corresponding to ZZ# = −6.25 (binary -0.11001×2^3)

1	03	3	Numeric type indicator	
2	00	0	Second character	of name
3	53	83	First character	
4	04	4	Length of string	
5	15	21	Low byte	of address
6	43	67	High byte	

(d) Storage format for a string variable, corresponding to S$ = "ABCD"

1	04	4	Numeric type indicator	
2	35	53	Second character	of name
3	41	65	First character	
4	35	53	Less significant byte	distance parameter
5	00	0	More significant byte	
6	02	2	Number of subscripts	
7	04	4	Less significant byte	limit of second subscript
8	00	0	More significant byte	
9	03	3	Less significant byte	limit of first subscript
10	00	0	More significant byte	
11 through 58			Bytes that represent the values for 12 members of the array	

(e) Storage format for the header of an array, corresponding to the statement DIM A5(2,3)

Figure 1

would expect that one of the bits would be required to store the sign information. However, advantage is taken of the fact that the fraction part is always normalized. "Normalizing" means that the scale factor is adjusted so that the leading bit of the fraction is always non-zero. For example, the binary fraction 0.00011 (equal to 3/32) can be normalized by rewriting it as 0.11 x 2-3. If the number has leading ones before the binary point (as in 10.1, equal to 2 1/2), shifting the binary point to the left with a corresponding adjustment of the scale factor gives 0.101 x 22 as the normalized form.

If the fraction part is always normalized, the first bit of information in the fraction is really redundant; we know without looking that it is a one. In the TRS-80 BASIC language this first bit is left a one for a negative fraction but is reset to a zero if the fraction is positive. This provides one more bit of precision in the fraction at the expense of slightly more complex software that performs the arithmetic operations on floating point numbers.

In summary, floating point quantities are stored with one byte representing the biased exponent for the base of two, giving the scale factor. Three bytes (for single precision) or seven bytes (for double precision) are used to hold the normalized binary value of the fraction, with the first bit reset to zero if the number is positive. Some examples are shown in Figure 2.

There is one final point needed to complete the description of floating point representation. Zero is a special case because its fraction part cannot be normalized. It is conventional, and TRS-80 abides by the convention, to store zero as all zeros in both the fraction and the exponent.

As shown in Figures 1b and 1c, single and double precision numbers differ only in the number of bytes used to store the fraction. Since only one byte is used for the exponent in either case, the range is approximately the same, from -1.7 x 1038 to +1.7 x 1038 as decimal equivalents. The smallest non-zero magnitudes are +1.47 x 10-39. For single precision numbers the precision is equivalent to about 7 decimal digits, while double precision is equivalent to about 17 decimal digits. (When functions are generated, somewhat less precision is given in most cases, due

to inaccuracies in the computational routines.)

String Variables

After the rather involved way in which floating point quantities are stored, it is not hard to describe how string variables are stored among the list of variables. As shown in Figure 1d, a one byte type code (03) is followed by two bytes for the name, then a one byte length value and finally a two byte address. Since only one byte is allowed for the length, string variables are limited to a maximum of 255 characters. The address bytes (stored as low address followed by the high address byte) point to the location in memory where the defining characters are located. If the definition is an assignment statement or a DATA statement within the BASIC program, the pointer points to that location within the program. If the value of the string is input from the keyboard, the characters are stored in the special reserved area allocated to strings (at the high end of memory just ahead of the space reserved for machine language programs).

Note how the numeric type indicator relates to the number of bytes used for storage of each kind of variable. In each case it is exactly equal to the number of bytes associated with the name. For numeric quantities, these bytes hold the value directly. For a string variable, it equals the bytes used for the length parameter and the address pointer. Considering that there are additional bytes to hold the name and the type designator itself, there are always T + 3 bytes used for each variable where T is the numeric type code.

With subscripted variables (arrays) space is allocated separately from that for the simple variables and this space immediately follows that used by the simple variables. All the values for the members of each array are stored in a group, one after another in order of ascending subscript value. (When more than one subscript is used, the first subscript goes through its entire range before the next subscript is incremented.) At the beginning of each set of array values, a header is stored. Figure 1e gives the contents of each byte of the header.

The first byte in memory for an array is the type indicator, and the next two bytes hold the ASCII values for the name, exactly as for simple variables. The next two bytes give the distance to the next array variable, with the less significant byte coming first. How this distance parameter is determined is shown later. Now comes a single byte that gives the number of subscripts associated with this variable. (In theory, one could use 255 subscripts).

For each of the subscripts, there is a two byte count of the size limit for the subscript, ordered from the last subscript to the first. (This would permit each subscript to be dimensioned to a size of over 1,000,000 if there was memory space to hold them). Note that the size associated with each subscript is one more than the value specified in the DIM statement because the number zero is a valid subscript. The dimension statement DIM X%(1,3,2) would have 2,4 and 3 for the subscript limit values.

The distance measure stored in bytes 4 and 5 of the header can be computed as follows:

Dist = :D1 + 1)(D2 + 1)(...)
(DN + 1)(T) + (2)(N) + 1
where D1, D2, ..., DN are the subscript sizes as specified in DIM statement. T is the numeric type code. N is the number of subscripts.

In the above formula, the factor (D1 + 1) (D2 + 1) (...) (DN + 1) gives the number of members of the array. The formula provides for T bytes for each member, plus two bytes for each subscript to hold its limit value plus one for the number of subscripts. For the example in Figure 1e, this distance is (2 + 1) (3 + 1) (4) + (2) (2) + 1 = 53. It is measured from the sixth byte of the header. One can determine the total memory space used by any array by adding six to its distance parameter.

The headers for all four types of arrays follow this same pattern. The number of bytes associated with the values for each of the members does differ, of course.

You may not be surprised to learn

Determine the storage format of 4.25:
Convert to binary 100.01
Normalize + 0.10001 x 2^3
Bias the exponent 3 + 128 = 131 (83 hex)
Reset first bit because positive .00001 (08 00 00 as 3 hex bytes)
Storage form 08 00 00 83 ; 80 00 01 31
fract expt fract expt
hex decimal

Determine the storage form of -0.125:
Convert to binary -0.001
Normalize -0.1000 x 2-2
Bias the exponent -2 + 128 = 126 (7E hex)
Leave first bit alone because negative .1000 (80 00 00 as 3 hex bytes)
Storage form 80 00 00 7E ; 12 8 0 0 12 6
fract expt fract expt
hex decimal

If the storage form is 20 00 00 82 (hex), what is the value?
Write fract in binary .0010 0000
Number is positive so set first bit to one .1010 0000
Determine exponent 82 - 80 = 2 (hex) (130 - 128 = 2)
Shift binary point + 10.10 0000
Convert to decimal + 2.5 is the value

If the storage form is Co 00 00 7F (hex), what is the value?
Write fract in binary .1100 0000
Number is negative, leave first bit a one .1100 0000
Determine exponent 7F - 80 = -1 (hex) (127 - 128 = -1)
Shift binary point -0.0110 0000
Convert to decimal -(3/8) = -0.375

Figure 2. Examples of storage format for floating point values.

that the system keeps pointers associated with the storage of values for variables. There are four of them. One, at locations 16633/4 (40F9/A in hex), points to the memory location where the first variable begins. This happens to be three bytes beyond the last text byte of the program itself. A second pointer, at locations 16635/6 (40FB/C) points to the beginning of the array variables, while a third, located at 16637/8 (40FD/E) points to the beginning of the array variables, while a third, located at 16637/8 (40FD/E) points to the first byte after the end of the last array variable. A dynamic pointer, kept at 16607/8 (40DF/E0) is continually updated during the execution of the program to show the location of the last variable accessed.

To some degree, the array beginning and ending pointers are also dynamic. Even though an array is specified in a program (either through an explicit DIM statement or implicitly by using a subscripted variable that invokes an automatic DIM(10)), no space is assigned until the DIM statement is encountered during execution. Consequently, before a RUN command is issued, the two array limit pointers point to the same location as the beginning-of-variables pointer. As simple and array variables are encountered, these array limit pointers are continually changed so they point to higher and higher memory locations.

Variables Lister Program

Figure 3 lists a program that displays the name and type for each simple variable, followed by the name, type and dimension sizes for each array. The dimension size values are those that would be specified in a DIM statement. The user can then display the current values for each variable by a succession of PRINT commands. (A routine was written to display these values automatically, but when finished it was found to be so slow and so long that it was judged not useful).

The strategy behind the Variables Lister Program is straightforward. Beginning at the location of the first variable, the name and type are picked out and displayed. The program advances to the start of the next variable (the displacement is just the type number plus three) and the process is repeated. Displaying of simple variables is terminated when the program finds the first variable within itself, which is VL. Because of this, that variable name should not be used in the program whose variables are being listed. (By modifying lines 2050 and 2520 to remove the IF conditionals, the routine can be made to include its own variables in the listing).

After encountering its own first variable name, the routine moves to a listing of array variables. These are each displayed in the order in which they occur. Advancing from one array variable to the next is made easy by the distance parameter in bytes 4 and 5 of the header. The end of arrays pointer shows when all the array variables have been scanned.

Using the lister program is probably best done by calling it as a subroutine. At any point in a program where it is desired to list all variables that have been used so far, or defined, a call to the subroutine is inserted. The routine can be made into a standalone program by changing the RETURN in line 2180 to STOP.

Figure 4 is a test program together with the output from the lister program. □

```
2000 PRINT "LIST OF SIMPLE VARIABLES"
2010 PRINT "NAME", "TYPE"
2020 VL%=0: VS%=0: VP%=16633
2030 VA%=PEEK(VP%)+PEEK(VP%+1)*256
2040 VP%=PEEK(16635)+PEEK(16636)*256
2050 GOSUB 2500: IF VN$="VL" THEN 2080
2060 PRINT: VA%=VA%+VT%+3
2070 IF VA%<VP% GOTO 2040
2080 VA%=VP%: VP%=PEEK(16637)+PEEK(16638)*256
2090 IF VA%=>VP% RETURN
2100 PRINT: PRINT: PRINT "LIST OF ARRAYS"
2110 PRINT "NAME","TYPE","DIM(S)"
2120 GOSUB 2500
2130 VS%=PEEK(VA%+5)
2140 FOR VL%=1 TO VS%
2150   PRINT PEEK(2*(VS%-VL%)+VA%+6)+PEEK(2*(VS%-VL%)+VA%+7)*256-1;
2160 NEXT VL%: PRINT
2170 VA%=VA%+PEEK(VA%+3)+PEEK(VA%+4)*256*5
2180 IF VA%<VP% THEN 2120 ELSE RETURN
2500 VT%=PEEK(VA%)
2510 VN$=CHR$(PEEK(VA%+2))+CHR$(PEEK(VA%+1))
2520 IF VN$="VL" THEN RETURN ELSE PRINT VN$,
2530 IF VT%=2 PRINT "INT",: RETURN
2540 IF VT%=3 PRINT "STR": RETURN
2550 IF VT%=4 PRINT "SNG": RETURN
2560 IF VT%=8 PRINT "DBL": RETURN
2570 PRINT "ERROR. VARIABLE ";VN$;" HAS TYPE NUMBER OF";VT%: STOP
```

Figure 3. A Variables Lister Program

Variables Used —

- VL% for loop control
- VS% number of subscripts in an array
- VP% utility pointer
- VA% beginning address of variable currently being processed
- VN$ name of the variable being processed

```
10 REM A TEST PROGRAM FOR VARIABLES LISTER
20 AA=2.22
30 K%=1234
40 C3$="AAAAAA"
50 DIM XX(2,3)
60 DIM Y$(5)
70 XX(2,2)=12.34
80 PRINT "READY TO LIST VARIABLES"
90 INPUT "STRIKE ENTER TO LIST THEM";A$
100 GOSUB 2000
110 STOP
Output when RUN:
LIST OF SIMPLE VARIABLES
NAME          TYPE
AA            SNG
K%            INT
C3            STR
LIST OF ARRAYS
NAME          TYPE          DIM(S)
XX            SNG           2 3
Y$            STR           5
BREAK IN 110
```

Figure 4. Test Program and Output

Note: If any key other than ENTER is hit to begin the listing, A$ will be included in the listing.

Catching Those Incorrect Imprints

The Challenge of Error Trapping

Mike K. Summers and John B. Willett

Introduction

In the School of Education at the University of Hong Kong we are designing and evaluating a variety of Computer-Assisted Learning (CAL) packages for use with low-cost microcomputers at school and university level. A typical CAL package consists of a computer program together with a student workbook and a teacher's guide. The teacher's guide should contain full program documentation together with a description of the educational aims of the package and suggestions for integrating the CAL work for the particular topic into the curriculum. Under the general guidance of the teacher, the student uses the microcomputer in an interactive mode to investigate problems posed in the student workbook. It is the interactive nature of the relationship between learner and machine that often creates problems for both the user and the CAL package designer. This is because the packages themselves must be user-proof. When the program requests the student to input information, inappropriate or incorrect responses (for example, typing errors) must be trapped. A trap is simply a program routine which checks the validity of the user's response to the computer's request for input. In the case of inappropriate or incorrect responses the trapping routine will inform the student of the nature of his error and suggest corrective action. In the absence of such a trap it is perfectly possible for an unwitting user to interrupt the program or even delete

Mike Summers, John Willett, School of Education, University of Hong Kong, Hong Kong.

sections of it. This article is concerned with the general problem of trapping. Although the discussion below originates from work on computer-assisted learning, the information is of general interest since nearly all software designed for small computers involves some form of direct user-computer interaction (for example, gaming, simulation, computer art, small business applications and so on). The particular examples of trapping techniques described in this article refer to the Radio Shack TRS-80 microcomputer with Level II BASIC and 16K RAM, since this is the machine for which we are designing CAL materials. However, most of the practical experiences and suggestions are equally applicable to other inexpensive microcomputer systems with extended BASIC interpreters.

The need for traps

When running a CAL program, the student is often required to input data. This data can be either NUMERICAL or ALPHABETIC. Depending on the nature of the data, a different type of input routine must be used. A typical routine asking for numerical input is listed below:

```
 50 REM **** INPUT ROUTINE
    #1 ****
 60 PRINT "WHICH STUDY DO
    YOU REQUIRE. TYPE 1 OR 2"
 70 INPUT X
 80 IF X = 1  THEN 100
 90 IF X = 2  THEN 300
100 REM **** START OF STUDY
    ONE ****
 ,
 ,
 ,
 ,
 ,
 ,
300 REM **** START OF STUDY
    TWO ****
```

Obviously, this type of routine satisfies the essential requirements of a CAL package in which the student is offered a choice of two studies. In practice, however, such a simple routine cannot guard against the possibility of erroneous inputs. For example, if the student responds with any number other than 1 or 2, lines 80 and 90 will fail to detect the input and the program will automatically run to line 100 (the start of study one). If the student inputs alphabetic data, then the BASIC interpreter (which is expecting numerical input) returns its own error message. In the case of the TRS-80, the message REDO? appears on the screen. A further problem arises if the student inputs punctuation marks. With the TRS-80, a semicolon input produces a REDO? message, while commas and colons produce the error message EXTRA IGNORED (after which the program runs on to line 100). The trouble with messages of this kind is that they are either in poor English, or are abbreviations which may be totally incomprehensible to the package user. However, there is a more important reason for designing programs which prevent machine error messages of this kind appearing. The advent of cheap microcomputers with memory-mapped video displays and graphics facilities allows the designer of CAL packages to present information to the student page-by-page. Such a page mode of operation should be contrasted with the Teletype in which output is printed sequentially line by line. In the Teletype mode, the entire display scrolls up when the screen is full, so that information is eventually lost from the top of the screen. A microcomputer with graphics and memory-mapped video used in the paged mode allows very rapid presentation of information (a whole screen of information can be generated almost instantaneously) in a visually captivating form, and is ideal for CAL. An important focus of CAL program design is then to plan whole pages of attractive and interesting information with which the student is to interact. However, if an unforeseen machine error message such as REDO? is generated by the computer during program use, a carefully designed and attractive display can be destroyed. In the case of the TRS-80, REDO? may erase previously generated lines of display. More seriously, REDO? causes the whole page to scroll upwards, so that planned information is premanently lost from the top of the screen. To avoid the problems, the CAL program designer must include trapping routines which not only check the validity of user inputs, but also print comprehensible corrective messages in good English at appropriate places on the screen. In other words, all input errors should be trapped by the CAL program itself so that the BASIC interpreter is never required to display its own error messages.

Alpha and Numeric INPUTS

Consider now an input routine designed for accepting alphabetic data. The routine below, for example, uses alphabetic data to 'turn the pages' of a video display:

```
 50 REM **** INPUT ROUTINE
    #2 ****
 60 PRINT "DO YOU WISH TO
    CONTINUE"
 70 INPUT R$
 80 IF R$ = "YES" THEN 100
 90 IF R$ = "NO" THEN 999
100 REM **** START OF NEXT
    PAGE ****
999 END
```

This routine is similar to the first one in that if the user responds correctly (in this case, with "YES" or "NO") the computer reacts appropriately. Also, if the user inputs any other alphabetic string (for example, "PERHAPS"), neither lines 80 nor 90 recognize R$ and the program automatically runs to line 100 where the page is turned. However, this input routine has one very important advantage over the routine designed to accept numerical input. If the user accidentally inputs numeric data, then the computer reacts in the same way as when an incorrect alphabetic string is input. This illustrates an important assymetry in the computer's handling of input data. In the case of the first input routine, an accidental alphabetical input when the computer is expecting numerical data results in an error message. In the second routine, an accidental numerical input when the computer is expecting alphabetic data produces no machine error message. This is simply a reflection of the fact that it is quite legitimate to store numbers in string form, but it is most definitely not legitimate to store alphabetic inputs as numbers. In both cases the user inputs are invalid, but the second routine does not upset the screen display with the unwanted REDO? error message. This fact is exploited in some of the trapping routines described later. The reaction of the second routine to punctuation mark inputs is similar but not identical to that of the first routine. This time commas, colons and semicolons all allow the program to run to line 100 where the page is turned without user permission. As before, the comma and the colon produce the error message EXTRA IGNORED, but this time the semicolon does not produce the disruptive REDO?.

The above discussion of problems which can arise from erroneous user inputs in interactive programs without traps, used two very simple input routines to provide examples. However, it should be noted that more sophisticated input routines can lead to even more catastrophic results. For example, the sophisticated programmer might decide to delete lines 80 and 90 from the first of the above input

routines and replace them by the single line

```
85 ON X GOTO  100,300
```

However, if the user now responds to the input request with a negative number, the TRS-80 prints the error message ?FC ERROR IN 85, indicating an illegal function call (the ON expression GOTO line number, line number - statement is not valid for negative X) and the computer returns to the command mode. The program run is interrupted and the user confused. But potentially, there is a far greater problem than this. If, as a result of this confusion, the user now inputs a positive number, the corresponding line number will be deleted. Although situations of this kind are unlikely to arise, the CAL program designer must guard against all such eventualities to the best of his ability. This means he must devise as near-perfect trapping routines as possible.

Types of trap

We will distinguish between two types of trap, and discuss them separately. The first is concerned with trapping invalid user responses to the computer's request for alphabetic input, while the second deals with invalid responses to requests for numerical input.

1. Alphabetic input

A modification of INPUT ROUTINE #2 which works reasonably well is given below:

```
50 REM **** INPUT ROUTINE
   #3 ****
60 PRINT "ARE YOU READY
   TO CONTINUE"
70 INPUT R$
80 IF R$ = "YES" THEN 110
90 PRINT "INVALID RE-
   SPONSE. PLEASE RETYPE"
100 GOTO 70
110 REM **** START OF NEXT
   PAGE ****
```

In lines 60 and 70 the computer requests alphabetic input. Line 80 tests for the positive response ("YES") which, if detected, results in a jump to the new page beginning at line 110. All other inputs, including both numbers and letters (or combinations), result in display of the corrective message INVALID RESPONSE. PLEASE RETYPE. Line 100 then causes a jump back to line 70 where the computer again waits for a valid user entry. In the case of the TRS-80, the only way the user can defeat this trap is by responding to the input request with a comma or a colon. The computer then returns the error message EXTRA IGNORED, but program execution proceeds correctly (the INVALID RESPONSE message is displayed). However, a spurious and undesired error message will have been generated and, in the paged mode, this may completely spoil a carefully designed display.

Marginal improvements of the above trap are possible using a combination of the TRS-80 statements PRINT @ **position, item list** and CHR$ (expression), together with judicious use of semicolons. We will first briefly review the function of these two statements and of the semicolon. The PRINT @ **position, item list** statement allows information to be printed starting at any one of 1024 separate locations on the memory-mapped screen. Existing lines can be overwritten, so that a message asking for user input can be replaced by a message indicating on invalid response, at exactly the same screen location. The CHR$ **expression** statement returns a one-character string whose character has the specified decimal ASCII code. For example, CHR$ (65) would return the letter A, since the decimal ASCII code for A is 65. The interesting thing about this statement is that it can also be used with the ASCII codes for control functions. Of particular use is CHR$ (30), where 30 is the ASCII code for the control function which erases to the end of the line (later). A semicolon at the end of a program line indicates to the computer that, when the line has been executed, the cursor should not move to the next line of output display, but should wait at the end of the current line of display. If, for example, the program line prints a request for user input on the screen, the cursor does not move to the next line to accept that input but waits at the end of the sentence requesting input. Let us now see how the above can be combined to produce a better trap:

```
50 REM **** INPUT ROUTINE
   #4 ****
60 PRINT @448, "ARE YOU
   READY TO CONTINUE";
70 INPUT R$
80 IF R$ = "YES" THEN 110
85 PRINT @448, CHR$ (30)
90 PRINT @448, "INVALID
   RESPONSE. RETYPE";
100 GOTO 70
REM **** START OF NEXT
   PAGE ****
```

As before, lines 60 and 70 request alphabetic input. However, this time both the question and user response occur one after the other on the same line (the line beginning at screen location 448) because of the semicolon at the end of line 60. Line 80 detects the positive response ("YES") and causes a jump to the new page starting at line 110. If the user inputs an invalid response, the existing question and the response are erased by line 85 and replaced (beginning at the same screen location i.e. 448) by the INVALID RESPONSE message of line 90. The interesting thing about this trap is that it is almost foolproof. All inputs except commas and colons are trapped as with input routine #3. As before, comma and colon inputs result in the error message EXTRA IGNORED, but with input routine #4 this is immediately erased, and the trap continues to operate correctly (the INVALID RESPONSE message is displayed). We leave it to the more perverse reader to discover why this trap is only **almost** foolproof!

Another interesting trap for either alphabetic or numeric input is one in which the user response is searched (parsed) for particular alphanumeric characters or combinations of characters. Consider the following input routine:

```
50 REM **** INPUT ROUTINE
   #5 ****
60 PRINT "INPUT YOUR
   RESPONSE"
70 INPUT R$
80 FOR N = 1 to LEN (R$)
90 K$ = MID$ (R$, N, 1)
100 IF K$ = "E" THEN 140
110 NEXT N
120 PRINT "INVALID RE-
   SPONSE. RETYPE"
130 GOTO 70
140 REM **** CONTINUATION
   OF PROGRAM ****
```

This routine searches the user input string to see if it contains the letter E in any position. Program execution is only allowed to continue if an E is detected. The core of the trap is contained in lines 80 through 110 which search the input string one letter at a time ooking for the letter E. If this is detected, there is a jump from line 100 to line 140, where the program continues. Use is made of two statements found in many extended BASICs (LEN (string) and MID$ (string, x, y)). Len (string) returns the number of characters in the string in decimal form. MID$ (string, x, y) returns a substring extracted from the specified string. This substring is of length y and is extracted starting at position x (i.e., x characters from the start of the specified string). In the above routine, each character of the string R$ is extracted in turn and

becomes the substring K$. This substring is compared to the letter "E" in line 100, and if equivalence is not detected, the INVALID RESPONSE message is displayed and the program returns to line 70 and awaits a new input. Such a trap is useful in some CAL programs, but more obviously in word, spelling and code-type games.

Several interesting modifications of input routine #5 are possible. One modification involves use of the TRS-80 statement ASC (string), which returns the decimal ASCII code of the first character of the specified string. This, for example, allows replacement of line 100 of input routine #5 by

```
100 IF ASC(K$) = 69 THEN 140
```

since the decimal ASCII code for the letter E is 69. In this particular instance there is no advantage in such a modification, but there are cases where use of ASC (string) can be of great value. One example might be a program in which the user response to the computer's request for input must consist only of alphabetic characters in a particular range (e.g., G through P). This can be achieved by rewriting line 100 as:

```
100 IF ASC (K$) > = 71 AND
    ASC (K$) < = 80 THEN 140
```

Since the decimal ASCII codes of the characters G and P are the numbers 71 and 80 respectively, the trap will only let through combinations of letters in the allowed range.

A final point to note about input routine #5 is that it suffers from the same pitfalls as input routine #3 described earlier. However, use of the modifications included in input routine #4 will render the various forms of input routine #5 virtually foolproof.

2. Numerical input

Earlier we noted an important assymetry in the reaction of the computer to different types of input. An accidental alphabetic input when the computer is expecting numerical input results in display of the machine error message REDO?, but accidental input of numeric data when alphabetic input is expected produces no such message. This assymetry can be usefully exploited when designing effective traps for numerical input. Consider the following:

```
 50 REM **** INPUT ROUTINE
    #6 ****
 60 PRINT "WHICH STUDY DO
    YOU REQUIRE. TYPE 1 OR 2"
 70 INPUT R$
 80 R = VAL (R$)
 90 IF R = 1 THEN 130
100 IF R = 2 THEN 300
110 PRINT "INVALID RE-
    SPONSE. RETYPE"
120 GOTO 70
130 REM **** START OF
    STUDY 1 ****
,
;
,
300 REM *** START OF STUDY
    2 ****
```

In line 70 the computer is expecting a string input, and when a number is input it is stored as the string R$. It is, of course, quite legitimate to store numbers in string form and no machine error message is returned. Likewise, line 70 will accept accidental alphabetic inputs without display of a machine error message, but these will not be allowed through the trap. This is because line 80 extracts the numerical value of the input string using the VAL (string) statement, where VAL (string) automatically returns a numerical value of zero for string characters other than numbers. Lines 90 and 100 detect the allowed inputs 1 or 2, while other inputs result in display of the INVALID RESPONSE message of line 110. A combination of the above input routine with input routine #4 produces a trap for numerical input which is highly user-proof.

Final point

Although none of the traps described in this article are completely user-proof, some are very nearly so. By appropriate combination of the various techniques, the prospective program author should be able to design an effective trap suited to his needs. However, let there be no doubt, the search for the perfect trap goes on! □

It Can Be Done In Basic

Structured Programming Techniques in Basic

Patrick C. Moyer

Regardless of the language being used, the concepts behind structured programming are worth investigating. Perhaps you can improve your programming style with these techniques.

Patrick C. Moyer, 40 Stuyvesant Manor, Geneseo, NY 14454.

Anyone who has read a recent data processing magazine or one of the flurry of articles on Pascal has run across the term "structured programming." Tradltionally, computer programs have been written in a very idiosyncratic style. Structured programming is an attempt to formalize the logic and structure of programs. The intended result being greater programmer productivity and programs which are easier to write, debug, and maintain.

Structured programming has its origins in the concept of modular program construction. Modular programming divides every program into a number of smaller specialized subsections or modules. Each subsection of the program performs a

narrow function: one module would contain input statements; another might contain calculations; another would control output and so on. The activity of these individual modules is coordinated by one module, known as the driver or mainline module. The driver orders and controls the activities of the other modules through an ordered set of call statements (in BASIC; GOSUB).

This structure provides a sharp division of labor among parts of the program. If, for instance you have a problem in output, you need only to look in the output module which is associated with that output. This can save time and effort when trying to debug a very large program. The modular style, also makes writing similar or subsequent programs easier because of the portability of modules from program to program.

In BASIC, each module would consist of either a function or a subroutine. The mainline or driver module would contain an order set of GOSUBS and/or function calls. The decision whether to use function or subroutine is based on how general the application of the module. If the exact activity is used in many different programs it might be wise to use a function: if your module is very specific to one or two programs you'd probably use the subroutine structure.

Another concept which characterizes the structured style is extensive use of internal documentation. Some programming languages such as COBOL and PASCAL are specifically designed to be self-documenting. BASIC, however, was designed to be brief and to the point. Thus a structured BASIC program must employ numerous comments (REM statements) to provide sufficient internal explanation of the program function. At the minimum, it is necessary to give a title and short one sentence description of the program's function. In addition, all modules must be labeled according to function. Figure 1 shows a listing of a structured program with the minimum necessary labeling. It should be noted a REM statement is, also, used to delineate the beginning and end of each subroutine.

The concept which sets structured programming apart from a well-documented modular program is a strict adherence to only three logic constructs. It is the structuralist's contention that any program problem can be solved using only these three logical forms. The most straightforward of these is the sequential processing form. In this form all operations are done from beginning to end in sequence with no external branches or loops. The program in Figure 1 is an example of this logical form.

```
10 REM *PROGRAM: EXAMPLE*
20 REM *THIS PROGRAM SHOWS
   SEQUENTIAL LOGICAL FORM*
30 REM *MAINLINE ROUTINE*
40 GOSUB 100
50 GOSUB 200
60 GOSUB 300
70 GOSUB 400
80 END
90 REM **********************
100 REM *INITIALIZATION ROUTINE*
110 LET A=0
120 LET B=0
130 LET C=0
140 RETURN
150 REM **********************
200 REM *INPUT ROUTINE*
210 INPUT A
220 INPUT B
230 RETURN
240 REM **********************
300 REM *CALCULATION ROUTINE*
310 LET C=A*B
320 RETURN
330 REM **********************
400 REM *OUTPUT ROUTINE*
410 PRINT A; "*"; B, "="; C
420 RETURN
430 REM **********************
```

Figure 1. Listing of a structured program.

The second logical form is the IF-THEN-ELSE logic. In structured programming all decisions must be reduced to "yes" or "no" questions. These types of decisions may be strung together to form more complex questions, but the basic decision unit must remain "yes" or "no." Control of the program must return to the next full statement after the IF or IF string. With some versions of BASIC (Applesoft II, PET, TRS-80 Level-I) the IF-THEN construct allows a GOSUB type transfer, but not an ELSE structure. In these cases, we must adapt the IF-THEN structure to allow the IF-THEN-ELSE construction. We construct our IF-THEN so that the line ends with a GOTO. The line number indicated would be the next line after the IF string. The ELSE is inserted by use of a REM statement for documentation purposes. The drawback to this is only executable where multiply statement lines are allowed. Figure 2 shows a listing of a program using this construct.

```
5 REM *PROGRAM: EXAMPLE 2*
10 REM *THIS PROGRAM DEMONSTRATES
   THE USE OF*
15 REM *THE CONSTRUCTED IF-THEN-ELSE*
20 REM *MAINLINE ROUTINE*
30 GOSUB 100
40 IF A=B THEN GOSUB 200: GOTO 90
50 REM *ELSE*
60 IF A>B THEN GOSUB 300: GOTO 90
70 REM *ELSE*
80 GOSUB 400
90 END
95 REM **********************
100 REM *INPUT ROUTINE*
110 INPUT A, B
120 RETURN
130 REM **********************
200 REM *EQUAL PRINT ROUTINE*
210 PRINT "A IS EQUAL TO B"
220 RETURN
230 REM **********************
300 REM *GREATER PRINT ROUTINE*
310 PRINT "A IS GREATER THAN B"
320 RETURN
330 REM **********************
400 REM *LESS PRINT ROUTINE*
410 PRINT "A IS LESS THAN B"
420 RETURN
430 REM **********************
```

Figure 2. Listing of a program using IF-THEN construct.

In other versions, the TRS-80 Level II, for instance, the full IF-THEN-ELSE structure is available, thus coding can be more straightforward. Figure 3, demonstrates this more straightforward approach.

```
5 REM *PROGRAM: EXAMPLE 2*
10 REM *THIS PROGRAM DEMONSTRATES
   THE USE OF*
15 REM *THE NATURAL IF-THEN-ELSE*
20 REM *MAINLINE ROUTINE*
30 GOSUB 100
40 IF A=B THEN GOSUB 200 ELSE IF
   A>B THEN GOSUB 300 ELSE GOSUB 400
50 END
60 REM **********************
70 END
80 REM **********************
100 REM *INPUT ROUTINE*
110 INPUT A, B
120 RETURN
130 REM **********************
200 REM *EQUAL PRINT ROUTINE*
210 PRINT "A IS EQUAL TO B"
220 RETURN
230 REM **********************
300 REM *GREATER PRINT ROUTINE*
310 PRINT "A IS GREATER THAN B"
320 RETURN
330 REM **********************
400 REM *LESS PRINT ROUTINE*
410 PRINT "A IS LESS THAN B"
420 RETURN
430 REM **********************
```

Figure 3. The IF-THEN-ELSE structure.

The third logical form is the DO-WHILE or DO-UNTIL. This form instructs the machine to perform a module until a particular condition is met. In BASIC, we would construct this form from a FOR-NEXT loop. The FOR statement would set up the condition, one or more GOSUB's would indicate what is to be done, and NEXT would indicate the end of the sequence. An example of this construct is shown in Figure 4.

```
5 REM *PROGRAM: EXAMPLE 3*
10 REM *THIS PROGRAM DEMONSTRATES THE*
15 REM *USE OF THE DO-UNTIL CONSTRUCT*
20 REM *MAINLINE ROUTINE*
30 GOSUB 200
40 FOR K=1 TO 10
50 GOSUB 300
60 NEXT K
70 FOR K=1 TO 10
80 GOSUB 400
90 NEXT K
100 GOSUB 500
110 END
120 REM **********************
200 REM *SET-UP ROUTINE*
210 DIM A(10)
220 LET B=0
230 LET K=0
240 RETURN
250 REM **********************
300 REM *INPUT ROUTINE*
310 INPUT A(K)
320 RETURN
330 REM **********************
400 REM *ADDITION ROUTINE*
410 LET B=B+A(K)
420 RETURN
430 REM **********************
500 REM *OUTPUT ROUTINE*
510 PRINT "THE TOTAL IS "; B
520 RETURN
530 REM **********************
```

Figure 4. FOR-NEXT Loop.

In actual practice a structured program will contain a number of these constructs. Notice that there are no loops or GOTO statements in the driver routine. There should be none. It is part of the structuralist's doctrine that these structures are unnecessary and thus illegal. This prohibition on GOTO's would normally extend to all portions of the program, but we must bend this rule in those cases where the IF-THEN-ELSE structure is illegal.

The advantages to this method of programming are apparent. Increased readability and distinct division of function within the program increases the programmer's ability to find and diagnose problems. The modular construction allows portability between programs. Many programs can be built from modules which were originally designed for other applications. Programming can become merely a task of arranging and collecting modules from a standard library of routines.

There are two major disadvantages of this style for the personal user. First, the internal documentation eats up core at an amazing rate. The user must balance this core capacity against his ability to maintain his programs. Given the choice, most users would choose the short run advantage of extra core.

The second drawback is more philosophical. It has been argued that structured programming is too formalized and limits the creativity of the programmer. Many programmers feel stifled by the limited logical forms and wordiness. The philosophers of computer science will be arguing this point for years to come. For now, it is left to the individual user to decide for himself. □

No Frills

Random Access for the TRS-80

James H. Garrett

How many times have you sat down at your computer to master random access, but given up in despair?

I had attempted it at least a dozen times and had given up as many times. But, when one of my business programs, using sequential files returned an "out of memory" error, there was no more putting it off. So, out came the manuals, books on programming, all of the magazine articles I had been saving and anything else I could find on the subject.

Here I was again, totally confused, and wanting to quit, but knowing that this time I could not. Everything seemed either to be written for an expert programmer, using very technical language, or was buried in such a maze of gosubs and inkey strings that solving an Adventure game would be a snap compared to figuring it out.

I was eventually able to put it all together, but it was a slow process. The result is a simple program with no bells or whistles. It was written with one objective: learning to use random access files. When this is accomplished, it will be easy to add refinements such as the INKEY$ function, subroutines, and other programming enhancements or frills.

This article does not propose to answer all questions about random access and may even generate more questions than it answers. Many things are passed over lightly with the thought that the reader who wants more detailed information can get it from other sources. The sole purpose of this article is to explain, in simple language, how to get a random access program up and running.

The first thing that must be done is to OPEN a file for random access, assign a buffer, and give the file a name. I chose the name "FILES," but any name will suffice. In my program this is done eight times, one for each time the files are used for the purposes listed in lines 190-260. Each OPEN statement is the same.

The Field Statement

The next step is writing the FIELD statements. These statements are *not* the same. In fact, three different FIELD statements are used, depending on what action is to be taken with the file. This

James H. Garrett, Box 781, Holly Hill, FL 32017.

was not explained in any manual or article that I read, and is one of the main reasons for the difficulty I had in learning random access. Let's take a closer look at what the FIELD statement is and why three different ones were required.

Information is stored in chunks of 255 bytes, but forget the word byte and use the word character. Thus, information is stored in chunks of 255 characters. A character is a letter, a number, a space or a punctuation symbol. These 255 characters are called a *physical record*, and when we write to, or read from, the disk, this entire physical record is moved from the computer to the disk or from the disk to the computer. The space for 255 characters is taken up by a physical record regardless of how little information we put into it.

If the information that we wanted to store was exactly 255 characters long the FIELD statement would be simple. In most cases, however, the information is considerably shorter and contains information that has to be separated, such as names and addresses. As a result we have to "define the field."

Put another way, we have to tell the computer how many characters are in the name and how many are in the address. Easy enough. We can estimate that 99% of all names will not exceed 63 characters, and the same is true for most addresses. However, 63 + 63 = 126, so we would not be anywhere near the 255 characters that will be taken up by the physical record.

What we do then, is devise a method of storing two names and two addresses in each physical record. If we do this, 126 + 126 = 252, so we leave the space for only three characters unused. We then have two *subrecords* in each physical record.

Lines 300 to 340 are a FOR-NEXT loop used to input two names and two addresses into the subscripted variables N$(I) and A$(I). When this is done, the program moves to line 350, OPENS the file, and then goes into another FOR-NEXT loop at line 360.

Line 370 begins one of the three FIELD statements used in the program. The first time through the FOR-NEXT loop, I is equal to 0. Zero times anything is zero, so DU$ takes up zero space, N1$(I) is assigned 63 characters and A1$(I) is assigned 63 characters. DU$ is a "dummy" string and is not used for storing information. Its purpose is to format the 255 available characters and enable us to create subrecords. Many confusing explanations of how DU$ is used are already in print so I will not cloud the issue further by giving my own version. It is sufficient to say that "this is one way that it will work."

Now let's look at the FIELD statements

```
10 '            * - * - * - * - * - * - * - * - *
20 '            *                               *
30 '            *          NO FRILLS            *
40 '            *                               *
50 '            *        RANDOM ACCESS          *
60 '            *                               *
70 '            *         TECHNIQUES            *
80 '            *                                *
90 '            *             BY                *
100 '           *                               *
110 '           *      JAMES H. GARRETT         *
120 '           *          BOX 781              *
130 '           *    HOLLY HILL, FL. 32017      *
140 '           *                               *
150 '           * - * - * - * - * - * - * - * - *
160 CLEAR 6000
170 DIM N$(20)'                              *** FOR ALPHA SORT ***
180 CLS
190 PRINT "1. ADD TO OR CREATE FILE"
200 PRINT "2. SELECT ONE ITEM TO REVIEW"
210 PRINT "3. REVIEW ENTIRE FILE"
220 PRINT "4. CORRECT ONE ITEM"
230 PRINT "5. CHECK FILES FOR UNUSED RECORDS"
240 PRINT "6. INPUT FILES FOR ALPHA SORT"
250 PRINT "7. CHECK FOR END OF FILE"
260 PRINT "8. SEARCH FOR NAME"
270 INPUT "SELECTION" ; X
280 ON X GOTO 290 , 480 , 600 , 740 , 930 , 1110 , 1300 , 1390
290 CLS'                                  *** ADD TO OR CREATE FILE ***
300 FOR I = 0 TO 1
310 INPUT "NAME" ; N$(I)
320 IF N$(I) = "@" THEN 350
330 INPUT "ADDRESS" ; A$(I)
340 NEXT
350 OPEN "R" , 1 , "FILES"
360 FOR I = 0 TO 1
370 FIELD 1 , ( I * 126 ) AS DU$ , 63 AS N1$(I) ,63-AS A1$(I)
380 NEXT
390 FOR I = 0 TO 1
400 LSET N1$(I) = N$(I)
410 LSET A1$(I) = A$(I)
420 NEXT
430 I = LOF (1) + 1
440 PRINT "LOF = " I
450 PUT 1 , I
460 CLOSE
470 GOTO 180
480 CLS'                              *** SELECT ONE ITEM TO REVIEW ***
490 INPUT "WHICH RECORD DO YOU WISH TO SEE" ; X
500 PR = INT ( ( X - 1 ) / 2 ) + 1
510 SR = X - 2 * ( PR - 1 )
520 OPEN "R" , 1 , "FILES"
530 FIELD 1 , (( SR - 1 ) * 126 ) AS DU$ , 63 AS N1$ , 63 AS A1$
540 GET 1 , PR
550 N$ = N1$ : A$ = A1$
560 PRINT N$ , A$
570 CLOSE
580 INPUT "PRESS ENTER TO CONTINUE" ; X
590 GOTO 180
600 CLS'                                  *** REVIEW ENTIRE FILE ***
610 OPEN "R" , 1 , "FILES"
620 FIELD 1 , 63 AS N1$(0) , 63 AS A1$(0)
630 FIELD 1 , 126 AS DU$ , 63 AS N1$(1) , 63 AS A1F$(1)
640 FOR I = 1 TO LOF (1)
650 GET 1 , I
660 FOR Y = 0 TO 1 : N1$ = N1$(Y) : A1$ = A1$(Y)
670 N$(Y) = N1$ : A$(Y) = A1$(Y)
680 PRINT N$(Y) , A$(Y)
690 INPUT "PRESS ENTER TO CONT" ; Z
700 NEXT Y
710 NEXT I
720 CLOSE
730 GOTO 180
```

in lines 620 and 630. These are similar to the ones discussed above, but are not quite the same. A FOR-NEXT loop is not used; instead the computer is directed to use the specific subscripted variables of N1$(0) and A1$(0) in line 620 and N1$(1) and A1$(1) in line 630. It is possible to use the same method as above, but I wrote it this way intentionally to provide a different approach to the problem.

The third FIELD statement is used at line 530. The reason that it is used this way is because we want to look at a subrecord contained within a physical record. This part of the program begins on line 490 where we are asked the number of the item that we want to observe and to input it as the variable "X." Let us say that we want to look at our entry number 410. Thus X = 410. On line 500, PR (the physical record number) is set equal to the integer of 410 - 1 (409) divided by 2 (204) + 1. Thus, PR is equal to 205. The information we want to observe is located on the disk in physical record number 205, but we only want to observe the information contained in the subrecord.

Line 510 seems a bit confusing unless we remember that operations in parenthesis are done first, then multiplication and then subtraction. SR (the subrecord) is set equal to PR(205)-1 times two, or 408. This is then subtracted from "X" (410) to give us the number 2. We are now looking for subrecord 2 in physical record 205.

In line 530, we FIELD SR - 1 (or 1) times 126 as DU$ and get subrecord 2 into N1$ and A1$. Notice that these are not subscripted variables this time. Had we wanted the information that was stored in subrecord 1, SR would have been equal to 1. 1-1 = 0. 0 times 126 would equal 0 and DU$ would consume nothing. The first 63 characters would go into N1$ and the second 63 characters into A1$.

Now you know as much about FIELD statements as I do.

The GET Statement

Continuing at line 540 we use the GET statement to get the physical record from disk, and we tell the computer that we want number PR or 205. This it does and the information is stored in N1$ and A1$, which are at this point called "buffer field names" and are not, in the ordinary sense, string variables.

To stay out of trouble in this area, do not use buffer field names as variables elsewhere in your program. Line 550 accomplishes this by making N$ equal to N1$ and A$ equal to A1$. Line 560 prints N$ and A$. We could just as easily have printed N1$ and A1$, but then our buffer field name would have become a variable name and problems could creep in. Again,

```
740 CLS'                                    *** CORRECT ONE ITEM ***
750 INPUT "WHICH RECORD DO YOU WISH TO CORRECT" ; X
760 PR = INT ( ( X - 1 ) / 2 ) + 1
770 SR = X - 2 * ( PR - 1 )
780 OPEN "R" , 1 , "FILES"
790 FIELD 1 , ((SR - 1) * 126) AS DU$ , 63 AS N1$ , 63 AS A1$
800 GET 1 , PR
810 N$ = N1$ : A$ = A1$
820 PRINT N$ , A$
830 PRINT "ENTER @ TO DELETE FROM FILE"
840 INPUT "CHANGE NAME" ; N$
850 IF N$ = "@" THEN 870
860 INPUT "CHANGE ADDRESS" ; A$ : GOTO 880
870 A$ = ""
880 LSET N1$ = N$
890 LSET A1$ = A$
900 PUT 1 , PR
910 CLOSE
920 GOTO 180
930 CLS'                    *** CHECK FILES FOR UNUSED RECORDS ***
940 C = 0
950 PRINT "THE FOLLOWING RECORD NUMBERS ARE OPEN"
960 OPEN "R" , 1 , "FILES"
970 FIELD 1 , 63 AS N1$(0) , 63 AS A1$(0)
980 FIELD 1 , 126 AS DU$ , 63 AS N1$(1) , 63 AS A1$(1)
990 FOR I = 1 TO LOF (1)
1000 GET 1 , I
1010 FOR Y = 0 TO 1 : N1$ = N1$(Y) : A1$ = A1$(Y)
1020 C = C + 1
1030 N$(Y) = N1$ : A$(Y) = A1$(Y)
1040 IF LEFT$ ( N1$ , 1 ) = "@" THEN 1050 ELSE 1060
1050 PRINT C
1060 NEXT Y
1070 NEXT I
1080 CLOSE
1090 INPUT "PRESS ENTER TO CONTINUE" ; X
1100 GOTO 180
1110 CLS'                       *** INPUT FILES FOR ALPHA SORT ***
1120 C = 0
1130 OPEN "R" , 1 , "FILES"
1140 FIELD 1 , 63 AS N1$(0) , 63 AS A1$(0)
1150 FIELD 1 , 126 AS DU$ , 63 AS N1$(1) , 63 AS A1$(1)
1160 FOR I = 1 TO LOF (1)
1170 GET 1 , I
1180 FOR Y = 0 TO 1 : N1$ = N1$(Y)
1190 IF LEFT$ ( N1$ , 1 ) = "@" THEN 1220
1200 C = C + 1
1210 N$(C) = LEFT$ ( N1$ , 4 )
1220 NEXT Y
1230 NEXT I
1240 CLOSE
1250 FOR I = 1 TO C
1260 PRINT I , N$(I)
1270 NEXT
1280 INPUT "PRESS ENTER TO CONTINUE" ; X
1290 GOTO 180
1300 CLS'                              *** CHECK FOR END OF FILE ***
1310 OPEN "R" , 1 , "FILES"
1320 FIELD 1 , 63 AS N1$(0) , 63 AS A1$(0)
1330 FIELD 1 , 126 AS DU$ , 63 AS N1$(1) , 63 AS A1$(1)
1340 PRINT "END OF FILE IS "LOF (1) : X = LOF (1)
1350 PRINT : PRINT "THERE ARE " X * 2 " RECORDS IN THE FILE"
1360 CLOSE
1370 INPUT "PRESS ENTER TO CONTINUE" ; X
1380 GOTO 180
1390 CLS'                                    *** SEARCH FOR NAME ***
1400 C = 0
1410 INPUT "WHAT IS THE NAME THAT YOU ARE LOOKING FOR" ; N$
1420 OPEN "R" , 1 , "FILES"
1430 FIELD 1 , 63 AS N1$(0) , 63 AS A1$(0)
1440 FIELD 1 , 126 AS DU$ , 63 AS N1$(1) , 63 AS A1$(1)
1450 FOR I = 1 TO LOF (1)
1460 GET 1 , I
1470 FOR Y = 0 TO 1 : N1$ = N1$(Y)
1480 C = C + 1
1490 IF LEFT$ (N1$ , 3) = LEFT$ (N$ , 3) THEN 1500 ELSE 1510
1500 PRINT C , N1$
1510 NEXT Y
1520 NEXT I
1530 CLOSE
1540 INPUT "PRESS ENTER TO CONTINUE" ; X
1550 GOTO 180
```

suffice it to say, "this is one method that will work." If you want to know why, and you are a glutton for confusion, consult the manuals.

We just looked at subrecord number 410, but the file has not yet been created. Let's see how to create or add to a file.

In lines 300 to 340 we input information for storage. Lines 350-380 OPEN the file, name it and set up the field. Line 390 starts another FOR-NEXT loop and in line 400 we run into a new word, "LSET." LSET N1$(I)=N$(I) means to set or place N$(I) into N1$(I) on the left side. In line 370 we FIELDed 63 as N1$(I) so N1$(I) is 63 characters long. If the name that we entered as N$(I) in line 310 is "Smith" (5 characters), N1$(I) would start on the left as Smith and would be followed by (63-5) or 58 spaces.

In the same manner, if the name that we entered was 65 characters long, we would only have room for 63 of them, so the last two would be chopped off. The number of characters that we FIELD is what is used; no more and no less. And this brings up a problem for you to solve. Because all 63 characters are used, many blanks are on the right and should be removed when we bring the record back from the disk. What is a fast way to remove them?

On with the program. Line 430 establishes where the end of the file is so that our new information can be PUT at the end. Line 440 tells us where the end is, and line 450 PUTs it there. Line 460 CLOSEs the file, and we then return to the menu.

Correcting

Now we can add to the file and review one item in the file. How do we correct an item? Glad you asked. Correcting is the same as reviewing (lines 490 to 560), but at that point correcting takes a different direction. If you are with me up to here, I don't think that we need to tell each other that a subroutine could have been used.

Anyway, lines 750 to 820 are the same as 490 to 560 which were reviewed earlier. Line 830 prompts us to enter an "at" symbol if we want to delete the name from the file, or type in the name as it should be. If the name is deleted, line 870 deletes anything stored in A$ to eliminate possible future problems. The rest of this section is the same as adding to the file.

The other parts of the program are "goodies" that would be used as internal parts of other routines, but are shown as stand alone routines for clarity. For example, "Check Files for Unused Records" could be put into the "ADD TO FILES" part of the program to automatically fill in subrecords that had been deleted in the past.

The "Input for Alpha Sort" would be used so that names could be machine sorted instead of disk sorted, which will save disk wear. Using only the first four characters of the names to be sorted will speed up sorting and conserve memory.

The "Check for End of File" should be built into the program so that it is impossible to attempt to put more information on a disk than it will contain.

The "Search for Name" routine was thrown in because some manuals indicate that you have to have a written copy of your files showing the record number, or remember each record number, to find anything. This routine will give you all names that begin with the first three letters of the name that you are seeking. This could have been done more efficiently with a MID$ routine, but here again I wanted to keep the overall program simple.

Anyway, this should be enough information to get you started with random access. I hope it will save you much of the frustration that I encountered, and maybe will get you to the point that the manuals can be understood. □

Two Techniques for Creating One

Blinking Cursor for the TRS-80

Ray C. Horn, Jr.

Blinking Cursor! A really nice feature if you happen to own an Apple II. But what if you own a TRS-80? Are you doomed forever to have nothing but a do-nothing underline cursor? Never fear, an answer is close at hand.

In this article I present four Assembly listings outlining two techniques which almost any TRS-80 owner can use to get an action-packed blinking cursor.

Ray C. Horn, Jr., PSC Box 3303, Edwards AFB, CA 93523.

The first technique involves using the keyboard device control block driver address that Basic calls whenever keyboard input is needed, so the cursor only blinks when Basic is waiting for user input.

The second technique involves using the real time clock hardware interrupt generated from the expansion interface as a time base for a blinking cursor. This makes it possible to have a continuously blinking cursor as long as the user does not press the system reset.

All TRS-80 owners should be able to use the first technique since all TRS-80s use a certain amount of keyboard input through the running of normal Level II Basic, but there may be isolated cases where a TRS-80 owner may have the expansion box connected without disks and a software spooler running off the keyboard DCB. I decided to cover these few isolated cases because I own a 48K TRS-80 with an expansion box and no disks, so I can run a blinking cursor using either technique.

The relative strengths and weaknesses of the two methods depend upon whether or not you have an expansion interface

with the real time clock feature. Both techniques take an equal amount of memory.

There is one important consideration to bear in mind when deciding to use the real time clock interrupt driven blinking cursor. Cassette I/O will work reliably only with Model I Level III Basic. This is because without Level III Basic, cassette I/O will become garbled because the ISR (interrupt service routine) will be called continuously every 25ms during cassette operation.

With Level III Basic there are some new cassette I/O commands that were designed to be used with the real time clock. 'LOAD' disables all interrupts before cassette input then re-enables the interrupts before going back to Basic, and the 'SAVE' command works the same way. Since I did not write the blinking cursor routines with Level III in mind I'll leave it up to the user to determine the necessary changes that are needed in order to get BLINKs 3.0 and 3.21 (see Listing 3 and 4) to work properly.

BLINKS 3.0 and 3.21 were designed to be used in a clean Level II environment without any cassette I/O as demonstration routines to show that the basic principles do work. The same goes for BLINKs 2.0 and 4.0 as they, too, are only for demonstration purposes. It would take a bit more work to get the BLINKs to work properly outside of a clean Level II set up.

Assembly Listing Comments

There are four Assembly Language listings included with this article. Listing 1, BLINK 2.0, was originally intended to be used at the top of my 48K PMC-80 computer but may be easily relocated using an assembler by altering the 'ORG' on line 140.

Listing 2, BLINK 4.0, is an optimized BLINK 2.0 using only 51 bytes as compared to Listing 1, which needs a total of 71 bytes for a saving of 20 bytes.

Listing 3, BLINK 3.0, is actually BLINK 2.0 retrofitted for use with the real time clock hardware interrupt service routine.

Listing 4, BLINK 3.21, is an optimized BLINK 4.0 using only 86 bytes.

All four listings are well commented and all techniques are fairly straightforward with few, if any, programming tricks. Careful study of all four listings should prove enlightening for assembly hackers having difficulty implementing a blinking cursor on a TRS-80 or PMC-80. □

Listing 1.

```
                ;       Blinking Cursor Patch Version 2.0
                ;       Written by: Ray C. Horn Jr.
                ;       <c>.copyright 9 August 1980

FFB8                    ORG     0ffb8h  ;change as necessary

                ; Patch Link Initialization:

FFB8 2A1640     LinkB   Ld      Hl,(4016h) ; get old vector
FFBB 22FDFF             Ld      (Jump+1),Hl; old address
FFBE 21CDFF             Ld      Hl,Blink   ; get new vector
FFC1 221640             Ld      (4016h),Hl ; Link to Blink
FFC4 C37200             Jp      0072h      ; back to BASIC

                ; System Variables & etc.:

                ; Constants:
FFC7 8F         Cursor  Defb    8fh     ; allow user modification
FFC8 F401       Limit   Defw    500     ; user may modify Blink rate

                ; Variables:
FFCA 00         Cmode   Defb    0       ; 0=off, 1=on
FFCB 0000       Count   Defw    0       ; initially 0

                ; End System Variables...

                ;       Blink 2.0 Patch Code:

FFCD 2ACBFF     Blink   Ld      Hl,(Count)      ; get last 'Count'
FFD0 23                 Inc     Hl              ; Bump it
FFD1 EB                 Ex      De,Hl           ; De=Count, Hl=?
FFD2 2AC8FF             Ld      Hl,(Limit)      ; get upper limit
FFD5 AF                 Xor     A               ; zap carry flag
FFD6 ED52               Sbc     Hl,De           ; compute max limit?
FFD8 EB                 Ex      De,Hl           ;Hl=Count,De=Limit-Count
FFD9 22CBFF             Ld      (Count),Hl      ; updated Count
FFDC 201E               Jr      Nz,Jump         ; If not max limit!
FFDE ED53CBFF           Ld      (Count),De      ; else zero Counter
FFE2 3ACAFF             Ld      A,(Cmode)       ; get mode setting
FFE5 FE00               Cp      0               ; does mode=off?
FFE7 2808               Jr      Z,Turnon        ; if so, change mode
FFE9 AF                 Xor     A               ; else mode=on, turn off
FFEA 32CAFF             Ld      (Cmode),A       ; set mode=off
FFED 3E20               Ld      A,20h           ; get cursor 'off' char
FFEF 1807               Jr      SetIt           ; go and store char

FFF1 3C         Turnon  Inc     A               ; mode=off, turn on
FFF2 32CAFF             Ld      (Cmode),A       ; set mode = off+1 = on
FFF5 3AC7FF             Ld      A,(Cursor)      ; get cursor 'on' char
FFF8 2A0042     SetIt   Ld      Hl,(4200h)      ; get cursor address
FFFB 77                 Ld      (Hl),A          ; store new cursor
FFFC C3E303     Jump    Jp      03e3h           ; changed by LinkB


FFFF                    End
```

Listing 2.

```
                ;       Blinking Cursor Patch Version 4.0
                ;       Written by: Ray C. Horn Jr.
                ;       <c>.copyright 9 August 1980

FFCC                    ORG     0ffcch  ; change as necessary

                ; Patch Link Initialization:

FFCC 2A1640     LinkB   Ld      Hl,(4016h)      ; get old vector
FFCF 22FDFF             Ld      (Jump+1),Hl     ; complete circuit
FFD2 21E0FF             Ld      Hl,Blink        ; get new vector
FFD5 221640             Ld      (4016h),Hl      ; store it
FFD8 D9                 Exx                     ; switch CPU banks
FFD9 210000             Ld      Hl,0            ; zero counter
FFDC D9                 Exx                     ; switch CPU banks back
FFDD C37200             Jp      0072h           ; Back to BASIC
```

```
18:
19:                 ;       Blink 4.0 Patch Code:
20:
21:  FFE0 D9        Blink   Exx                     ; switch to Hl'
22:  FFE1 23                Inc     Hl              ; bump count
23:  FFE2 CB4C              Bit     1,H             ; Check Hl=512 from 0
24:  FFE4 2815              Jr      Z,Jump01        ; if count < limit
25:  FFE6 CB8C              Res     1,H             ; zero counter
26:  FFE8 CB7C              Bit     7,H             ; mode=off ? (0)
27:  FFEA 2806              Jr      Z,Turnon        ; if so, change mode
28:  FFEC CBBC              Res     7,H     ; else mode=on (1), set mode=off
29:  FFEE 3E20              Ld      A,20h           ; get cursor off char
30:  FFF0 1804              Jr      SetIt           ; go and store char
31:
32:  FFF2 CBFC      Turnon  Set     7,H             ; mode=off, so turn on
33:  FFF4 3E8F              Ld      A,8fh           ; get cursor on char
34:  FFF6 ED5B2040  SetIt   Ld      De,(4020h)      ; get cursor address
35:  FFFA 12                Ld      (De),A          ; store char
36:  FFFB D9        Jump01  Exx                     ; switch CPU banks back
37:  FFFC C3E303    Jump    Jp      03e3h           ; changed by LinkB
38:
39:  FFFF                   End
40:
```

Listing 3.

```
 1:                 ;       Blinking Cursor Patch Version 3.0
 2:                  ;      Written by: Ray C. Horn Jr.
 3:                  ;      <c>.copyright 9 August 1980
 4:
 5:  FFA1                    ORG     0ffa1h          ; change as necessary
 6:
 7:                  ; Patch Link Initialization:
 8:
 9:  FFA1 F3         LinkB   Di                      ; Disable interupts
10:  FFA2 ED56               Im      1              ; interupt mode 1
11:  FFA4 3EC3               Ld      A,0c3h         ; get jump code byte
12:  FFA6 321240             Ld      (4012h),A      ; store it at RST 38h
13:  FFA9 21B7FF             Ld      Hl,Isr  ;Interupt Service Routine address
14:  FFAC 221340             Ld      (4013h),Hl     ; store it at RST 38h+1
15:  FFAF FB                 Ei                     ; start blinking
16:  FFB0 C37200             Jp      0072h          ; Back to BASIC!
17:
18:                 ; System Variables & etc:
19:
20:                 ; Constants:
21:  FFB3 8F         Cursor  Defb    8fh             ; user selectable
22:  FFB4 0A         Limit   Defb    10              ; user selectable
23:                 ; End Constants...
24:
25:                 ; Variables:
26:  FFB5 00         Cmode   Defb    0
27:  FFB6 00         Count   Defb    0
28:                 ; End Variables...
29:
30:
31:                 ; Interupt Service Routine:
32:  FFB7 E5         Isr     Push    Hl              ; save active registers
33:  FFB8 C5                 Push    Bc              ; on stack.
34:  FFB9 F5                 Push    Af              ;
35:  FFBA 3AE037             Ld      A,(37e0h) ; get Interupt Latch status
36:  FFBD CB77               Bit     6,A     ; check floppy disk request
37:  FFBF 2033               Jr      Nz,Fdc          ; if bit 6=1
38:  FFC1 CB7F               Bit     7,A     ; check real time clock status
39:  FFC3 2832               Jr      Z,Xit           ; if bit 7 <> 1
40:  FFC5 3AB6FF     Blink   Ld      A,(Count)       ; get last count value
41:  FFC8 3C                 Inc     A               ; bump it
```

```
42:  FFC9 47              Ld      B,A             ; move into B register
43:  FFCA 3AB4FF          Ld      A,(Limit)       ; get upper limit in A
44:  FFCD 90              Sub     B               ; (A-B) Limit-Count
45:  FFCE 78              Ld      A,B             ; get count back in A
46:  FFCF 32B6FF          Ld      (Count),A       ; store it
47:  FFD2 201E            Jr      Nz,Jump ; if max Limit not reached
48:  FFD4 AF              Xor     A               ; zap Accumulator
49:  FFD5 32B6FF          Ld      (Count),A       ; zero counter
50:  FFD8 3AB5FF          Ld      A,(Cmode)       ; get cursor mode setting
51:  FFDB FE00            Cp      0               ; mode = off ?
52:  FFDD 2808            Jr      Z,Turnon        ; if so, change mode
53:  FFDF AF              Xor     A               ; zap Accumulator
54:  FFE0 32B5FF          Ld      (Cmode),A       ; set mode=off
55:  FFE3 3E20            Ld      A,20h           ; cursor off char
56:  FFE5 1807            Jr      SetIt           ; go and store char
57:
58:  FFE7 3C        Turnon Inc     A               ; mode=off, so turn on
59:  FFE8 32B5FF          Ld      (Cmode),A       ; set mode=off+1=on
60:  FFEB 3AB3FF          Ld      A,(Cursor)      ; get cursor on char
61:  FFEE 2A2040    SetIt  Ld      Hl,(4020h)      ; get cursor address
62:  FFF1 77               Ld      (Hl),A          ; store char
63:  FFF2 1803      Jump   Jr      Xit             ; return from interupt
64:  FFF4 3AEC37    Fdc    Ld      A,(37ech) ;reset floppy disk controller
65:  FFF7 3AE037    Xit    Ld      A,(37e0h) ;reset real time clock interupt
66:  FFFA F1               Pop     Af              ; restore active
67:  FFFB C1               Pop     Bc              ; working registers
68:  FFFC E1               Pop     Hl              ; from stack.
69:  FFFD FB               Ei                      ; enable interupts anyway
70:  FFFE C9               Ret                     ; Back to the farm
71:
72:  FFFF                  End
73:
```

Listing 4.

```
 1:                ;       Blinking Cursor Patch Version 3.21
 2:                ;       Written by: Ray C. Horn Jr.
 3:                ;       <c>.copyright 9 August 1980
 4:
 5:  FF90                  ORG     0ff90h          ; change as necessary
 6:
 7:                ; Patch Link Initialization:
 8:
 9:  FF90 F3       LinkB   Di                      ; disable interupts
10:  FF91 ED56            Im      1               ; interupt mode 1
11:  FF93 3EC3            Ld      A,0c3h          ; get jump code byte
12:  FF95 321240          Ld      (4012h),A       ; store it at RST 38h
13:  FF98 21A4FF          Ld      Hl,Isr  ;Interupt Service Routine addrs.
14:  FF9B 221340          Ld      (4013h),Hl      ; store it at RST 38h+1
15:  FF9E FB              Ei              ; start cursor blinking now.
16:  FF9F C37200          Jp      0072h           ; Back to BASIC!
17:
18:                ; System Variables and etc:
19:
20:                ; Constants:
21:  FFA2 8F      Cursor  Defb    8fh
22:                ; End Constants...
23:
24:                ; Variables:
25:  FFA3 00      Count   Defb    0
26:                ; End Variables...
27:
28:
29:                ; Interupt Service Routine:
30:  FFA4 E5      Isr     Push    Hl              ; save active registers
31:  FFA5 F5              Push    Af              ; on stack.
32:  FFA6 3AE037          Ld      A,(37e0h)
33:  FFA9 CB77            Bit     6,A
34:  FFAB 202F            Jr      Nz,Fdc
35:  FFAD CB7F            Bit     7,A
36:  FFAF 282E            Jr      Z,Xit
37:  FFB1 3AA3FF  Blink   Ld      A,(Count)       ; get last count value
38:  FFB4 3C              Inc     A               ; bump it
39:  FFB5 32A3FF          Ld      (Count),A       ; put it back
40:  FFB8 CB5F            Bit     3,A             ; test A=8 ?
41:  FFBA 281E            Jr      Z,Jump          ; If A <> 8, then...
42:  FFBC CB9F            Res     3,A             ; else reset counter
43:  FFBE 32A3FF          Ld      (Count),A       ; put counter back
```

```
44:  FFC1 CB7F            Bit     7,A             ; mode=off (0) ?
45:  FFC3 2809            Jr      Z,Turnon        ; if so, change mode
46:  FFC5 CBBF            Res     7,A             ; else mode=on (1)
47:  FFC7 32A3FF          Ld      (Count),A       ; set mode=off
48:  FFCA 3E20            Ld      A,20h           ; get cursor off char
49:  FFCC 1808            Jr      SetIt           ; go and store it
50:
51:  FFCE CBFF      Turnon Set    7,A             ; mode=off, so turn on
52:  FFD0 32A3FF          Ld      (Count),A       ; set mode
53:  FFD3 3AA2FF          Ld      A,(Cursor)      ; get cursor on char
54:  FFD6 2A2040    SetIt Ld      Hl,(4020h)      ; get cursor address
55:  FFD9 77              Ld      (Hl),A          ; store char
56:  FFDA 1803      Jump  Jr      Xit             ; return from interupt
57:  FFDC 3AEC37    Fdc   Ld      A,(37ech)       ; reset Fdc latch
58:  FFDF 3AE037    Xit   Ld      A,(37e0h)       ; reset latch
59:  FFE2 F1              Pop     Af              ; get working registers
60:  FFE3 E1              Pop     Hl              ; back from the stack.
61:  FFE4 FB              Ei                      ; make sure clock=on
62:  FFE5 C9              Ret                     ; return to system
63:
64:
65:  FFE6                 End
66:
```

Updating and Restoring Files

Automated File Handling Techniques

Irwin Doliner

You have just updated the PAYROLL file for your 100 employees when you realize the time cards you're using are the wrong ones. You also realize that you didn't make a copy of the PAYROLL file before updating it! The only thing to do is update again to delete the changes you just made — and hope that you don't make any mistakes. You make a note to remind yourself to copy the file before updating it — next time.

Maybe the update was made correctly and the program worked as it should, but what if there was a power failure in the middle of updating the file? Or there might have been a temporary malfunction during the update which made the file unreadable in the subsequent run. In either of these cases, if you did not make a back-up file, the only recourse you have is to try to reconstruct the file, and vow that you will never again forget to make a back-up.

Irwin Doliner, Interactive Data Systems, PO Box 290 Owings Mills, MD 21117.

Automatic Backups

Your program should prepare for these eventualities by retaining the current file as back-up and creating a new file containing the changes. The only requirement is that you tell the program the names of the current file and the one which will contain the updated data. Specifying the names correctly is a snap — if you can find the listing from the previous update run.

For example, you may have specified the input file as PAYROLL1 and the output file as PAYROLL2 in the previous run. Then, for this run, the input file is PAYROLL2 and the output file is PAYROLL1. You must keep accurate records of each run so that you specify the correct names for input and output. If you correctly specify PAYROLL2 and input and PAYROLL1 as output then everything is fine. But if you inadvertently reverse the names, you will have lost the update from the previous run.

This article discusses a file handling method which takes care of retaining the back-up file automatically. It also eliminates the need for the operator to know the names of the input and output files. This method will "tell" the program the correct file names without any operator intervention. The operator will know the file simply by a generic name (e.g., "PAYROLL").

Back-up files help to insulate you from system problems and help you to recover from such problems as

FILE LINK errors, DISK I/O errors and problems caused by power failures. The following technique will provide this necessary protection automatically without operator intervention.

How Is It Done?

It will be necessary to create three files to implement this method. However, the program will "know" them by a single (generic) name. To illustrate, suppose that you want to create a file called "PAYROLL." For this method you must create files called "PAYROLL," "PAYROLL1" and "PAYROLL2." Your program will only "know" the name "PAYROLL" but your data will actually be stored in the other two files.

The file with the generic name is the master file, and it contains only the names of the two data files. The order of occurrence of the names in the master file determines which data file is the most current and which is the back-up. In the above example, "PAYROLL" is the master file and "PAYROLL1" and "PAYROLL2" are the data files. Program A demonstrates how this system of files would be initialized (also see Figure 1).

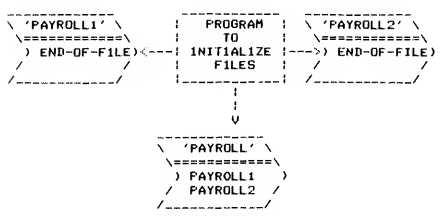


Figure 1. Initializing Files.

Program A

```
10 OPEN "O",1,"PAYROLL"     <-create master file.
20 PRINT #1,"PAYROLL1"      <-add names of
30 PRINT #1,"PAYROLL2"      <-two data files.
40 CLOSE 1                  <-close master file.
50 OPEN "O",1,"PAYROLL1"    <-create data file #1.
60 CLOSE 1                  <-close data file #1.
70 OPEN "O",1,"PAYROLL2"    <-create data file #2.
80 CLOSE 1                  <-close data file #2.
```

Thus, "PAYROLL1" is established initially as the current data file, and "PAYROLL2" as the back-up. When you run your program you will read data from the current file and write the updated data into the back-up file and then reverse the order of the names. Program B is a partial program which will demonstrate this technique (see Figure 2).

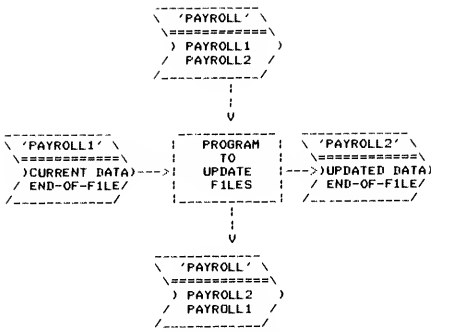


Figure 2. An Update Run.

On the next run the program will automatically be given "PAYROLL2" as the input file and "PAYROLL1" as the output file, since PAYROLL2 is first in the master file and PAYROLL1 second. The operator need never be concerned with the actual data file

Program B

```
10 OPEN "I",1,"PAYROLL"     <-open the master file.
20 INPUT #1,F1$             <-name of current file.
30 INPUT #1,F2$             <-name of backup file.
40 CLOSE 1                  <-close master file.
50 OPEN "I",1,F1$           <-open the input file.
60 OPEN "O",2,F2$           <-open the output file.
       .
       .
       .
 ------------
|            |
|   coding   |
|     to     |
|     do     |
|    the     |
|  required  |
|   updates  |
|------------|
       .
       .
       .
1000 CLOSE 1                <-close input file.
1010 CLOSE 2                <-close output file.
1020 OPEN "O",1,"PAYROLL"   <-open master for output.
1030 PRINT #1,F2$           <-reverse the order of
1040 PRINT #1,F1$           <-the data file names.
1050 CLOSE 1                <-close master file.
```

names and is, thus, less concerned with the workings of the computer.

The hypothetical problems posed at the beginning of this article could have been solved very simply if this method had been employed. The incorrect update would have been "undone" effectively by merely reversing the order of the file names in the master file. This restore function could have been accomplished with coding similar to that in Program C. (Figure 3 is a flowchart of the program.)

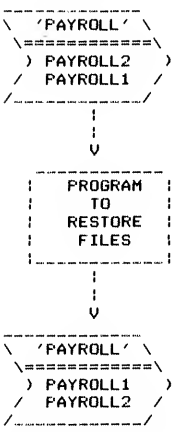


Figure 3. File Restoral (Deleting The Last Update).

Program C

```
1000 OPEN "I",1,"PAYROLL"
1010 INPUT #1,F1$
1020 INPUT #1,F2$
1030 CLOSE 1
1040 OPEN "O",1,"PAYROLL"
1050 PRINT #1,F2$
1060 PRINT #1,F1$
1070 CLOSE 1
```

One way to improve the usefulness of this method is to make the file handling method a subroutine and use a variable for the name of the master file. For example, your program would pass the name of the master file to the subroutine as follows:

```
100 M$="PAYROLL"
110 GOSUB 32000
       .
       .
       .
32000 OPEN "I",1,M$
32010 INPUT #1,F1$
32020 INPUT #2,F2$
32030 CLOSE 1
32040 OPEN "I",1,F1$
32050 OPEN "O",2,F2$
32060 RETURN
```

Summary

You may become very inventive with this method. There is no reason why the master file may only contain the names of the data files. You might wish to include the date that each update was made, the initials of the operator who made it, or the total number of updates made so far.

What is most important about this method is the human engineering aspect it will lend to your programs. It allows the operator to concentrate on the function he is trying to perform and not on the way that the computer operates to help him perform it. □

Using Field Specifiers on the TRS-80

John Adams

The first "big" program that I wrote for my TRS-80 had to do with household budget and expenditures, and it came close to being my last. I had not done much programming and I was still working in Level I. The program slowly and painfully worked out to my satisfaction, except for the printouts. To give you an example of what I was getting RUN the following lines.

Listing 1.

```
1   REM * MONEY PRINTING
    ROUTINE
10  FOR X=1 TO 5:READ P(X):NEXT
20  DATA 254.13,1041.19,42.90,6.00,.79
30  CLS:PRINT"SALES SLIP #",
    "AMOUNT"
40  FOR X=18255 TO 18259:PRINT X,
    "$";P(X-18254):NEXT
```

All of the information is there, but that number column looks downright seedy. I spent a couple of days and almost went berserk trying to line up those miserable figures. Level II, however, offers us some solutions to this problem. It includes a group of "field specifiers" which take a lot of the misery out of generating reports.

As applied to computers, the word "field" has several meanings. If we are creating reports, "field" means an area into which a specific category of data is to be printed. Look at Table 1.

This report has five fields; one for the month, one for food, one for rent, one for utilities and one for the total. The specifiers under discussion allow us to design a pattern or "format" for each of these fields. These formats are stored in memory locations and are called into use by the PRINT USING instruction. An example follows:

```
SALES SLIP #     AMOUNT

 18255           $ 254.13
 18256           $ 1041.19
 18257           $ 42.9
 18258           $ 6
 18259           $ .79
```

Printout of Listing 1.

```
10  A$="###":B=456
20  PRINT USING A$;B
```

Line 10 sets up the format in A$ and loads a value into B. All formats are loaded into string locations. Line 20 is actually a routine which takes the value stored in B and prints it out in accordance with the specified pattern. A semicolon is used to separate the format string name from the data which follows. Notice that the format loaded into A$ is enclosed in quotes. Such patterns may also be loaded by the READ-DATA method (10 READ A$:DATA ###) in which case the quotes are not needed. Formats are composed of various symbols found on the keyboard.

Table 1. Breakdown of Expenses.

MONTH	FOOD	RENT	UTILITIES	TOTAL
JANUARY	$221.50	$425.00	$145.89	$792.39
FEBRUARY	$199.58	$425.00	$155.35	$779.93
MARCH	$240.04	$425.00	$139.20	$804.24

John Adams 13126 Tripoli Ave., Sylmar, CA 91342.

Since much formatting is done with numbers we should perhaps start there.

The number or pound sign (#) is used in the format to specify the number of digit spaces needed in the field. Look back at line 10 and you will see that "###" was put into A$. This sets the field length at three. To get a look at what this does, enter and RUN the following lines.

```
10  A$="###"
20  FOR X=1 TO 5:READ
    P(X):NEXT
30  DATA 123,4567,8,-90,7.5
40  FOR X=1 TO 5:PRINT-
    TAB(20)P(X);TAB(40)USING
    A$;P(X):NEXT
```

The left-hand column is printed without the format, the right with the format. Several differences are immediately apparent. On the left all numbers are lined up from the leftmost digit. Actually the positive numbers all begin at tab 21 as a space is left for the "+" sign which is not needed. On the right the numbers (with the exception of the second) are lined up from the rightmost digit. Unless there is some reason not to, this is the way tables of numbers are usually printed.

So why is the second number not lined up properly? And where did the percent sign come from? It is an indication that the field size is too small. Line 10 specified three spaces but the number has four digits. The percent sign tells you so, then goes ahead and prints the entire number anyway. It is out of line because the field cannot contain it. It is therefore prudent to include enough number signs in your format to hold the largest number that you will be using. As we go along you will see that adding other symbols to the format increases the field size. Change line 10 to read A$="####". The percent sign, and the lineup problem, disappear.

```
1   REM * REVISED MONEY PRINTING ROUTINE
5   A$="$$#,###.##"
10  FOR X=1 TO 5:READ P(X):NEXT
20  DATA 254.13,1041.19,42.90,6.00,.79
30  CLS:PRINT"SALES SLIP #", "      AMOUNT"
40  FOR X=18255 TO 18259:PRINT X,USING A$;P(X-18254):NEXT
```

Listing 2.

There is also a difference in the last item, which was changed from 7.5 to 8. The format used did not specify that decimals were wanted, so none were furnished, and the 7.5 was rounded up to 8. Regular 5/4 rounding was used here; 7.4 would round down to 7 and 7.5 up to 8. This feature can be used to advantage. When working with decimal numbers we can use a format to round to whole numbers (or any number of decimal places) and this can eliminate one program line.

Decimals may be requested by inserting a period in the format, and its position will indicate the number of places to be returned. Change line 10 to read A$="####.#". The 7.5 is now printed properly, and a bonus is picked up in the "trailing" zeroes following the numbers. More about that in a minute. Changing line 10 to read A$="####.##" will cause the numbers to be printed with two decimal places. You can, by adding number signs to the right of the decimal, get up to 16 decimal places. Be careful with precision, however, as decimal digits may be incorrect after the sixth place unless you have declared double precision. If you are unsure about precision see "Variable Types" on page 1/3 and DEFINT, DEFSNG and DEFDBL in Chapter 4 of your manual.

The bonus that was mentioned has to do with printing money figures. The TRS-80 has a built-in function to suppress trailing zeroes. Enter PRINT 5.0000 and the monitor will return 5. Since the two numbers represent equal amounts the zeroes are not needed. Entering PRINT 5.0001, on the other hand, will return 5.0001. The zeroes are now "significant" as the two numbers are different amounts. In the original money-printing routine, the third and fourth items of data were 42.90 and 6.00 and if you recall these were printed as $ 42.9 and $6. Using a format will guarantee that money columns will have two, and only two, decimal places.

We might also want dollar signs in our money columns. Adding two dollar signs ($$) to the beginning of the format will act as a floating dollar sign which will print immediately to the left of the first digit.

The comma is useful in printing large numbers. It is easier to read 12,345,678.90 than 12345678.90, as we are accustomed to seeing large numbers divided into periods of three. Inserting a comma any place *after* the first number sign and before the decimal will do this for us. Only one comma is needed; the computer will then put commas in front of each group of three digits.

Using these four symbols (number sign, period, dollar sign and comma) we can polish up that first routine. The revised version is shown in Listing 2.

Note the differences from the original version in lines 5, 30 and 40. The addition of the four spaces before the word AMOUNT in line 30 is to adjust the column heading. There is a better way to do this which will be discussed later.

If the computer is used to print checks or other numbers that could be altered, using a pair of asterisks in the format will fill in any unused spaces in the field with stars. The following lines illustrate:

```
10  A$="######,####":B$=
    "**####,####":A=1234
20  PRINT USING A$;A
30  PRINT USING B$;A
```

The asterisks and the dollar sign can be used in conjunction. Change line 10 so that B$ reads "**$###,###". For this format only one dollar sign is used (**$). You should be aware that adding asterisks, dollar signs, periods, etc., increase the field size. There are now eleven characters in the B$ format, so that field could contain a dollar sign, eight digits and two commas. Again, be careful with precision.

Just as the TRS-80 suppresses trailing zeroes, it also omits the "+" sign in front of positive numbers. Should you need such signs they can be incorporated into the format. An example is shown in Listing 3.

RUN this and notice that the leading signs are positive and negative as the occasion demands. Putting the plus or minus signs at the end of the format will cause them to be printed as trailing signs. See page A/7 of the manual.

The last of the number formatting signs is the "up" arrow, normally used for indicating exponentiation. Including four such arrows at the end of the format will cause printout in exponential or "scientific" notation. Such notation is used with very large or very small numbers and expresses them more compactly as small groups of significant numbers times powers of ten. If you have need for such numbers you probably know what they are and how they work. For more information consult any math reference book.

Up to this point we have been dealing with formatting numbers, but there are two symbols that can assist in formatting strings. Remember that strings can contain any mixture of letters, numbers and symbols. The first of these signs is the exclamation point and it is used to excerpt the first character of any given string. RUNning the following will return the letter A.

```
10  A$="ABCDE"
20  PRINT USING "!";A$
```

If such a format will be used frequently it

Listing 3.

```
10  A$="+##.#"
20  CLS:PRINT"TEMPERATURE DEVIATION FOR MAY 1 TO MAY
    10":PRINT
30  PRINT"DATE", "AVERAGE","TEMPERATURE","DEVIATION"
40  FOR X=1 TO 10:B(X)=RND(10)+80:C=C+B(X):NEXT:D=C/10
50  FOR X=1 TO 10:PRINT"MAY";X,D,B(X),
60  PRINT USING A$;B(X)-D:NEXT
```

can be stored in a variable location and called when needed as in the following.

```
1Ø  A$="ABCDE":Z$="!"
2Ø  PRINT USING Z$;A$
```

Two or more exclamation points may be used in combination and spaces can be included before, in between and following the points to establish the string field. RUN the following example.

```
1Ø  A$="ABCDE":B$=
    "12345":Z$="  !  !  "
2Ø  PRINT USING Z$;A$,B$
```

This creates a string field in Z$ which contains two spaces, the first letter of the first listed string, two more spaces, the first letter of the second listed string, and then two more spaces. The printout therefore reads space-space-A-space-space-1-space-space. This is handy for creating acronyms from titles, such as CBS from Columbia Broadcasting System.

The second of the symbols is the percent sign (%). Using a pair of these will establish a string field size of 2 *plus* the number of spaces included in between the signs. This allows us to lift any number of characters from a given string, starting from the first character. Try the lines in Listing 4.

```
1Ø  READ A$,B$,C$:DATA JANUARY, FEBRUARY, MARCH
2Ø  Z$="% %":CLS:PRINT "TOTALS";
3Ø  PRINTTAB(20)USING Z$;A$;
4Ø  PRINTTAB(30)USING Z$;B$;
5Ø  PRINTTAB(40)USING Z$;C$
```

Listing 4.

The field in Z$ is set to three, so the first three characters of each listed string are returned. Note how PRINT USING is used with PRINTTAB. It works equally well with PRINT@, but in this case a comma is inserted after the location number as in PRINT@50,USING Z$;A$. A short routine which involves both of these symbols is shown in Listing 5.

Line 1Ø sets up two string fields. In Y$ the percent signs define a four space field which will pick up the first four characters of the indicated string. In line 5Ø therefore, the word THE with its trailing space will be printed. In Z$ the second field contains three characters consisting of the first letters of the three strings indicated.

As a final exercise let us go back to the revised version of the money printing routine. Remember that we added four spaces before the word AMOUNT in line 3Ø. We did this to move our column heading over to correspond with the money column. Since a comma was used as a position indicator before the word AMOUNT, it printed out starting in the first space of the second screen zone, or in space number 17. Also recall that when we define a field, the item in that field will be printed out with the *last* character in the rightmost position of the field. Look back at line 1Ø and count the characters in the field format stored in A$; 2 dollar signs, 6 number signs, one period and one comma for a total of ten. There are only 6 number signs, but a space in the field has been added for each additional symbol. So any data in that field will be printed with the last character in position 27 (17 plus the 10 in the field). To line up the column heading we took the number of letters in AMOUNT (6) and added the four leading spaces so the string field size was also 10.

This string field can be defined at the beginning of the program along with the number field. Change line 5 to read: 5 A$="$$#,###.##":B$=" % %". The field in B$ now has 10 spaces. In printout the first four will show as spaces,

Listing 5.

```
1Ø  Y$="%  %":Z$="!!!"
2Ø  READ T$,F$,B$,0$,I$
3Ø  DATA THE ,FEDERAL ,BUREAU ,OF ,INVESTIGATION
4Ø  PRINT T$;F$;B$;0$;I$;"IS SOMETIMES CALLED"
5Ø  PRINT USING Y$;T$;:PRINT USING Z$;F$,B$,I$
```

and the percent signs will then put the first six characters in the word AMOUNT (all of them) into the end of the field. This gets our heading lined up. The printout line must also be changed and should look like this.

```
3Ø  CLS:PRINT"SALES SLIP #",
    USING B$;"AMOUNT"
```

As we can see the field specifiers are a versatile lot. Experiment with them. Once you have the hang of them, your reports will really get that professional look. □

Programming Techniques for TRS-80 Model I

Re-Discovering Level II

Robert Spahitz

Although Radio Shack has replaced its Model I computer with the Model III and Color Computer, I continue to use my Model I, and I suspect that many other people do, too.

This article is written specifically for those who still program on a Model I, Level II, but it may also be of use to Model III users and other programmers as well. It includes some simple programming techniques, a description of how to create an error which should not be (a bug in Radio Shack Basic...still extant in the Model III computer), a mini-review of Radio Shack's *T-Bug* and machine language, a mention of how to put control characters and arrows directly into text (inside PRINT statements for example), and an explanation of how to create sound without any extra peripherals.

Radio Shack Basic includes many unusual idiosyncracies that are rarely mentioned. For example, the Level II manual tells you that you should use THEN with all IF statements, when in reality, THEN is not needed in most cases, and in one case it is always needed. (THEN uses up one extra byte every time it is used, and it rarely makes the program any more understandable.)

If you are comparing only numeric data, THEN is usually needed only when the true part of the IF statement is an assignment with LET omitted. If you are comparing string data, THEN is always required for an ELSE clause to work; otherwise THEN is needed only if the dollar sign is omitted, because the variable being used is predefined in a DEFSTR clause (see Chart for examples).

Also, beware of hidden keywords when you program (Chart examples 12-16), especially when you are doing logical comparisons.

According to the Level II manual, INKEY$ "returns a one character string... the last key pressed." However, some keys do not work the way you might hope, specifically SHIFT, BREAK, and @.

Although there are two shift keys, they have no effect on INKEY$ unless pressed with another key. If that other key is @, program execution freezes. To have the program accept SHIFT @ as data, both keys must be pressed twice.

The other troublemaker is the BREAK key which almost always interrupts the program. To overcome this, POKE 16396,7 will disable the BREAK function and allow that key to act as any other key (CHR$(1)) until execution of POKE 16396,0. By placing these around input statements, you can prevent accidental BREAKing until after data input. Be sure to POKE into 16396 and not elsewhere, or you may not be able to recover your program.

Errors Or Not?

The INPUT statement, although useful, has several problems. If the variable is omitted after INPUT, no error will be picked up if the length of the input entered is zero before ENTER is hit. For example,

510 INPUT or 510 INPUT "HIT ENTER";

work fine if ENTER is hit immediately or if, as an example, HELLO and then five backspaces are typed before ENTER. Obviously this method of input is bad unless you know that the only input will be ENTER; otherwise the computer will try to store the information in the nonexistent variable, causing an error. Note: CLEAR can be used to replace backspaces.

Another INPUT problem has to do with the way data from the INPUT are interpreted. It affects both Model I and Model III, presumably because nobody at Radio Shack ever noticed this built-in error. When I first encountered this error, I thought there was a spelling mistake in my program line, because I was put into edit mode after a ?SN ERROR IN LINE xxx. When I listed the line, I saw nothing wrong, so I searched through the rest of my program. I still found nothing, so I ran the program again. After several tries, I found out that my input had caused the error.

Try typing the following program and RUNning it:

```
10 INPUT A
20 PRINT A
30 GO TO 10
```

By typing any number, you will get either the number printed (or rounded off first) or an overflow if the number is too large. Typing non-numbers or combining numbers and nonnumbers usually gives a ?REDO or ?EXTRA IGNORED message. However, a certain combination gives ?SN ERROR IN LINE 10, and puts you in EDIT mode for line 10.

In example 1 of Figure 1, the computer interpreted the period as meaning floating-point. In example 2, % means integer. In example 3, $ is in no way numeric. In example 4, the computer got confused. If asked what interest rate you will be using in a program, example 4 is a likely response, but the computer picks it up as an error and makes you lose your last 30 minutes of typing.

This error is obviously in ROM because input to a program is not supposed to cause the program to bomb with a syntax error. What causes it?

As far as I know, this error occurs on all Level II and Model III computers whenever a period is followed by a percent symbol but before either a comma or a colon, and if the data is being put into any numeric variable (either directly through INPUT or LET, or indirectly by converting a bad string to numerics). That is, this error will occur if you try to put 5.%, -3.14159%, .1%234, .% or any similar combination into an integer, single or double precision variable, or a string

Robert Spahitz, 143 Dewey Ave., Staten Island, NY 10308.

Figure 1.

	Entering this input	Should give this response
example 1	5.0	5
example 2	5%	5
example 3	$5	?REDO
example 4	5.0%	?SN ERROR IN LINE 10

variable of which the program will take the VALue.

Why does this error occur? Apparently, the computer interprets your input character by character after you hit ENTER. When it finds a period, it sets up storage for a floating point number. When it finds % it attempts to store the number as an integer. Since floating point numbers cannot be integers, the computer gets confused and claims that there is something wrong with your INPUT statement, LET statement, or VAL command.

How do I cure this problem? The only way I know to avoid it is to always use string inputs, either with INPUT or INKEY$, then edit out any % and other stray characters using LEFT$, MID$, and RIGHT$ (or do not allow stray-character input when using INKEY$), and finally convert the input to numeric data using the VAL command.

If you are really daring, you can create your own input routine using machine language. Two very useful tools for this are *How To Program The Z80* by Rodnay Zaks, Radio Shack catalog number 62-2066, $10.95 and *T-Bug*, Radio Shack catalog number 26-2001, $14.95.

How To Program The Z80 is a book that tells you exactly that. It has a listing of all Z80 instructions, tells how they work, and describes how to put them together with other instructions to create useful programs. Note: A working knowledge of Basic will help you better understand machine language instructions.

T-Bug is a Radio Shack creation which allows you to write, edit, and execute machine language programs. It is loaded from cassette and accepts eight different commands.

Along with the *T-Bug* cassette comes a very handy reference manual that carefully explains each command, includes a Level I to Level II conversion program, lists all of the Z80 opcodes and their symbolic names (unfortunately, no description of the commands is included), pictures a memory map of both Level I and Level II memory, lists some useful ROM subroutines (such as getting a character from the keyboard, and using the cassette and printer), contains a conversion table for binary-hexadecimal-decimal numbers, and finally lists some books to read for more information about machine language.

All in all, *T-Bug* is very useful for the machine language novice, but a book on the Z80 is a must if you want to understand the Z80 commands. If you are serious about speeding up your programs and using less memory, $15 for *T-Bug* and $11 for a good book is very reasonable.

Control Characters in Text

Moving along, let us examine a useful aspect of Level II. Through use of the EDIT command, any control character from 1 to 27 can be sneakily stored into one byte instead of the usual four or five bytes needed to store CHR$(NN) where NN would be a number from 1 to 27. This is tricky but worth the effort if you enjoy conserving memory and confusing anyone who reads your listings. The procedure follows:

1. Type the line in which you want the code, leaving one space where it belongs.
2. Enter EDIT mode by typing EDIT line number ENTER.
3. Space up to and over (not past) the space to be changed (usually right after the first quotation mark).
4. While holding down the down arrow, type C and release only the C key.
5. Hold the shift key down.
6. Type the letter of the alphabet corresponding to the code you want printed (A=1 B=2 C=3...Z=26 ↑=27)
7. Release all keys and hit ENTER or E to exit EDIT mode and store the code.

This may sound complex, but it is rather easy once you do it a few times. Try the following example:

```
10 PRINT CHR$(23) "HELLO"
```

To convert the above, first type

```
10 PRINT "HELLO"
EDIT 10 (ENTER)
```

and space past the first quote. Now hold the down arrow key, depress and release the C for change, hold the SHIFT key and type W, the twenty-third letter of the alphabet. If you do not hold the down arrow or type it after the C, your results will not be the same, so do it in the proper order. The down arrow somehow affects the TRS-80 character generation section, so by holding this down with other keys you will produce other characters.

This leads me to my next trick: producing arrows directly in text. This is much simpler than inserting control characters into text and is probably more useful. There are several ways to do this. I find this way easiest:

First depress and hold the Z and 2 keys, then backspace twice to erase the Z and the 2.

Then Press	To Get
3	↑3
4	↓4
5	←5
6	→6
7	_7

Finally, backspace once to erase the number, and you end up with the arrow. No EDIT mode was needed. Note: Sometimes the number comes before the arrow. In that case, just erase the arrow and the number, and try again.

Listing 1.

```
10 FOR Z = 1 TO 25
20 A = RND(300) - RND(600)
30 B = RND(300) + 300
40 C = RND(25)
50 D = RND(RND(9))
60 E = RND(RND(9))
70 GOSUB 100
80 NEXT Z : REM MAKE SOUND AGAIN
90 END
100 FOR Y = A TO B STEP C
110 OUT 255,0
120 FOR Z = 0 TO D
130 NEXT Z : REM DELAY
140 OUT 255,4
150 FOR Z = 0 TO E
160 NEXT Z : REM DELAY AGAIN
170 NEXT Y : REM REPEAT CLICK
180 RETURN
```

Listing 2.

```
10 OUT 255,8 : REM CHANGE SCREEN TO
   32-CHARACTER MODE
20 PRINT "T Y P E   A ,   B   O R
   C"
30 A$ = INKEY$
40 IF A$ < "A" OR A$ > "C" THEN 30
50 IF A$ = "A" PRINT "STOP" : STOP
60 IF A$ = "B" PRINT "END"  : END
70 PRINT "END OF PROGRAM"
```

When To Use 'THEN' In 'IF' Statements

Program Line	Works as Expected	Solution
1. IF A=B PRINT C	Yes.	
2. IF A=B C=5	No. Computer interprets B and C as one variable.	Use THEN after B.
3. IF A=B LET C=5	Yes.	
4. IF A=B*5 C=5	Yes!	
5. IF A=5*B1 C=5	No. Similar to number 2.	Use THEN after 1.
6. IF A$=B$ PRINT C$ ELSE PRINT	Yes, if A$=B$; No, if A$≠B$... ELSE clause disregarded.	Use THEN after B$.
7. IF A$=B$ C$="HI"	Yes.	
8. DEFSTR A,B:IF A=B PRINT C	Yes.	
9. DEFSTR A,B:IF A=B C=5	No. Same as number 2.	Use THEN after B.
10. DEFSTR A,B:IF A=B$ C=5	Yes. (A=B$ is correct)	
11. IF E=2 DISC=SQR(B ↑ 2-4*A*C)	No. New variable ISC is created and E is compared to 2D (double precision).	Use THEN after first 2.
12. IF A=T AND F=3 THEN STOP	Yes, *if* there is a space between the first T and the A in AND, otherwise interpreted as IF A=TAN...	Put space after T.
13. IF PM=SE THEN STOP	Similar to above: if space after first E is omitted, interpreted as IF PM= SET...	Put space after E.
14. FOR A=GO TO FINISH	NO! Does not work, even with spaces.	GO must be POKEd into memory; easier to change variable.
15. FOR A=AU TO LE	Only if space after U.	Put space after U.
16. PRINT TAUT OR LOOSE	Yes, with spaces, otherwise keyword AUTO gets in the way of this logical OR.	Put space before O.

Many are very similar to #16, so beware when logical ORing or ANDing.

Notes:

10. IF A=B$ C=5 is not an error.
11. This gave me different results, but the square root is not stored in DISC.

12-16. By leaving spaces after variables that effect the beginning of a keyword, most problems like this are resolved. For example, use =AU TO instead of =AUTO. This will not work when GO is next to TO because the computer disregards spaces, linefeeds and certain other CHR$ codes when trying to form the GOTO keyword.

Sound Too?

Finally, I have discovered that the TRS-80 can produce sound without any new software, hardware, or even a speaker. Although you do not get a symphony, you do get hums, buzzes, and clicks. There seems to be a flip-flopping device inside the keyboard that controls cassette input/output. By making it flip-flop at different speeds, you can produce several sounds directly from the keyboard. Since this device directly affects the recorder, it is wise either to turn off the cassette or to unplug the leads before attempting to make sounds.

The command to make sound is the infrequently used OUT statement. See Listing 1 for a sample program. By changing the delays in lines 120 and 150, or by changing the values in line 100, different sounds are produced.

The OUT statement can also be used to change the screen from 64-character mode to 32-character mode or vice versa, but it does this only during program execution. When the program stops, the screen reverts to its original mode. As a programmer, I find this useful because I can run my program in 32-character mode, and when I want to stop and list my program, I do not have to clear the screen, thereby losing any output I may have wanted to examine.

Try the sample program in Listing 2 to see how this works. Some problems do arise with this method, but they are minor. First, if CLS is encountered, OUT 255,8 must be re-executed. Second, all PRINTs that contain literals (anything inside quotes) must be spaced properly to appear on the screen correctly (see line 20 in Listing 2).

Summary

- THEN is not usually needed, but should be included when the true part of IF statement is an assignment, or if strings are being compared.
- INKEY$ does not accept SHIFT as a single character; you must type SHIFT-@ twice to store it; you must disable the BREAK key to be able to store it (POKE 16396,7).
- INPUT does not need a variable, but if you omit it, the only acceptable input is ENTER; if INPUT is used with numeric variables, or the program takes the VAL of a string, the program will bomb if the input contains a period followed by %.
- The EDIT subcommand C can be used to get control characters to be stored directly in text.
- By holding down the 2 and the Z keys, you can get arrows or an underscore directly on the screen by typing 3, 4, 5, 6 or 7.
- Sound can be produced by switching from OUT 255,0 to OUT 255,4 and back, and including delays between them.
- OUT 255,8 will convert to 32-character mode during program execution as long as CLS is not encountered, and stay in that mode until the program is stopped.

I hope you have found these items useful and will include them to make your programs more efficient and, ultimately, more exciting.

If you have questions or comments, I would be pleased to read and respond to any that contain a self-addressed envelope. □

Model II and III ROM Look Alikes

Joseph Cesaitis

Now that the Model III TRS-80 computer has made its way into the marketplace, it appears to have successfully replaced the Model I and left over 200,000 owners out in the cold with a software compatibility problem. Or has it?

Many Basic programs will run on both machines with few or no changes. Machine language programs which do not use any ROM code should function properly assuming no critical timing loops are used.

The Model III Z-80A CPU runs slightly (14%) faster than its predecessor. Machine language programs which utilize ROM routines may or may not have problems depending on the routine used.

Table 1 lists the addresses (from-to) which are identical in both machines. As you can see much of the code is identical (thank you, Tandy). In those cases where the from and to addresses are the same only a single byte matched. This table should aid those who wish to write machine language software for both machines, but only have access to one machine. □

Joseph Cesaitis, 10380 Launcelot Lane, Columbia, MD 21044.

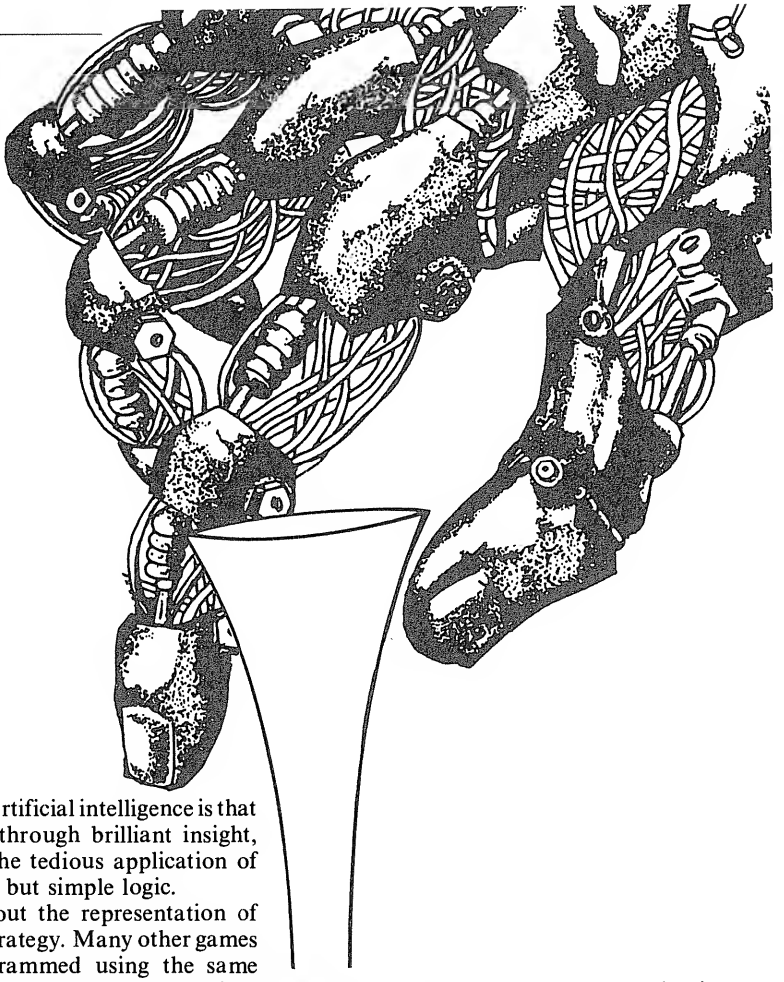
0000-0002	0005-000D	0010-0046	0049-004F	0063-0065
0071-0081	0083-00A9	00AB-00AD	00AF-00B1	00B5-00C5
00C7-00E9	00EB-00FE	0102-0105	010B-010C	0110-0111
0116-0117	011C-011C	0122-0124	012D-01D9	01F0-01F0
01F2-01F2	01F5-01F7	0210-0211	0232-0234	0246-0246
025F-0263	0283-0283	02A8-02E1	02E5-03C1	03EA-03EA
0469-046A	0495-0495	049F-049F	04B8-04B8	050D-050D
0533-0533	05D0-05D0	05D4-0673	0708-124B	124E-1917
1919-191B	191D-1B5C	1B60-206C	206E-2072	2074-2074
2076-2076	20B9-20BB	20BD-20F6	20F8-213A	213C-2166
2168-2269	226F-2B84	2B89-2B8B	2B8F-2B90	2B94-2C1E
2C43-2C79	2C80-2C80	2C83-2C89	2C8D-2FFB	—

Table 1. Model I and Model III ROM comparison (start address-end address).

Chapter VI
Games

Golf Tee Puzzle Madness

David A. Bennett



I was visiting relatives during a holiday, and had just finished a sumptuous feast when it began. My father-in-law handed me a small triangular piece of wood with 15 regularly spaced holes and 14 pegs resembling golf tees, and said, "You have a logical mind; see if you can jump the pegs over each other until only one peg remains."

Of course, having a logical mind, I couldn't resist this sinister trap and proceeded to spend the better part of a Saturday afternoon mumbling to myself and fending off invitations from my family to break away from the puzzle's deadly spell.

After this initiation I came to two realizations: I had spent a great deal of time and had actually found a solution to the puzzle; and a computer could have spent less time and found many solutions to the puzzle. This article presents the results of my resolve to teach a computer to manipulate those infuriating pegs.

This game is representative of a broad class of games which can be mastered by a computer through the construction of a tree of moves and counter-moves. Most such games are either too simple to be of interest to human players, or too complex to be solved completely through construction of a tree. Therefore, this game finds a happy medium where one can bounce back and forth between studying a mechanical solution and studying a human solution, and not be bored or confused by either.

The program and accompanying notes presented in this article will thus satisfy three goals:

- It illustrates a computer generated answer to a rather complex question. The machine demonstrates "artificial intelligence" in the sense that it finds puzzle solutions which people are not "smart" enough to find themselves. The irony of such artificial intelligence is that it works not through brilliant insight, but through the tedious application of well-specified, but simple logic.
- It teaches about the representation of board game strategy. Many other games may be programmed using the same building blocks of tree generation, pruning rules, move generators, board evaluators, and so forth. (Of course, other game implementations will require entirely different versions of these modules. However, the concepts are similar and the basic structure may remain the same.)
- It illustrates some fascinating programming concepts, such as recursion and graphic animation.

After reading this, you will have been exposed to some fundamental computer game simulation concepts. These will give you insight into how others have programmed games and provide the tools to create your own simulations.

The remainder of the article contains a brief historical perspective on games and computers, a synopsis of the puzzle, a detailed presentation of the way the program plays the game, and finally some enlightening conclusions.

Artificial Intelligence, Checkers, and Cannibals

"Artificial Intelligence" (AI) means many things to many people. M. L.

David A. Bennett, 76 Prospect Hill Ave., Summit, NJ 07901.

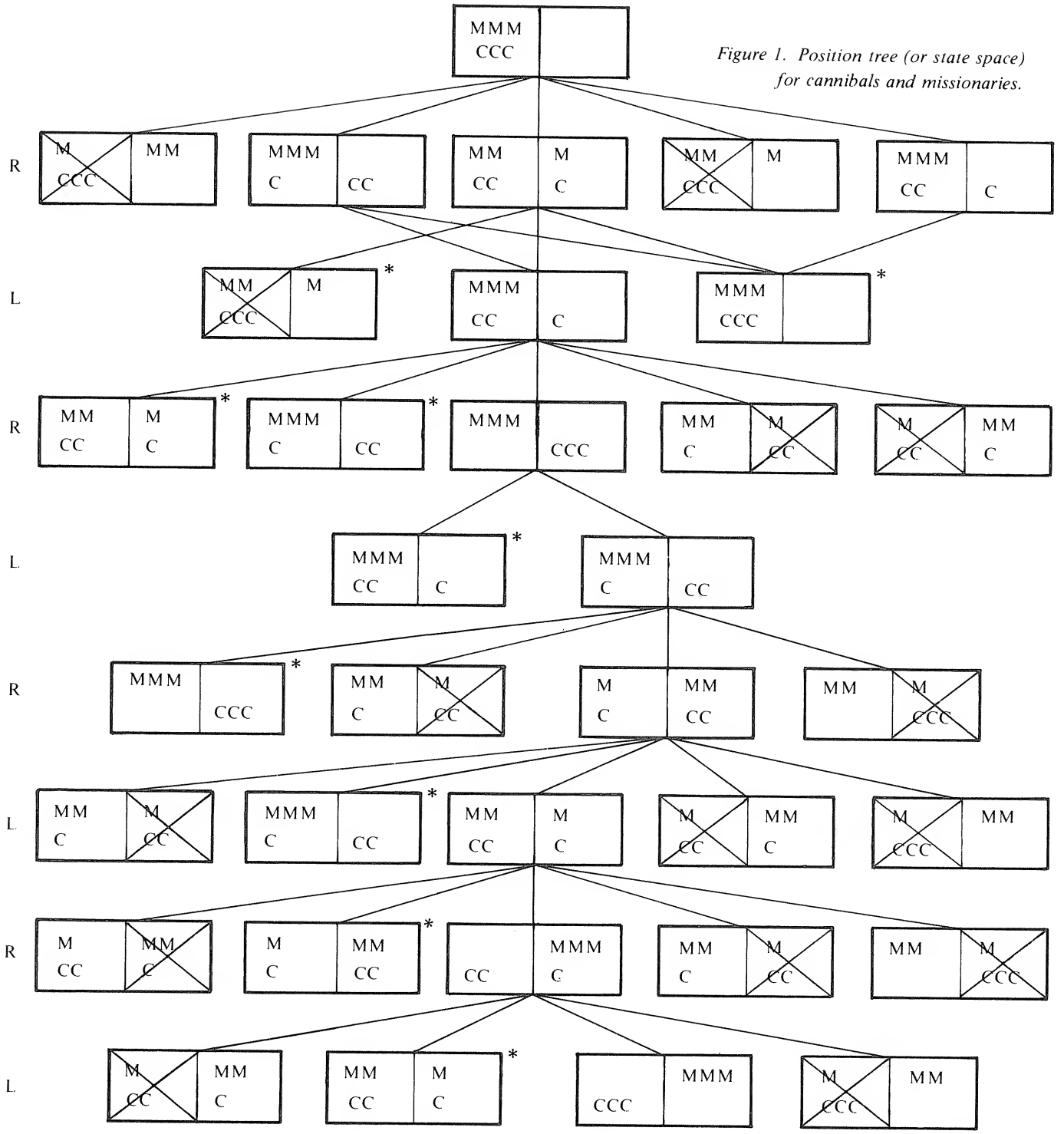


Figure 1. Position tree (or state space) for cannibals and missionaries.

Minsky, one of the leaders in the field, has said that "artificial intelligence is the science of making machines do things that would require intelligence if done by men."

This is a very broad definition, and certainly in the early days of AI research the programming of machines to play games such as chess and checkers was considered mainstream stuff.

One of the first and most successful AI projects was a program by Samuels to play checkers. It had the ability to remember all games it had played and learn from its mistakes.

It played by examining possible future moves and evaluating each board position derived. The quality of each position was a function of the rules concerning piece deployment and the quality of boards reachable from the current board.

As it accumulated a history of play, the relative weights of future boards dominated its evaluation. The program quickly surpassed its author in skill, and moved on to defeat checkers masters.

One secret to Samuels' success was the organization of board positions into a tree.

Each board position was a node in the tree, and positions obtainable through a single move were stored as subnodes under the starting node.

Trees in most games are incredibly large, and cannot be examined exhaustively. Therefore, programmers introduce *heuristics* to teach the program to make intelligent guesses in the absence of complete information.

A typical heuristic for Samuels' checkers program was to examine whether the machine controlled the center of the board. Center control makes the position stronger.

In the case of the golf tee puzzle program, heuristics are not used. The program assumes that I will let it run long enough to examine all positions. While the search is certainly large, this approach is feasible for a small game.

The idea of a complete search of all positions (or the *state space* of the game) is nicely illustrated by the familiar cannibal and missionary problem. Figure 1 shows the complete tree for this problem.

Recall the situation: three cannibals and three missionaries are on the left bank of a river, and they have a two-person boat. The missionaries wish to cross the river, but are deterred by the fact that if missionaries are outnumbered by cannibals they will be eaten.

In Figure 1, the letters L and R indicate whether the boat is on the left or right bank of the river. A star (*) next to a node shows that the node (and its subtree) have already been explored and need not be considered further.

The topmost box shows all six people and the boat on the left bank. There are five possibilities from here; two of them result in the immediate demise of one or more missionaries. The other three are to take a cannibal, a missionary, or one of each across the river. Following the tree down to the bottom yields a solution after eight river crossings. (Note that the cannibals could all cross the river in only two additional moves.)

A similar tree will be built by the golf tee program to find a path from the 14-peg starting configuration to the one peg solution. At each position, the possible next moves are considered, and used to produce additional positions. The tree is not actually constructed in the memory of the computer, but examined one small piece at a time, by systematically hopping from one branch to the next. But I digress . . . more about this later.

There are two approaches to constructing trees: depth first and breadth first. A depth-first construction takes a path from the top straight down to the bottom, and examines only a vertical slice at any one time. The specific path chosen is not necessarily based on evaluating the "best" choice at each level, but focuses on rapid motion toward an end position.

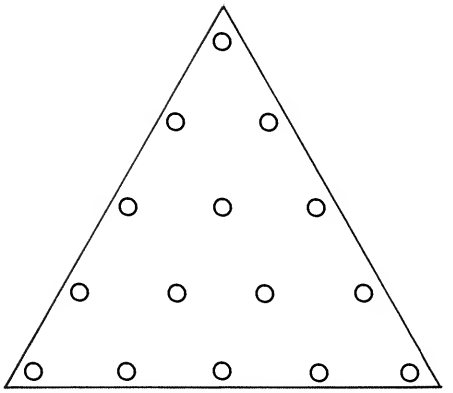


Figure 2. The board.

A breadth-first approach, on the other hand, takes a look at all nodes of the current level, moving downward only to examine the entirety of the next level.

A breadth-first search requires enough memory to hold information about all nodes at the current (and possibly several past) levels. Depth-first, on the other hand, requires memory only to hold a node for each level passed through. This can be an important concern, particularly to users of small computers. The golf tee algorithm uses a depth first algorithm.

The Puzzle

The puzzle is played with some very simple equipment, and can be duplicated at home if desired. The board is a small triangular wooden block measuring about 6″ per side and ¾″ to 1″ thick. Set into this board are 15 holes in an arrangement shown in Figure 2.

To play, 14 pegs (golf tees will do nicely) are placed into 14 holes of the player's choice. Because of the symmetry of the board, there are only four unique starting configurations, but solutions are by no means equally distributed among the four configurations.

A move consists of jumping a peg from its current position over another peg and into an empty hole. The destination hole becomes filled with the jumping peg, and the peg which was jumped over is removed.

Jumps may take place only in a straight line on the board, but may go in any direction. Thus, if you start with the top corner open there are only two possible moves: you may bring up either the left or right hand center edge peg to the top corner hole, removing one peg from the second row.

You win if you jump in a sequence such that only one peg remains on the board after 13 moves.

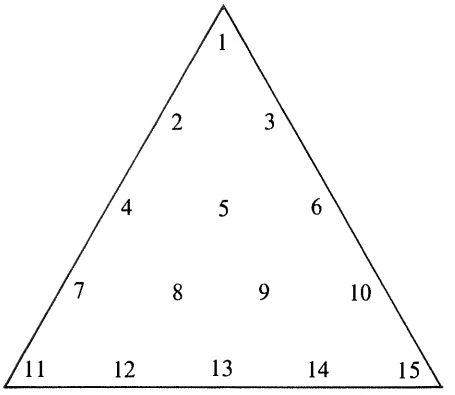


Figure 3. The hole encoding.

A novice will soon discover that it is quite easy to end up with three pegs on the board. This is confirmed by computer analysis.

Approximately one half of all sequences from all the starting positions end up with three pegs on the board. To get two pegs requires some fiddling, and to pare the board down to one peg calls for nothing short of dogged persistence.

After some trial and error, intermediate positions can be catalogued mentally, to be quickly reached from new positions. While the creature here is certainly not in the same league with Rubik's cube in complexity, it shares with the cube the property that many solutions are not obvious even when the current position is within a few moves of winning.

The Data Model

If the machine is to play the game, it must have an internal model of such things as holes, pegs, jumping rules, and so forth. This model, or encoding, should be efficiently accessible by the Basic interpreter.

The model chosen has three com-

Figure 5. Data model sequence.

Board Position	Move
.TTTTTTTTTTTTTT	(1,2,4)
T.T.TTTTTTTTTT	(6,5,4)
T.TT..TTTTTTTTT	(1,3,6)
...T.TTTTTTTTT	(7,4,2)
.T...T.TTTTTTT	(10,6,3)
.TT....TT.TTTTT	(12,8,5)
.TT.T...T.T.TTT	(13,9,6)
.TT.TT....T..TT	(2,5,9)
..T..T..T.T..TT	(3,6,10)
........TTT..TT	(15,10,6)
.....T..T.T..T.	(6,9,13)
..........T.TT.	(14,13,12)
..........TT...	(11,12,13)
............T..	

ponents: an encoding of the holes of the board, a representation for an instance of a board position, and "legal move arrays" defining legitimate peg jumps.

The encoding of the 15 holes is illustrated in Figure 3. The holes are numbered from 1 to 15. The fact that hole 1 is on top is arbitrary, as the board is symmetrical about the three corners.

I have chosen a 15-character string to represent a specific peg distribution. Each character is either a T or a period, meaning peg and no-peg respectively. The leftmost character in the string corresponds to hole number 1; the rightmost to hole number 15.

The use of strings makes a board directly printable, a handy feature while debugging programs. The string functions LEFT$, RIGHT$, and MID$ are used to access individual hole positions. For example, the string "TTTTTTTTTTTTTT." is a board with all positions filled but the lower right hand hole.

Are there other choices than the use of strings? Certainly. For starters, an integer will hold 16 bits of information, and the board contains 15 bits of information. Therefore, a very compact and seemingly elegant scheme would call for each bit in the integer to represent one hole in a board.

The machine could hold thousands of such boards with room to spare. (The machine I used to prepare the program, a TRS-80 Model III, has 24K or 16-bit words of memory, which is enough for a large number of peg boards.) There is even a leftover bit in each word as a bonus, available to use in some as yet unspecified but clever way.

Unfortunately, accessing the bits from TRS-80 Model III Basic is quite awkward. Furthermore, the algorithm does not require storage of more than one or two dozen board positions at any one time. It thus cannot take advantage of the compactness of the notation. Due to these drawbacks, I rejected the "bit per hole" encoding. Note that if the algorithm had needed to store more than just a few board positions, the storage savings inherent in this approach would have made it much more attractive.

Another possibility is to use arrays of integers where subscript 1 corresponds to hole 1, subscript 2 corresponds to hole 2, etc. Access to specific positions is here quite straightforward. In terms of space, each peg requires 2 bytes since there are no 1-byte integers in the TRS-80 Basic. This would not be objectionable; the only serious drawback is that they cannot be referenced in their entirety with a single name.

A better representation would be possible if the language included user defined and enumerated data types, such as are found in Pascal. Variables could then have been declared as shown in Figure 4.

Figure 4.

```
type board = array [1..15] of holes;

     holes = (peg, empty);

     game_stack =

          record pointer: int;

                 positions: array [1..STKSIZ] of board;

                 end;

var puzzle: board;

    stack: game_stack;
```

This declares a primitive data type of "board," which can then be used as easily as integers, reals, and strings. "Holes" become primitive constructs which can take on only two values: peg or empty. Holes can be thought of as boolean variables with the two states being "peg" or "empty" instead of true or false.

A game stack is an array of boards along with a stack pointer. This logical grouping makes the programmer's intent clear from the outset with regard to the stacking of game positions during play.

Using this data representation, I would reap the benefits of indexed access, compiler-dependent space or time optimization, and mnemonic object reference all in one fell swoop.

Using the string data model, Figure 5 illustrates a sequence of string values which constitute a game solution. The notation for each move consists of three numbers. The first is the position of the jumping peg, the second is the hole which is jumped over, and the third is the position in which the jumping peg ends up. In a legitimate move, the peg in the middle position is removed.

It turns out that there are exactly 36 such legitimate moves, and they are encoded in three arrays of length 36 each. They are called START, OVER, and FIN in the program. For a particular board configuration, fewer than 36 moves will be applicable.

The move generator is thus quite simple; each of the 36 possible moves is compared to the current board. If the START and OVER holes have pegs in them and the FIN hole is empty, then that move can be taken.

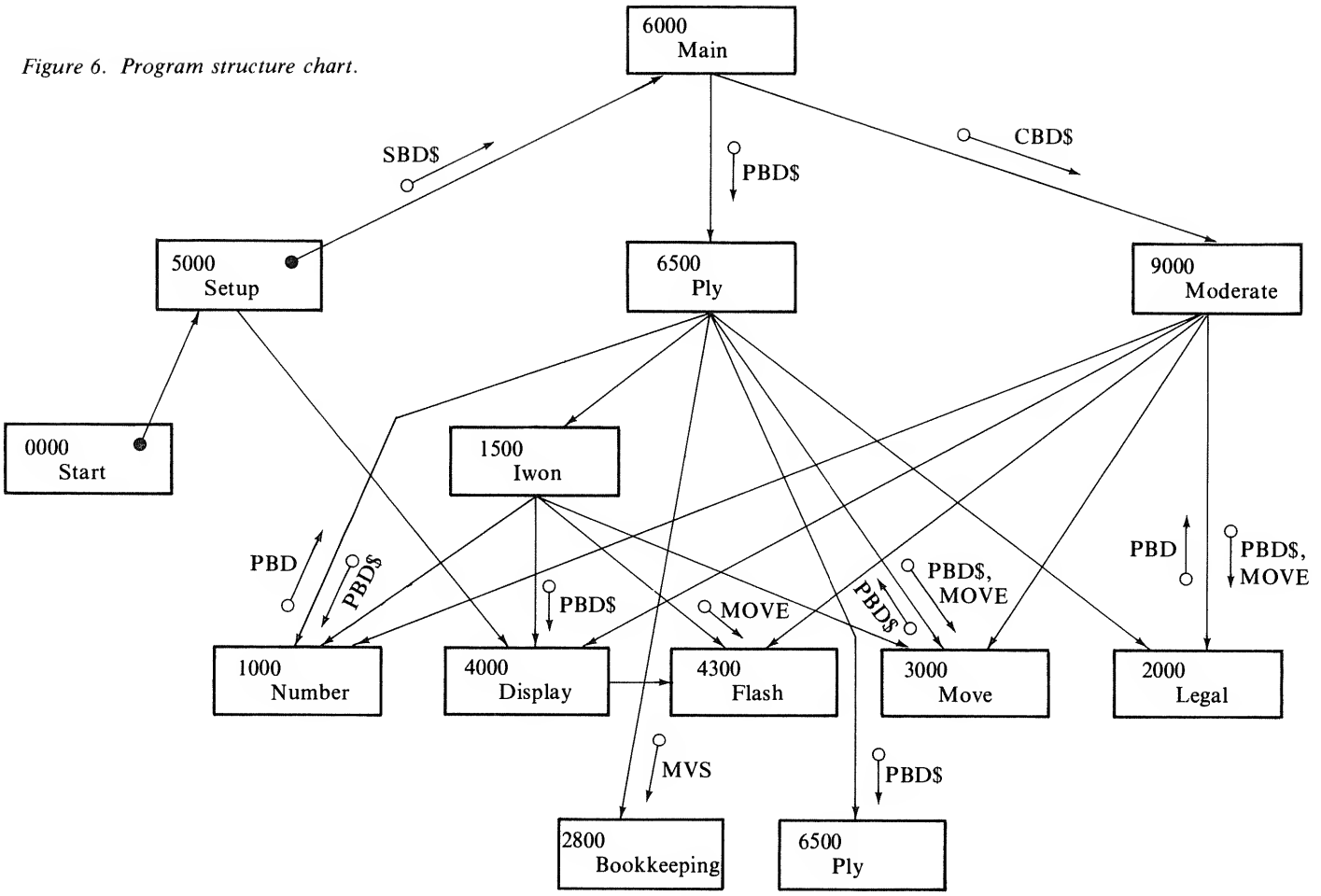
Program Logic

Figure 6 is a Yourdon type structure chart showing the modules comprising the program and how they relate to each other. Arrows connecting boxes indicate that the box pointed to is called or transferred to by the other box; arrows labelled with variable names show that the variable is either being passed into a subroutine or returned as a function value.

Only variables of major importance are actually drawn on the chart, as a complete diagram would be unnecessarily cluttered. Table 1 is a companion to the structure chart summarizing the function of each module. Table 2 describes the major program variables.

In Basic, module boundaries and calling parameters are somewhat artificial. Even in Basic, though, all is not lost. The systematic use of dummy module declarations helps to give the feeling that the whole program is composed of many independent and cohesive "building blocks."

Figure 6. Program structure chart.



Main

Referring to Figure 6, the Main routine is simple, as it should be. Lines 6000 through 6140 suffice to drive the entire game. These establish which mode is desired by the user, and then either GOSUB or GOTO other code to implement that mode. When the play ends, Main prints out summary statistics.

Start and Setup

You have now mastered the entire top level of the structure chart. Moving right along, the first module in level 2 is Start. Start executes a few DIM and initialization statements, and then jumps directly to Setup. This may seem a bit unstructured, but you have to start somewhere, right?

Setup does considerably more work than start did, so things should begin to get interesting. The first step in Setup is to read in from DATA statements the values of the START, OVER, FIN, and PAT arrays. Recall that corresponding elements of START, OVER and FIN define a legal peg jump sequence.

The numbers in these arrays are therefore the embodiment of the knowledge the program has regarding how to play the game. The PAT array contains the screen co-ordinates of each of the 15 hole positions. It is used by Display and Flash.

Statement 5140 is worthy of mention here, as it makes this part of the program "self checking." The statement reads in an additional data item with a uniquely recognizable value.

If a typographical error has occurred in the previous data list, then this unique data value will not be read into the check variable. (Numeric transpositions will not be trapped by this test, unfortunately. I leave it as an exercise for the reader to construct a checking scheme which can detect these and other errors.)

Setup makes use of the INKEY$ function to perform I/O in an unusual way. I wanted Setup to blink the picture of each hole on the computer screen while the user was deciding whether to put a peg there or not. A quick scan of the TRS-80 manual convinced me that Radio Shack did not provide a "blink" mode.

Further, if the program asks the user for a response through the INPUT statement, it cannot also blink any displays. What to do? INKEY$ to the rescue.

INKEY$ is a built-in function which returns immediately with the value of the most recently pressed key on the keyboard. If a key has not been pressed since the keyboard was last sampled, INKEY$ returns an empty string. Otherwise, INKEY$ returns the character pressed as a string of length one.

This allowed me to write the logic between 5345 and 5480 where the computer sends out alternating light and dark characters to continuously blink on the screen while waiting for keyboard input.

Ply

The Ply module is the most fascinating of all the modules. It contains the heart of the game model. The job of Ply is to take a single board configuration and generate from it all possible ways to play that board out to end positions. The algorithm can be concisely summarized with the piece of

pseudo code in Figure 7.

Notice that Ply can only do real work when it is handed a board from which no more moves can be made. Under these circumstances, it figures out whether play is blocked because you have found a solution or because there are simply no moves to be made.

These two cases are reported differently, if the board is a solution, a special module is invoked (Iwon) to celebrate appropriately. Iwon has several options having to do with the way the user wishes to record each victory. The most fun is to have Iwon replay with animation the entire winning sequence one move at a time.

If you are not looking for any single win, but the accumulation of a great many winning games, another option allows you to print out sequence summaries on the screen or the printer (if you have one).

Whenever a solution is found, the variable SOLS is incremented. If the board has more than one peg left, then you have simply become stuck. This is recorded in the variable EDGMS (EnDGaMeS).

EDGMS has one element for each ply of the game tree, and the value at each level represents the number of times a board was generated at that level from which no further moves could be made. (At level 14, this value is equal to SOLS.)

Let's go back to the way Ply deals with boards which are not end games. If any possible moves from the board are passed into Ply as a parameter, Ply decides that the situation is too complicated for it to handle by itself, and it calls on some helpers to work on sub-problems.

Ply knows that there are never more than 36 possible moves from any one board. It therefore divides the problem into 36 smaller problems, one for each new board produced by applying a move to the board it was given. The helpers it needs turn out to be copies of itself.

This is perfectly reasonable, since Ply is

Table 1.

Program Module Summary

Line Number	"Declaration"	Function
0000	Start	Declaration of variables; DIMs; Identifies comments and version.
1000	Number(PBD$)	Returns in PBD the number of pegs in PBD$.
1500	Iwon	Called by Ply whenever a winning board is generated.
2000	Legal(PBD$,move)	Returns PBD=1 if "move" applied to PBD$ is legal, PBD=0 otherwise.
2800	Bookkeeping()	Called by Ply to gather statistics whenever a legal move is generated.
3000	Move(PBD$,move)	Applies "move" to PBD$, and returns the new PBD$.
4000	Display(PBD$)	Draws PBD$ as a graphic on the CRT.
4120	Redraw(PBD$)	Same as Display, but does not clear screen or draw enclosing triangle.
4300	Flash(move)	Using animation, performs "move" on the CRT; assumes board image is already on screen.
5000	Setup	Reads in legal move array and gets an initial peg configuration from the user.
6000	Main	Driver for mainline logic; starts play in either of the two modes.
6500	Ply(PBD$)	Generates one ply of the game tree by invoking itself for each subtree under the current level.
9000	Moderate	Logic to interactively moderate an attempted solution by the user.

Table 2.

Major Program Variables

Name	Function
	** Legal Move Definition **
START(36),OVER(36),FIN(36)	These three arrays define which peg jump combinations constitute legal jumps. For example, START(1) = 1, OVER(1) = 2, and FIN(1) = 4, which means that peg 1 can jump over (and remove) peg 2 landing in peg slot 4.
	** Automatic Variables **
CBD$(14)	The "Current BoarD" array holds a copy of the board generated for each level of recursion.
MOVE(14)	The "move" array holds the move currently under consideration at each level of recursion.
EDGMS(14)	The "EnD GaMeS" array is a counter of the number of games for which there were no sub trees at each level of recursion.
STKPTR	The "STacK PoinTeR" keeps track of the current level of recursion.
	** Miscellaneous **
PAT(15)	The address in CRT "PRINT @" co-ordinates of each hole in the board.
PBD$	A string used to contain a copy of the current board when passed as a parameter to a subroutine or function.
SOLS	Counts the number of solutions found so far.
MVS	Counts the number of pegs moved so far.
HRDCPY	A flag to represent whether hard copy on the printer is desired. 1 means produce hard copy, 0 means no hard copy.
CRT	A flag to represent whether discovered solutions should be flashed on the CRT when found or not. 1 means YES, 0 means NO.
PSREPORT	An integer meaning the interval between each status report on the CRT. Status reports are helpful in assessing the progress of the program as it goes along.
REPORT	An integer representing the interval at which discovered solutions are to be reported.

defined to analyze all games playable from a given starting configuration. Each helper copy of Ply sees the same situation as its caller, namely a board configuration and instructions to find all playable games.

Helper configurations are always *exactly* one peg smaller than their caller configurations. (Convince yourself that this is the case.) Since configurations must have between 0 and 15 pegs, no more than 14 levels of helpers will be called. Actually, a board with 15 pegs on it is stuck, so 13 is the true limit of depth of calling.

To summarize, each copy of Ply will examine its board: if there are no moves to be made, Ply performs appropriate notification. If there are some moves, then it calls a helper copy of Ply for each move, and gives the helper copy the board implied by making that move on the board of the current Ply.

At this point it is appropriate to reflect on the nature of the Ply algorithm. There was talk at the beginning of the article on the subject of representing the "state space" of the game as a tree, and then traversing the tree in depth-first order. Well, do you see any declarations of "TREE" data structures in the code?

The fact is that there are no explicit declarations of any variables taking the shape of a tree. (There is a stack, but stacks and trees are different.)

The Ply algorithm develops the tree *implicitly* by the order in which it generates board positions and invokes helper copies to analyze them. This is perfectly all right, and as it happens, *not* making a copy of the entire state space is what allows the program to fit into a TRS-80 in the first place. The representation of legal moves as an ordered sequence of array elements is what lets the program keep track of which branch of the tree is being considered by which copy of Ply.

There are two problems with this kind of algorithm written in Basic that I have not yet mentioned. The first is keeping track of subroutine return addresses (remember, Ply must return into itself), and the second is providing each copy of Ply with its own copy of its variables. These are not as formidable as they might at first seem.

The return address problem has been thoughtfully solved for us by the Tandy Corporation. To wit, return addresses from GOSUBs are pushed onto an internal stack, such that returns go to callers in the inverse order from which they were called. All such addresses are treated as 16-bit numbers, and the machine does not care whether the programmer nests calls or not.

The second problem, that of providing each copy of Ply with its own variables, requires some programming to solve. In a manner analogous to the handling of return addresses, Ply keeps a stack of variables whose values must differ in each copy of the module.

```
procedure ply(PBD$):
        if "there are no more moves possible on PBD$"
           then begin if "there is only one peg left"
                      then "you win" else "you're stuck"
                      end
           else for each possible move  do
                    ply(move(PBD$))
         end
```

Figure 7.

Upon entry to Ply, these variables are preserved at the top of the stack and new values are created for the copy just entered. The saved values represent the variable context of the caller copy of Ply; the new values are used by the current (or helper) copy.

When Ply exits, the reverse process takes place. Copies of the variables are fetched from the stack and made the current copies. Thus, as far as each copy is concerned, the values of its variables are not affected by its helpers.

In the listing, these variables are noted as "automatic" variables, since in other languages this term is used to describe the fact that space is automatically allocated from the stack at each entry.

The net effect of the above algorithm is to perform a depth-first walk through an n-ary tree representing all possible board configurations reachable from the starting board. For any 14-peg configuration, this tree is quite large.

It might be reasonable to redefine the problem statement to allow the program to find all solutions which are "significantly" different from each other. This would result in what is commonly called "pruning" the tree.

Many game playing programs have a component missing in this one. That is, as each board position is generated, a module is invoked to assign a weight to that board. The weights serve to rank the generated boards in terms of how likely a board is to lead to a winning position later.

In this case, there are no weights as such, but it does make sense to talk about groups of solutions which are so similar that as far as the player is concerned, producing any one of them is adequate to describe them all.

If solutions (and entire sub-trees) could be so organized that only a single prototypical sequence was produced for each group of similar sequences, then this grouping would effectively prune the tree. Under these conditions, the program would run to completion in substantially less time than under the simpler algorithm described above, particularly if the pruning could take place at the upper levels of the tree.

Pruning

Such a pruning strategy consists of having the Legal module recognize certain classes of board symmetry, and return as legal only those moves on a single side of the symmetric relation. Figure 8 shows an example of symmetric peg configuration. Regardless of whether move 1 or move 2 is made, subsequent play remains the same.

The two resulting subtrees of configurations are identical except for left-right symmetry, and need not both be considered. There are bilateral symmetries around the three perpendicular bisectors of the three edges. (No pruning strategy is implemented in the program.)

Another pruning strategy (unfortunately not implemented in the program either) is a mechanism for remembering board positions which have already been examined and skipping over them. When you consider the effects of rotation and symmetry, there are likely to be many large subtrees currently examined by the program unnecessarily. This kind of strategy is complicated by the fact that accessing and comparing the current board to previous boards can be extremely time-consuming. The solution will require not only a compact storage notation for boards, but an efficient indexing scheme for relating new positions to old ones.

To set Ply into the context of other game

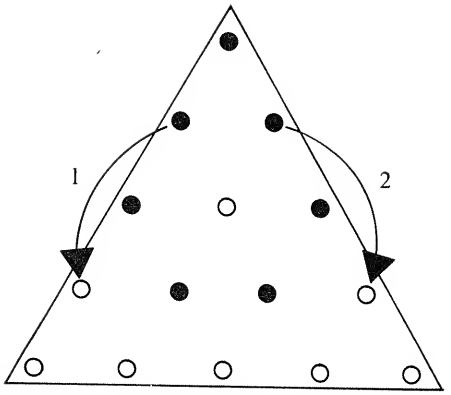


Figure 8. Symmetric board showing possible pruning.

simulations, it has been shown to have a data model, a move generator, a tree constructor, no board evaluator, and no implemented pruning strategy. The data model is the string representation of the board; the move generator is a FOR-NEXT loop in combination with the legal move arrays; the tree constructor is the recursive Ply logic; and the board evaluator is not necessary since a complete tree traversal is being made. A pruning algorithm has been omitted for the sake of exposition; including one would substantially enhance the program.

Moderate

The Moderate module (the final module of structure chart level 2) is relatively simple. It is selected by the user instead of Ply when the program commences. Its algorithm is as follows:

1. Display the starting board on the CRT.
2. Discover from Legal all possible moves from the current board, and print them on the bottom of the screen.
3. If there were no possible moves, then stop and print out whether the player won or lost.
4. Prompt for and get from the player a possible move.
5. Flash the move on the screen, update the current board, and go back to step 2.

If you have followed me so far, you have completely mastered the second level of the structure chart in Figure 6. If you can't remember anything about the structure chart of Figure 6, then you probably fell asleep somewhere back in the last few columns and you should probably go on to the next article. Otherwise, I will wrap up the program logic with a few words about some of the building block modules on lower levels.

```
10 REM   --- GOLF TEE PUZZLE ---
20 REM
30 REM   DAVID A. BENNETT
40 REM   WRITTEN JAN 1980
50 REM   REVISED FOR TRS MODEL III BASIC, JULY 1981
60 REM
61 REM  THIS PROGRAM GENERATES SOLUTIONS TO THE TRIANGULAR
62 REM  GOLF TEE PUZZLE (OFTEN FOUND IN RESTAURANTS IN THE
63 REM  NORTHEAST).  IT HAS TWO MODES: 1) FINDS  ALL  SOL-
64 REM  UTIONS GIVEN ANY STARTING CONFIGURATION,   AND  2)
65 REM  LETS A USER TRY TO FIND HIS/HER  OWN SOLUTIONS  BY
66 REM  ACCEPTING PEG MOVES INTERACTIVELY, AND THEN  SHOW-
67 REM  ING THEIR EFFECTS ON AN "ELECTRONIC BOARD."
68 REM
69 REM                          VERSION 2.2   8/9/81
70 REM       --- DECLARATION OF ARRAYS ---
80 DIM PAT(15),START(36),OVER(36),FIN(36)
90 DIM CBD$(14),EDGMS(14),MOVE(14),ILLEGALS(14)
100 REM     --- GLOBAL VARIABLE INITIALIZATION ---
110 TRUE=1: FALSE=0
120 STKPTR=0    :REM  STKPTR IS THE CURRENT LEVEL OF RECURSION
130 SOLS=0      :REM  SOLS IS THE NUMBER OF SOLUTIONS FOUND
140 MVS=0       :REM  MVS IS THE NUMBER OF PEG MOVES MADE
150 GOTO 5000
1000 REM "NUMBER(PBD$)"
1010 REM  RETURNS IN PBD THE NUMBER OF PEGS LEFT IN PBD$
1020 PBD=0: FOR J=1 TO 15
1030 IF MID$(PBD$,J,1)="T" THEN PBD=PBD+1
1040 NEXT J
1050 RETURN
1500 REM "I WON!"
1510 REM  INVOKED BY "PLY" WHEN A WINNING GAME IS FOUND
1520 SOLS= SOLS+1
1530 IF SOLS/REP <> INT(SOLS/REP) THEN RETURN
1540 IF HRDCPY <> TRUE THEN GOTO 1630
1550 PRINT " "
1560 PRINT "------- WIN NUMBER ";SOLS;"  MOVE NUMBER ";MVS;"--------"
1570 LEVEL=1
1580 PRINT "MOVE #";LEVEL;":";START(MOVE(LEVEL));"->";FIN(MOVE(LEVEL))
1590 PRINT "BOARD";CBD$(LEVEL);" MOVE";START(MOVE(LEVEL));"->"
     ;FIN(MOVE(LEVEL))
1610 LEVEL=LEVEL+1
1620 IF LEVEL < 15 AND CBD$(LEVEL)<>"" THEN GOTO 1580
1630 REM  ... NOW FOR THE CRT
1640 IF CRT <> TRUE THEN RETURN
1650 INPUT "READY FOR CRT DISPLAY OF WIN";X$: CLS:LEVEL=1
1654 PRINT @0,"SOLUTION #";SOLS;
1655 PBD$= CBD$:  GOSUB 3000
1657 CBD$(STKPTR+1)= PBD$: MOVE(STKPTR)=MOVE
1660 PBD$= CBD$(1)
1670 GOSUB 4000
1680 MOVE=MOVE(LEVEL)
1690 PRINT @896,"HIT <ENTER> WHEN READY FOR NEXT MOVE";
1700 INPUT T$
1710 GOSUB 4300 : REM FLASH THE MOVE
1720 PBD$= CBD$(LEVEL+1)
1730 GOSUB 1000 : REM COUNT THE PEGS AGAIN
1735 LEVEL=LEVEL+1
1740 IF PBD <> 1 THEN GOTO 1680
1760 PRINT @896,"THE SEARCH CONTINUES...
1770 RETURN
2000 REM "LEGAL(PBD$,MOVE)"
2010 REM RETURN IN PBD 0 IF MOVE IS ILLEGAL, 1 IF LEGAL
2020 PBD=0
2030 IF MID$(PBD$,START(MOVE),1)="."THEN RETURN
2040 IF MID$(PBD$,OVER(MOVE) ,1)="."THEN RETURN
2050 IF MID$(PBD$,FIN(MOVE)  ,1)="T"THEN RETURN
2060 PBD=1: RETURN
2800 REM "BOOKKEEPING ON EACH LEGAL MOVE"
2810 MVS=MVS+1
2820 IF INT(MVS/PSREPORT)<>MVS/PSREPORT THEN RETURN
2830 PRINT "STATUS:"
2840 PRINT MVS;"MOVES,";SOLS;"SOLUTIONS,";
2850 X=0: FOR J=1 TO 14
2860 X=X+EDGMS(J) : NEXT J
2870 PRINT X;"GAMES COMPLETED."
2880 RETURN
3000 REM "MOVE(PBD$,MOVE)"
3010 REM   CHANGES PBD$ TO REFLECT MOVE
3020 TB$=LEFT$(PBD$,START(MOVE)-1)+"."+RIGHT$(PBD$,15-START(MOVE))
3030 TB$=LEFT$(TB$,OVER(MOVE)-1)+"."+RIGHT$(TB$,15-OVER(MOVE))
3040 PBD$=LEFT$(TB$,FIN(MOVE)-1)+"T"+RIGHT$(TB$,15-FIN(MOVE))
3050 RETURN
4000 REM "DISPLAY (PBD$)"
4010 REM  DISPLAY THE BOARD ON THE CRT
```

The Building Blocks

The Number and Legal routines are both quite simple, so they will be dealt with first. Number is an integer valued function (not bad for Basic) which returns the number of pegs in a board configuration passed to it as a parameter. It uses a FOR-NEXT loop and the substring extraction function MID$ to isolate and count pegs in a string variable array.

Legal is a boolean valued function (that is, it returns one of two values) indicating whether a specific move can be made on a specific board configuration. Of course, the words "function" and "parameter" must be taken with a large dose of salt; however, I feel it is essential to follow reasonable programming conventions in order to make the components of a software system independent of each other, even when the conventions are not encouraged by the interpreter at hand.

Move is a module of two parameters: a board configuration and an integer representing a move. It modifies the board parameter to reflect the fact that the move has been made. Note that Move does no error checking, and as a result will make some pretty outrageous moves if you ask it to. Bookkeeping is called only by Ply, and its job is to do whatever seems appropriate to note the passing of a move. Right now it counts them, and will provide a status report on the progress of the program through the state space at regular intervals.

The final (and perhaps most interesting) module in the building block collection is the Display routine. This is because it contains a modification of Bresenham's algorithm to interpolate points on the line used in drawing a triangle on the CRT. An interpolation algorithm is needed because I wanted to draw a picture of the triangle on the CRT.

Such an algorithm approximates straight lines of arbitrary slopes on plotting devices with point addressability. The alternative would be to encode each point of the line separately in a DATA statement. I felt this to be prohibitively tedious.

The basic algorithm can be expressed in Pascal in Figure 9, and assumes e, x, and y real, and deltax, deltay are integers.

The version used in the program has been converted to use integers exclusively, and has the parameters factored in as constants.

A note about the TRS-80 Model III graphics is in order here to help explain Display. The CRT can be directly addressed under program control. That is, a programmer may position text anywhere on the screen, and may also selectively turn on or off any single picture element (pixel). Text is positioned using the "PRINT @" command (see line 4130).

```
4020 REM  FIRST, DRAW AN ENCLOSING TRIANGLE
4030 FOR J=35 TO 95: SET (J,35): NEXT J
4040 X=95: Y=35: DY=9
4050 SET(X,Y): X=X-1: Y=Y-1: DY=DY-1
4060 IF X=64 THEN GOTO 4080
4070 IF DY<>0 THEN GOTO 4050
4075 DY=9: X=X+1: GOTO 4050
4080 X=64: Y=3: DY=9
4090 SET(X,Y): X=X-1: Y=Y+1: DY=DY-1
4095 IF X=34 THEN GOTO 4110
4100 IF DY<>0 THEN GOTO 4090
4105 DY=9: X=X+1: GOTO 4090
4110 REM -- NOW FILL IN PEGS --
4120 FOR J=1 TO 15
4130 IF MID$(PBD$,J,1)="T"THEN PRINT@PAT(J), CHR$(171); ELSE PRINT @PAT(J),
     CHR$(160);
4140 NEXT J
4150 RETURN
4300 REM "FLASH(MOVE)"
4310 REM   THIS PROCEDURE FLASHES THE BOARD ON THE CRT TO INDICATE
4320 REM   A MOVE MADE, LEAVING THE CRT AS IF THE MOVE
4330 REM   HAD BEEN MADE.
4340 A(1)=START(MOVE): A(2)=OVER(MOVE): A(3)=FIN(MOVE)
4350 FOR J=1 TO 3
4360 FOR J1= 1 TO 3
4370 PRINT @PAT(A(J)),CHR$(191);
4380 FOR J2=1 TO 5: NEXT J2
4390 PRINT @PAT(A(J)),CHR$(128);
4400 FOR J2=1 TO 5: NEXT J2
4410 NEXT J1
4420 IF J=3 THEN PRINT @PAT(A(J)),CHR$(171); ELSE PRINT @ PAT(A(J)),
     CHR$(160);
4430 NEXT J
4440 RETURN
5000 REM "SETUP INITIAL ARRAYS, GET START FROM USER"
5010 REM   FIRST, THE LEGAL MOVE ARRAYS
5020 DATA 1,1,2,2,3,3,4,4,4,4,5,5,6,6,6,6
5030 DATA 7,7,8,8,9,9,10,10,11,11,12,12,13,13
5040 DATA 13,13,14,14,15,15
5050 FOR J=1 TO 36: READ START(J): NEXT J
5060 DATA 2,3,4,5,5,6,2,5,8,7,8,9
5070 DATA 3,5,9,10,4,8,9,5,8,5,6,9
5080 DATA 7,12,13,8,12,8,9,14,9,13,14,10
5090  FOR J=1 TO 36: READ OVER(J): NEXT J
5100 DATA 4,6,7,9,8,10,1,6,13,11,12,14
5110 DATA 1,4,13,15,2,9,10,3,7,2,3,8
5120 DATA 4,13,14,5,11,4,6,15,5,12,13,6
5130 FOR J=1 TO 36: READ FIN(J): NEXT J
5140 DATA 1234: READ J
5150 IF J<>1234THEN PRINT "DATA CHECK AT LINE 5150":GOTO 9999
5160 REM NOW GET VALUES FOR PEG SCREEN ADDRESSES
5170 DATA 160,286,290,412,416,420,538,542,546
5180 DATA 550,664,668,672,676,680
5190 FOR J=1 TO 15: READ PAT(J): NEXT J
5200 REM "GET AN INITIAL BOARD FROM THE USER"
5210 PBD$=".............": SBD$="............."
5220 CLS
5230 PRINT "----- GOLF TEE PUZZLE SOLVER -----"
5240 PRINT
5250 PRINT "This program accepts an initial peg configuration"
5260 PRINT "and finds all solutions to the puzzle, displaying"
5270 PRINT "them either on the line printer or on the screen."
5280 PRINT
5290 PRINT "The  program  can start with  any  combination of"
5300 PRINT "tees, and will play these out to all end games."
5310 PRINT
5320 PRINT "Please answer the following questions, to get going:"
5321 PRINT "Print results on the printer (Y/N)";:INPUT T$
5322 IF LEFT$(T$,1)="Y" THEN HRDCPY=TRUE ELSE HRDCPY=FALSE
5323 INPUT "Status report interval"; PSR
5324 PRINT "Interval for printing solutions";:INPUT REP
5325 INPUT "Display winning games on CRT (Y/N)";T$
5326 IF LEFT$(T$,1)="Y"THEN CRT=TRUE ELSE CRT= FALSE
5330 CLS: PRINT "INITIAL BOARD ENTRY:";
5340 GOSUB 4000: REM SHOW AN EMPTY PUZZLE
5345 PRINT @832,"      T=Enter a tee, <SPACE>=No tee, all else means go on."
5350 FOR PEG =1 TO 15
5360 REM BLINK THE CURRENT PEG
5370 PRINT @PAT(PEG),CHR$(191);
5380 FOR J1=1 TO 20: NEXT J1
5390 PRINT @PAT(PEG),CHR$(128);
5400 FOR J1=1 TO 20: NEXT J1
5410 K$=INKEY$
5420  IF K$="" THEN GOTO 5370
```

```
e := (deltay/deltax) - 0.5;

for i := 1 to deltax do begin

     plot(x,y);

     if e > 0  then begin

          y := y + 1;

          e := e - 1;

          end;

     x := x + 1;

     e := e ± (deltay/deltax);

     end;
```

Figure 9.

For the purpose of the PRINT @ function, character positions are numbered left to right, top to bottom between 0 and 1023 with 64 characters per line. Another capability is provided by the SET command. SET considers the CRT to be broken up into a grid of 128 by 48 pixels. The (0,0) pixel point is the upper left hand corner. Line 4050 uses SET to draw interpolated straight lines.

Conclusions

Should you have the sheer tenacity to key the entire program into your own computer, you will discover something immediately. That is, the program takes a long time to analyze an entire game. On the machine I am using, it generates a legal move about every 4 or 5 seconds. A useful exercise for the bold reader would be to recode the logic in assembly language and also to develop some effective tree pruning strategies as discussed above.

I have run a variant of the program on the PDP-11/70 computer using Fortran. The Fortran version included additions to the Legal module to recognize certain classes of board symmetry, and to prune the examined subtrees accordingly. Under these conditions, about two hours were needed to find all solutions to a 14-peg configuration. Incredibly, when you start with a corner hole empty, there are 13,426 unique solutions after pruning for symmetry! Starting with the center hole empty yielded the fewest solutions; there were only 775. To get some feel for the magnitude of the problem, there are 239,831 ways to lose when you start with a corner empty.

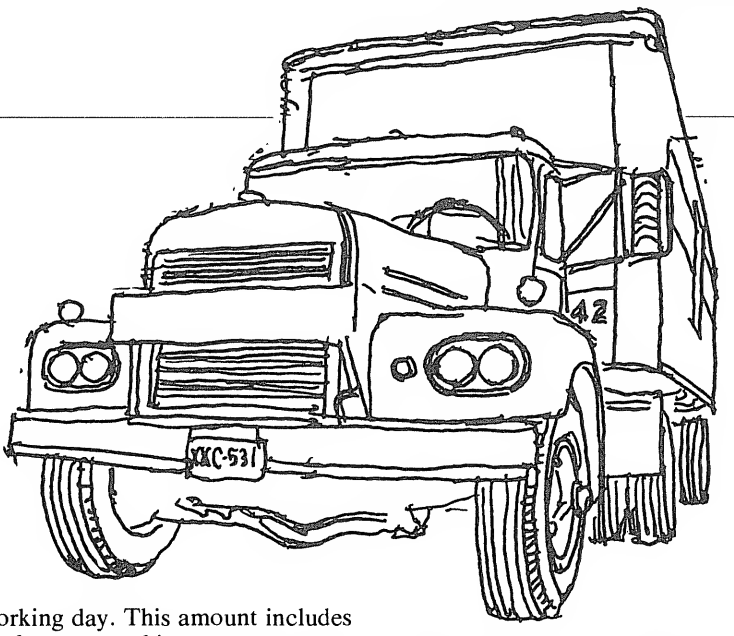
This program has served as an introduction to the world of game playing, modeling, and simulation. There are many more games waiting to be modeled and played; all you need to create them is your trusty computer, your imagination, and Basic. □

```
5430 IF K$<>"T" THEN GOTO 5460
5440 SBD$=LEFT$(SBD$,PEG-1)+"T"+RIGHT$(SBD$,15-PEG)
5450 PRINT @PAT(PEG),CHR$(171);: GOTO 5480
5460 IF K$<> " " THEN GOTO 5490
5470 SBD$=LEFT$(SBD$,PEG-1)+"."+RIGHT$(SBD$,15-PEG)
5475 PRINT @PAT(PEG),CHR$(160);
5480 NEXT PEG
5490 REM ....  DONE WITH INITIAL BOARD
6000 REM  START OF MAINLINE CODE
6010 CLS: PRINT "OK, now let's get down to business."
6015 PRINT: PRINT "There are two modes of play:"
6016 PRINT "  1)I will find the ways to win the board you"
6017 PRINT "    just entered, or"
6018 PRINT "  2)I will moderate your attempt to win the game."
6019 PRINT
6020 INPUT "Select 1 or 2, please";J
6030 IF J=2 THEN GOTO 9000
6040 PRINT " ":PRINT "  ...THEN LET THE GAME BEGIN!"
6050 PBD$=SBD$
6060 GOSUB 6500   : REM  DO ALL THE STUFF!!
6070 PRINT "----- PLAY HAS ENDED -----"
6080 PRINT MVS;" MOVES MADE DURING PLAY."
6090 PRINT "COMPLETED GAMES AT EACH LEVEL ARE:"
6100 FOR J=1 TO 14
6110 PRINT "LEVEL:";J;EDGMS(J);"   ";
6115 IF J=3 OR J=6 OR J=9 OR J=12 THEN PRINT " "
6120 NEXT J: PRINT " "
6130 PRINT SOLS;"TOTAL SOLUTIONS TO PUZZLE"
6140 END
6500 REM "PLY(PBD$)"
6510 REM  PLY RECURSIVELY FINDS ALL BOARD POSITIONS STARTING
6520 REM  FROM PBD$ (INCLUDING WINS).  IF THERE ARE NO
6530 REM  LEGAL MOVES FROM PBD$, PLY RETURNS.  OTHERWISE,
6540 REM  FOR EACH LEGAL MOVE IT CALLS PLY WITH THE BOARD
6550 REM  DERIVED BY MAKING THAT MOVE TO PBD$.
6560 REM
6570 REM     FIRST, SAVE VALUES OF AUTOMATIC VARIABLES
6580 CBD$(STKPTR) = CBD$: CBD$= PBD$
6590 EDGMS(STKPTR)= EDGMS: EDGMS= EDGMS(STKPTR+1)
6600 MOVE(STKPTR)= MOVE
6610 ILLEGALS(STKPTR)= ILLEGALS: ILLEGALS=0
6620 STKPTR=STKPTR+1
6630 FOR MOVE= 1 TO 36
6640 PBD$=CBD$: GOSUB 2000 : REM CHECK BOARD FOR LEGALITY
6650 IF PBD=1 THEN GOTO 6700
6660 REM  ... THIS MOVE IS ILLEGAL
6670 ILLEGALS= ILLEGALS+1
6680 GOTO 6780
6690 REM  ... THIS MOVE IS OK
6700 GOSUB 3000 :REM APPLY MOVE TO PBD$
6710 GOSUB 2800 :REM DO ANY REQUIRED BOOKKEEPING
6720 GOSUB 1000 :REM HOW MANY PEGS LEFT?
6730 IF PBD <> 1 THEN GOTO 6760
6740 REM  ... THIS MOVE IS A WIN
6750 GOSUB 1500 :REM DO ANY GOOD STUFF WHEN YOU WIN
6760 REM  NOW CALL PLY ON BOARD DERIVED FROM MOVE
6770 GOSUB 6500
6780 NEXT MOVE
6790 REM IF NO MOVES WERE LEGAL, NOTE THIS AS AN ENDGAME
6800 IF ILLEGALS=36 THEN EDGMS=EDGMS+1
6810 REM RETURN AFTER RESTORING AUTOMATIC VARIABLES
6820 EDGMS(STKPTR)= EDGMS
6825 STKPTR= STKPTR-1
6830 ILLEGALS = ILLEGALS(STKPTR)
6840 MOVE = MOVE(STKPTR)
6860 CBD$=CBD$(STKPTR)
6870 EDGMS=EDGMS(STKPTR)
6880 RETURN
9000 REM  LOGIC TO MODERATE A PERSON DRIVEN GAME.
9010 CLS: PBD$=SBD$
9020 GOSUB 4000
9030 PRINT @0,"--GOLF TEE PUZZLE--";
9040 GOSUB 1000:PRINT @64,PBD;" STARTING PEGS.";
9050 REM  GET ALL LEGAL MOVES
9060 PRINT @768,"POSSIBLE MOVES:";: X=0
9065 FOR J=783 TO 895: PRINT @J," ";: NEXT J:PRINT @783," ";
9070 FOR MOVE=1 TO 36
9080 GOSUB 2000
9090 IF PBD = 0 THEN GOTO 9110
9100 PRINT "(";MOVE;") ";START(MOVE);"->";FIN(MOVE);", ";:X=X+1
9110 NEXT MOVE
9115 IF X=0 THEN GOTO 9200
9120 PRINT @896,"SELECT A MOVE";: INPUT MOVE
9122 GOSUB 2000 : REM CHECK IF USER GAVE LEGAL MOVE
```

```
9123 IF PBD=1 THEN GOTO 9130
9124 PRINT @896,"** PLEASE SELECT FROM LIST ABOVE **"
9125 FOR J=1 TO 500: NEXT J
9126 PRINT @896,"
9127 GOTO 9120
9130 GOSUB 3000:    REM  APPLY IT TO BOARD
9140 GOSUB 4300:  REM FLASH IT ON THE SCREEN
9150 GOSUB 1000:  REM HOW MANY PEGS ARE LEFT?
9155 PRINT @40,PBD;"PEGS LEFT.";
9170 GOTO 9060
9200 PRINT @768,"NO MORE POSSIBLE MOVES"
9210 GOSUB 1000 : REM COUNT THE REMAINING PEGS
9220 IF PBD=1 PRINT "CONGRATULATIONS! YOU WON" ELSE PRINT "SORRY, YOU LOST."
```

Trucker

Richard R. Galbraith



Trucker is a program which simulates the problems facing a long-haul truck driver. Ideally, you can make a good living hauling freight coast-to-coast without exceeding the legal load limit. If all goes well, you can obey the speed limits and stop each night for eight hours sleep and still make the time schedule. On a good trip you will be able to earn well over $1,000. However, even the the best drivers run into occasional streaks of bad luck and may barely break even.

Bad weather, road construction or a flat tire can place you behind schedule and eat up your profits. You may try to increase your profits by skimping on sleep, driving fast, or carrying an overweight load. However, pushing too hard raises the risk of a traffic accident, and you will be fined if you are caught breaking the law.

Your Truck

You are driving an 18-wheel tractor-trailer combination that can hold 50,000 pounds of cargo (10,000 pounds more than the legal limit). You are buying your truck through a bank loan that requires payment of $1,955 per month, or $85 for each working day. This amount includes reserves for taxes and insurance.

Your truck has a 200-gallon fuel tank and gets 4.5 miles per gallon of diesel fuel. Your mileage decreases when you drive faster or slower than 55 miles per hour. Your fuel gauge is accurate to within 5 gallons and your speedometer is accurate to within 3 miles per hour.

Accidents

It is extremely unlikely that you will be involved in a traffic accident in good weather if you drive at a reasonable speed and get enough rest. The danger increases dramatically if you drive at an excessive rate of speed, fail to slow down in fog or a blizzard, or continue driving after you have become fatigued. An exhausted driver speeding through a snow storm is asking for trouble.

There is always the danger of losing time due to a flat tire. This danger can be reduced by purchasing retreads or more expensive tires before you start your trip, and by promptly replacing your spare tire after a flat.

Speeding

The speed limit is 55 miles per hour unless otherwise posted. Generally,

Richard R. Galbraith, 2124 E. Fremont Dr., Temple, AZ 85282.

Smokey will allow some leeway before pulling you over, but the faster you go the more likely you are to attract his attention. There are also a couple of places along the way where a radar speed trap may be in operation with strict enforcement.

Whenever you get a traffic ticket, you will lose time as you wait to pay your fine at the Justice of the Peace. If you receive more than three traffic tickets, you lose your Interstate Commerce Commission driver's license.

Truck Stops

Every three or four hours you will approach a truck stop. Each stop will take at least one hour while you get coffee, fuel and a spare tire if necessary. The price of diesel fuel and tires will vary unpredictably, but diesel fuel will average about $1.00 per gallon.

Truck stops are also the only places where you can sleep. You may choose when to sleep, but, if you attempt to sleep during the day, you will be disturbed by traffic noise.

Cargo

You can select one of three types of cargo to haul for each trip:

1. U.S. Mail: This contract will pay $.0475 per pound, or $1,900 for a 40,000 pound load upon delivery.
2. Freight Forwarding: This contract pays $.05 per pound, or $2,000 for a load. However, there is a 10% penalty that is subtracted if you are more than 12 hours late in delivering your freight.
3. Oranges: This contract will pay $.065 per pound of good oranges delivered to New York, which amounts to $2,600 for a standard load. You are required to run the air-conditioning unit in your trailer in order to keep the oranges from rotting or freezing. You will burn 7 gallons of diesel fuel per hour while you sleep.

Routes

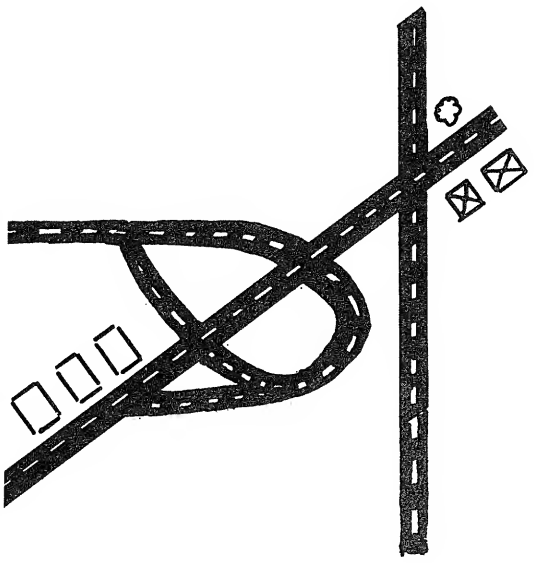
You can choose one of three routes; the northern route, the middle route or the southern route. Let's look at each route in detail:

Northern Route

This route is the shortest but also the riskiest. You will leave from Los Angeles on Interstate 15 and drive through Las Vegas and Denver. You then take Interstate 80 through Nebraska, northern Ohio and Pennsylvania. The total mileage is 2,710. You will pay a total of $195 in tolls and have one chance in eight of avoiding weighing stations. The danger of bad weather is high, and the speed limit is vigorously enforced.

Middle Route

The middle route follows old Route 66 from Los Angeles through northern Arizona and Oklahoma into St. Louis. Then you cut over to the Pennsylvania Turnpike and follow through to New York. The total distance to New York is 2,850 miles. The toll road portions will cost you $240 in fees. This route has fewer Smokies watching your speed and the weather conditions are much more favorable than the Northern route. However, watch the weight in your trailer since there are usually several truck scales in operation.



Southern Route

This route takes you from Los Angeles on Interstate 10 through Arizona, New Mexico and Texas. You then follow Interstate 20 to Atlanta before heading north to Washington, D.C. The last leg of your journey follows Interstate 95 up the Atlantic coast. The mileage is 3120, much longer than the other routes. However, it is the safest route because you avoid much of the bad weather. Tolls amount to only $95 and you will run into fewer police and fewer truck scales. If you cannot resist the temptation to take on an over-weight cargo or if you have a lead foot, this is the best route for you to take. □

TRUCKER/16K

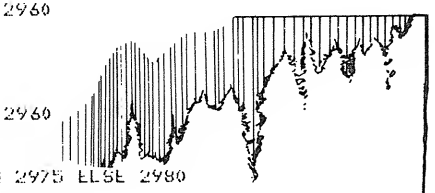
```
10      CLS
       :PRINT@220,"TRUCKER"
20      PRINT@336,"by Richard Galbraith, Tempe, AZ"
30      PRINT@404,"Copyright  October  1980"
40      PRINT@534,"All rights reserved"
50      GOSUB5650
       :GOSUB5650
       :RANDOM
70      DEFINTC-S
       :DIMMT(2),MP(2,25),MP$(2,25),MR$(2,25),ZM(2,25),D$(6),NT$(4)
80      DD$="$$$,###"
       :DC$="$$$,###.##"
90      NT$(1)="First"
       :NT$(2)="Second"
       :NT$(3)="Third"
       :NT$(4)="Fourth"
92      DS$(0)="Monday"
       :DS$(1)="Tuesday"
       :DS$(2)="Wednesday"
94      DS$(3)="Thursday"
       :DS$(4)="Friday"
       :DS$(5)="Saturday"
       :DS$(6)="Sunday"
1000    CLS
       :XC=190
       :MF=0
       :HL=3
       :HS=7
       :HR=0
       :GOSUB2100
1010    PRINT@128," "
1020    PRINT"You are at the Los Angeles Trucking Terminal"
1030    PRINT"Three types of cargo are available
       :"
1040    PRINTTAB(5)"1--ORANGES  (highest profit IF they don't spoil)"
1050    PRINTTAB(5)"2--FREIGHT FORWARDING  (penalty for late deliver
        y)"
1060    PRINTTAB(5)"3--U.S. MAIL  (lowest rate, but no hurry to arriv
        e)"
```

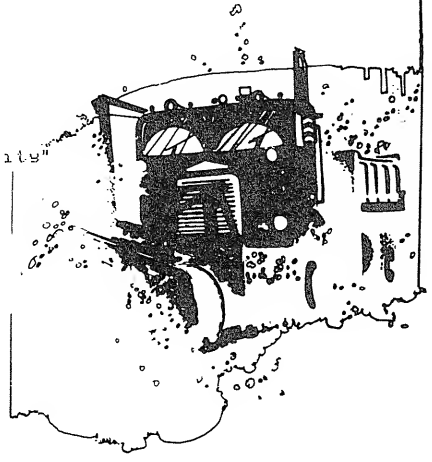
```
1070    PRINT"The cargo is due in New York BY 4 PM on Th.sday."
        :INPUT"Which type of cargo do you want";CT
1080    IFCT<1ORCT>3INPUT"PICK A NUMBER
        : 1, 2, OR 3";CT
        :GOTO1080
1090    INPUT"How many pounds will you carry (40000 is the LEGAL limi
        t)";WL
1100    IFWL<25000 PRINT"You can't make a living on half a load."
        :GOTO1090
1110    PRINT
        :PRINTTAB(5)"They are loading your truck now."
1120    RESTORE
1150    FORRT=0TO2
        :READNP,MT(RT)
        :FORI=1TONP

1170    READMP(RT,I),MP$(RT,I),MR$(RT,I),ZM(RT,I)
        :NEXTI,RT
1190    TC=10
        :WF=190
        :NP=1
        :TS=1
        :SL=55
        :XN=XN+1
1200    IFWL>50000THENWL=50000
        :PRINT"50,000 pounds of cargo has filled your trailer!"
        :GOSUB5650
1220    HR=HR+1
        :CLS
        :GOSUB2100
        :PRINT@128,""
1225    PRINT"You paid $ 190 for a nearly full tank of Diesel."
        :PRINT
1230    INPUT"Two of your tires are worn.  Do you want replacements";
        Z$
1240    IFLEFT$(Z$,1)="N"ORLEFT$(Z$,1)="n"THEN1350
1250    PRINT"A NEW tire costs $200.   A RETREAD costs $100."
        :PRINTTAB(5);
1260    INPUT"Which type do you want";Z$
        :PRINTTAB(5);
        :Z$=LEFT$(Z$,1)
1270    INPUT"How many";T
1280    IFT=3IFZ$="N"ORZ$="n"THENTS=2
        :T=2
        :XC=XC+200
1290    IFT<0ORT>2THEN1330
1300    IFT=0THEN1350
1310    IFZ$="R"ORZ$="r"THENTC=TC-3*T
        :XC=XC+100*T
        :GOTO1350
1320    IFZ$="N"ORZ$="n"THENTC=TC-4*T
        :XC=XC+200*T
        :GOTO1350
1330    PRINT"I did not understand your answers."
        :PRINT"Let's try again
        :"
        :PRINTTAB(5);
        :GOTO1230
1350    PRINT
        :PRINT"You may choose the Northern, Middle or Southern route."

1360    INPUT"   Which route do you choose";Z$
        :Z$=LEFT$(Z$,1)
1365    IFZ$="N"ORZ$="n"THENRT=1
        :RH=4
        :GOTO 1600

1370    IFZ$="M"ORZ$="m"THENRT=0
        :RH=2
        :GOTO 1600
1375    IFZ$="S"ORZ$="s"THENRT=2
        :RH=1
        :GOTO 1600

1380    PRINT"Please, answer
        : NORTH, MIDDLE, or SOUTH !"
1385    GOTO1360
1400    AF=SP↑2*CD*CR
1420    IFAF>RND(0)*1E7GOTO4000
1430    AF= SQR(MF+100)*TC
1440    IFAF>RH*25000*RND(0)GOSUB2600
1450    IFSP>SL-RH+10GOSUB2300
1460    HR= HR+1
        :HL=HL+1
1470    IFSL<40THENSL=55
1480    T=ABS(55-SP)
        :IFT>12THENT=12.5
1490    T1= SP/(4.5 -0.2*T)
1500    WF=WF-T1
        :IFWF<0GOSUB2500
1510    MF= MF+SP
1520    IFMF>MT(RT)THEN5000
1530    GOSUB5650
1550    CLS
        :GOSUB 2100
1560    PRINT@64,"Approximate FUEL
        :";INT(WF-5)+RND(10);TAB(36)"SPEED
        :";SP
1570    PRINTTAB(8)"Odometer
        :";MF;TAB(30)"Miles to go
        :";MT(RT)-MF
1580    PRINT
1600    IFMP(RT,NP)<=MFGOTO3100ELSEPRINT"Cruising on ";MR$(RT,NP)
1610    GOSUB 3000
        :PRINT"You are feeling ";CD$
1620    GOSUB 2800
        :PRINT"Current weather
        : ";CR$
1630    NS=NS+1
        :IFNS>3GOSUB1700
1640    INPUT"How fast do you wish to go";SP
1650    IFSP<20PRINT"YOU HAVE TO GO AT LEAST 20 -- ";
        :GOTO1640
1660    IFSP>INT(1.5*SL)THENSP=INT(1.5*SL)
        :PRINT"You can only set the old rig to go";SP;"mph on this roa
        d."
1670    GOTO 1400
1700 REM
1710    INPUT"TRUCK STOP AHEAD.   Do you want to stop";Z$
1720    IFLEFT$(Z$,1)="N"ORLEFT$(Z$,1)="n"THENNS=1
        :HL=HL+1
        :RETURN
1730    IFLEFT$(Z$,1)<>"Y"ANDLEFT$(Z$,1)<>"y"INPUT"MAKE UP YOUR MIND.
            YES or NO";Z$
        :GOTO1720
1740    T= 85 +INT(35*RND(0))
1750    PRINT"Diesel fuel costs";T;"cents a gallon."
1760    INPUT"    How many gallons do you want";T1
1770    IF T1>0 PRINT"PAY";
        :PRINTUSING"$$$$$.##";T*T1/100
        :XC=XC+T*T1/100
        :WF=WF+T1
```

```
1780    PRINT"So far, you have spent ";
       :PRINTUSING DC$;XC
1790    IF WF> 201 PRINT"Your tank only holds 200 gallons --";INT(WF-
        200);"gallons spilled !!"
       :WF=200
1800    IFTS>0THEN1900
1810    T=200+INT(50*RND(0))
       :T1=100 +INT(70*RND(0))
1820    PRINT"A NEW tire costs $";T;"     a RETREAD costs $";T1
1830    INPUT"     Do you want to buy a tire";Z$
1840    IFLEFT$(Z$,1)="N"ORLEFT$(Z$,1)="n"THEN1900
1850    INPUT"     Choose
       :  New or Retread";Z$
1860    IFLEFT$(Z$,1)="N"ORLEFT$(Z$,1)="n"THENXC=XC+T
       :TS=2
       :GOTO1900
1870    IFLEFT$(Z$,1)="R"ORLEFT$(Z$,1)="r"THENXC=XC+T1
       :TS=1
       :GOTO1900
1880    PRINT"I DID NOT UNDERSTAND YOUR ANSWERS."
       :GOTO1830
1900    HR= HR+1
       :NS=0
1910     INPUT"Do you want to set some sleep";Z$
1920    IFLEFT$(Z$,1)="N"ORLEFT$(Z$,1)="n"GOSUB2100
       :RETURN
1930    INPUT"     How many hours of rest";T
1940    IFT<1RETURN
1950    DH= HR -24*INT(HR/24)
1960    HR=HR+T
        :GOSUB5650
       :GOSUB5650
       :IFCT=1THENWF=WF-7*T
       :IFWF<0THENWF=0
       :GOSUB2570
1970    IF DH>21 OR DH<12 THEN T=INT(T/2+.6)
       :PRINT"Thanks to the daytime noise, you got only";T;"hours rea
        l sleep."
1980    HS=HS+T
1990    IFT>3THENHL=0ELSEHL=HL/2
2000    GOSUB5600
2010    GOSUB2100
       :PRINT"Time to hit the road again."
2015    IFCT=1PRINT"You now have ";
        :PRINTUSING"###";WF;
        :PRINT" gallons of fuel"
       :INPUT"Do you want to buy more";Z$
       :IFLEFT$(Z$,1)="Y"ORLEFT$(Z$,1)="y"THEN1740
2020    RETURN
2100    DH=HR+8
2120    DT=INT(DH/24)
        :DH=DH -24*DT
2130    IFDT>6THENDT=DT-7
       :GOTO2130
2140    DM$="am"
2150    IF DH=12 THEN DM$="noon"
       :GOTO 2200
```

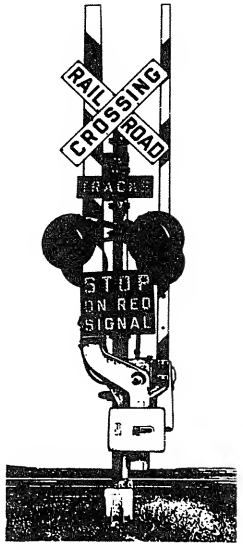
```
2160    IF DH>12 THEN DH=DH-12
       :DM$="pm"
2170    IF DH=0 THEN DH=12
       :DM$="midnight"
2200    T=PEEK(16416)
       :T1=PEEK(16417)
2210    PRINT@13,"Day
       : ";DS$(DT);TAB(37)"Time
       :";DH;DM$;"       ";
2220    POKE16416,T
       :POKE16417,T1
2230    RETURN
2300 REM
2310    IF(SP-SL+2*RH-5)*2<900*RND(0) RETURN
2320    PRINT"SMOKEY is behind you with his lights on.   PULL OVER!"
2340    GOSUB5650
2350    NT=NT+1
       :PRINT"See the JUSTICE of the PEACE for your ";NT$(NT);" offen
        se"
2360    PRINT"    Wait";NT;"hours for your hearing"
2370    HR=HR+NT
       :HL=HL+NT
2380    IF NT>3 THEN 2430
2390    T=NT*RND(5)
       :T1= 5*(RT+NT*RND(4))
2400    PRINT"    FINE is ";
       :PRINTUSING"$###";T1;
       :PRINT" plus $";T;"for each MPH over the limit."
2410    PRINT"    PAY ";
       :PRINTUSING DD$;T1+T*(SP-SL)
       :XC=XC+T1+T*(SP-SL)
2420    GOSUB5650
       :GOSUB5650
       :RETURN
2430    PRINT"  You are sentenced to 30 Days in Jail for reckless dri
        ving."
2440    GOSUB5650
2450    PRINT"Your I.C.C. Driver's License is revoked !"
2460    GOTO5500
2500    T1=T1+WF
       :WF=0
       :SP=0
2520    T=(4.5 -0.2*T)*T1
       :MF=MF+T
2530    PRINT"After";T;"more miles, you ran out of fuel  (DUMMY !!)"
2540    PRINT"    It cost $ 200 to set a barrel of diesel delivered."

2550    WF=55
       :T1=RND(5)
       :HR=HR+T1
       :XC=XC+200
       :HL=HL+T1
2560    PRINTTAB(5)"    You also wasted";T1;"hours by your carelessne
        ss."
2570    IF CT=1 THEN CX=CX+RND(3)
       :PRINT"    Sitting with the refer unit off is damaging the ora
        nges."
```

```
2580    FOR I=1 TO 500
       :NEXTI
2590    RETURN
2600    GOSUB5600
2620    PRINT"You just blew a tire !!"
2630    IF TS=0 THEN 2710
2640    TC=TC -2*TS
       :TS=0
2650    T=RND(2)
       :IF T=1 THEN T$="outside" ELSE T$="inside"
2660    PRINT"   It took";T;"hours to change the ";T$;" tire."
       :HR=HR+T
       :HL=HL+T+1
2670    GOSUB5650
       :RETURN
2710    PRINT"Since your spare has already been used, you have to cal
        l a tow  truck from town to deliver a new tire to you."
2720    PRINT"    This service cost $ 400 and took 4 hours."
2730    HR=HR+4
       :HL=HL+4
       :XC=XC+400
2740    GOSUB5650
       :RETURN
2800 REM
2810    AF=(3000 + MF)*RND(0)
       :ON (RT+1) GOTO 2870,2820,2910
2820    IF AF<3300 AND CR<>50 THEN 2960
2830    IF AF>4800 THEN 2965
2840    IF AF>4600 THEN 2970
2850    IF AF>3800 THEN 2975
2860    GOTO2985
2870    IF AF<3400 AND CR<>50 THEN 2960
2880    IF AF>4900 THEN 2965
2890    IF AF>4700 THEN 2970
2900    IF AF>4200 IF RND(3)=1 THEN 2975 ELSE 2980
2905    GOTO2985
2910    IF AF<4000 AND CR<>50 THEN 2960
2920    IF AF>5700 THEN 2965
2930    IF AF>5500 THEN 2970
2940    IF AF>4400 THEN 2980
2950    GOTO2985
2960    CR=1
       :CR$="CLEAR & DRY"
       :RETURN
2965    CR=50
       :CR$="B-L-I-Z-Z-A-R-D  !!"
       :RETURN
2970    CR=10
       :CR$="FOG -- Limited visibility"
       :RETURN
2975    CR=5
       :CR$="LIGHT SNOW"
       :RETURN
2980    CR=5
       :CR$="RAIN"
       :RETURN
```





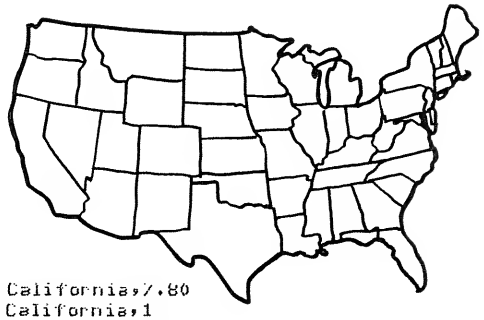

```
2985    CR=3
       :CR$="CLEAR, but roadway is wet"
       :RETURN
3000 REM**
3010    IF HL>19 OR HR/HS>4 THEN CD=100
       :CD$="..E.X.H.A.U.S.T.E.D.."
       :RETURN
3020    IF HL<4 AND CSNG(HR/HS)<2.3 THEN CD=1
       :CD$="RESTED & READING TO GO."
       :RETURN
3030    IF HL<8 AND CSNG(HR/HS)<2.5 THEN CD=2
       :CD$="FINE"
       :RETURN
3040    IF HL<12 AND HR/HS<=3 THEN CD=4
       :CD$="  B O R E D"
       :RETURN
3050    IF HL<16 AND HR/HS<=3 THEN CD=8
       :CD$="  T I R E D  !!"
       :RETURN
3060    CD=25
       :CD$="FATIGUED...You're getting sleepy"
       :RETURN
3100 REM
3110    PRINT"You have just passed ";MP$(RT,NP)
3120    ZH=ZM(RT,NP)
       :SL=55
3130    ON INT(ZH) GOSUB 3210,3310,3360,3410,3500,3710,3860
3140    NP=NP+1
       :IF INT(ZH)=8 THEN 5000 ELSE 1600
3210    PRINT"Time Zone changes -- Set clock ahead one hour"
3220    HR=HR+1
       :GOSUB 2100
3230    RETURN
3310    T=100*(ZH-INT(ZH))
3320    PRINT"STOP!   PAY TOLL of ";
       :PRINTUSING"$$$.$$";T
3330    XC=XC+T
3340    RETURN
3360    IF RND(0)< ZH -INT(ZH) RETURN
3370    PRINT"CONSTRUCTION AHEAD !!"
       :FOR I=1 TO 500
       :NEXTI
3380    PRINT"SLOW DOWN -- SPEED LIMIT 35 mph"
       :SL=35
3390    RETURN
3410    IF RND(0)< ZH -INT(ZH) RETURN
3420    T=SP +RND(5) -2
3430    PRINT"You were just clocked by RADAR at";T;"mph"
3440    IF T> SL+3 GOSUB 2320 ELSE PRINT"    No ticket this time."
3450    RETURN
3500    IF ZH=INT(ZH) IF RND(0)<.5 THEN 3520 ELSE RETURN
3510    IF RND(0)< ZH -INT(ZH) RETURN
3520    PRINT"WEIGHING STATION OPEN  --  TRUCKS MUST STOP"
       :GOSUB5650
3530    PRINT"Scale weighs truck with cargo, fuel & driver
       : ";
3540    T=19000 +WL +7*WF +25*RND(10)
3550    PRINTUSING"$$,$$$";T;
       :PRINT" POUNDS."
3560    T=INT(T-60000)
3570    IF T<1 PRINT"     You're O.K."
       :RETURN
```



```
 3580    IF ZH=5.00 THEN 3630
3590    T1=RND(4)+2
       :PRINT"    Overweisht fine is $ 200 plus";T1;"cents/pound"
 3600    XC=XC+200+(T*T1)/100
3610    PRINT"Pay fine of";
       :PRINTUSING DC$;200+(T*T1)/100
 3620    RETURN
3630 REM
3640    PRINT"You are not allowed to enter Lousiana with that load."
 3650    PRINT"    Take a 200 mile detour throush Arkansas with 45 mph
          limit."
3660    SL=45
       :MR$(RT,NF)="Arkansas County Roads"
 3670    FOR I=12 TO 25
       :MP(RT,I)=MP(RT,I)+200
       :NEXTI
 3680    MT(RT)=MT(RT)+200
 3690    RETURN
3710    IF RND(0)< ZH -INT(ZH) RETURN
 3720    T=RND(6)
 3730    PRINT"A ROCK_SLIDE has blocked the Alleshany Tunnel entrance"

3740    PRINT"    The highway Department will have it cleared in";T;"
         hours"
3750    HR=HR+T
       :GOSUB5650
       :IF CT=1 THEN WF=WF-7*T
       :IF WF<=1 GOSUB3820
 3760    IF T>1 THEN T1=INT(T/2 +.5) ELSE T1=0
3770    IF T1>3 THEN HL=0 ELSE IF T1>0 HL=HL/2
 3780    HS=HS+T1
3790    PRINT"    While waiting, you sot";T1;"hours of sleep"
 3800    GOSUB 2100
       :RETURN
 3820    PRINT"    You ran out of sas while waiting"
       :T=0
       :GOSUB2540
 3830    RETURN
 3860    IF CT>1 RETURN
 3870    IF RND(0)< ZH - INT(ZH) RETURN
 3880    PRINT"The trailer refriseration unit has failed endanserins t
         he carso"
 3890    PRINT"    Repairs take 2 hours and cost $ 100"
 3900    CX=CX +RND(4)
       :HR=HR+2
       :HL=HL+2
       :XC=XC+100
 3910    GOSUB2100
       :GOSUB5650
 3920    RETURN

5450    PRINT
       :PRINT
       :INPUT"Do you want to make another trip";Z$
5460    IF LEFT$(Z$,1)<>"N" AND LEFT$(Z$,1)<>"n" THEN 1000 ELSE CLS
       :END
 5470    PRINT"BAD TRIP. . . You lost";
       :PRINTUSING DC$;ABS(XT)
5480    IF XP>=0 GOTO 5430
5490    PRINT"    You are BANKRUPT !!!"
 5500    GOSUB5650
       :GOSUB5650
5520    PRINT
       :PRINT"Your ris has been repossessed."
```



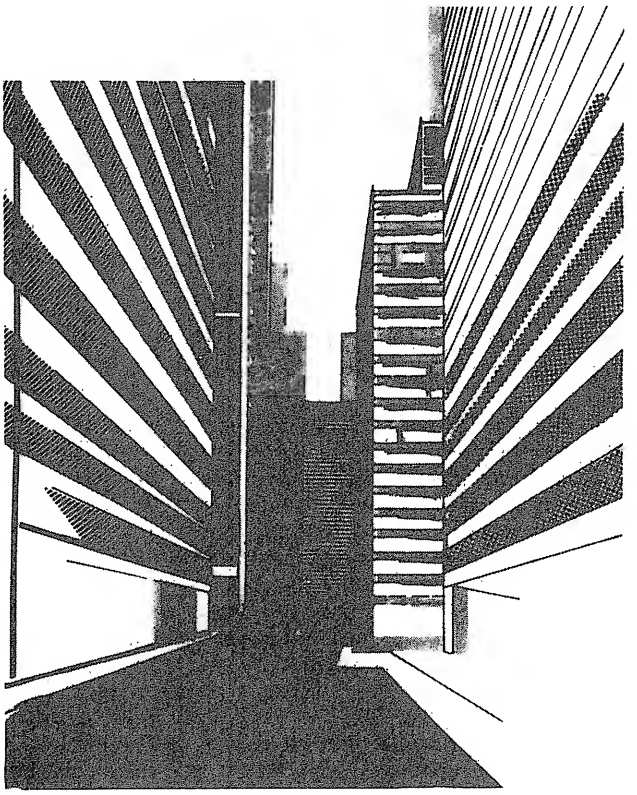
```
5530    PRINT
       :END
 5600    FOR1=1TO12
       :POKE14308,1
       :FORJ=1TO3
       :POKE14308,2
       :FORJ=1TO3
       :NEXTJ,I
       :RETURN
5650    FOR1=1TO800
       :NEXTI
       :RETURN
 9030    DATA 21,2850
 9040    DATA 90,BARSTOW,I-15 in California,7.80
 9050    DATA225,NEEDLES,I-40 in California,1
 9060    DATA 440,FLAGSTAFF,I-40 in Arizona,3.65
 9070    DATA 620,GALLUP,I-40 in Arizona,5.5
 9080    DATA 760,ALBUQUERQUE,I-40 in New Mexico,3.35
 9090    DATA 930,TUCUMCARI,I-40 in New Mexico,1
 9100    DATA 1040,AMARILLO,I-40 in Texas,7.80
9110    DATA 1155,OKLAHOMA Border,I-40 in Texas,5.5
 9120    DATA 1305,OKLAHOMA CITY,I-40 in Oklahoma,2.65
 9130    DATA 1530,MISSOURI Border,Oklahoma Turnpike,2.40
 9140    DATA 1815,ST. LOUIS,I-44 in Missouri,0
 9150    DATA 1980,TERRE HAUTE,I-70 in Illinois,5.5
 9160    DATA 2050,INDIANAPOLIS,I-70 in Indiana,0
 9170    DATA 2115,OHIO Border,I-70 in Indiana,1
 9180    DATA 2220,COLUMBUS,I-70 in Ohio,5.5
 9190    DATA 2350,WHEELING West Virginia,I-70 in Ohio,4.25
 9200    DATA 2410,NEW STANTON,I-70 in Pennsylvania,6.75
 9210    DATA 2570,HARRISBURG,Pennsylvania Turnpike,3.75
 9220    DATA 2760,NEW JERSEY Border,Pennsylvania Turnpike,2.95
 9230    DATA 2840,HOLLAND TUNNEL,I-70 in New Jersey,2.40
 9240    DATA 9999,NEW YORK,New York Streets,0
 9255    DATA 18,2710
 9260    DATA 90,BARSTOW,I-15 in California,7.80
 9270    DATA 245,LAS VEGAS,I-15 in California,1
 9280    DATA 365,UTAH Border,I-15 in Arizona,0
 9290    DATA 500,end of Interstate,I-15 in Utah,3.20
 9300    DATA 555,SALINA,US-89 in Utah,4.50
 9310    DATA 760,GRAND JUNCTION,I-70 in Utah,5.40
 9320    DATA 1010,DENVER,I-70 in Colorado,3.75
 9330    DATA 1190,NEBRASKA Border,I-76 in Colorado,1

 9340    DATA 1450,OMAHA,I-80 in Nebraska,5.50
 9350    DATA 1590,DEMOINES,I-80 in Iowa,4.75
 9360    DATA 1750,ILLINOIS Border,I-80 in Iowa,5.6
 9370    DATA 1910,GARY,I-80 in Illinois,2.50
 9380    DATA 2050,OHIO Border,Indiana Turnpike,2.45
 9390    DATA 2215,CLEVELAND,Ohio Turnpike,2.80
 9400    DATA 2280,PENSYLVANIA Border,I-80 in Ohio,4.16
 9410    DATA 2615,EAST STROUDSBERG,I-80 in Pennsylvania,3.33
 9420    DATA 2675,WASHINGTON BRIDGE,I-80 in New Jersey,2.20
 9430    DATA 9999,NEW YORK,City Streets,0
 9450    DATA 25,3120
 9460    DATA 75,PALM SPRINGS,I-10 in California,0
 9470    DATA 225,BLYTHE,I-10 in California,1
9480    DATA 375,PHOENIX,I-10 in Arizona,0
 9490    DATA 495,TUCSON,I-10 in Arizona,7.9
 9500    DATA 650,LORDSBURG,I-10 in Arizona,5.75
 9510    DATA 795,EL PASO,I-10 in New Mexico,0
 9520    DATA 965,PECOS,I-10 in Texas,1
 9530    DATA 1080,ODESSA,I-20 in Texas,0
 9540    DATA 1250,ABILENE,I-20 in Texas,3.80
 9550    DATA 1439,DALLAS,I-20 in Texas,0
```



```
9560     DATA 1610,LOUISIANA Border,I-20 in Texas,5.00
 9570    DATA 1785,VICKSBURG,I-20 in Louisiana,0
 9580    DATA 1965,ALABAMA Border,I-20 in Mississippi,1
 9590    DATA 2100,BIRMINGHAM,I-20 in Alabama,4.25
 9600    DATA 2200,GEORGIA Border,I-20 in Alabama,0
 9610    DATA 2255,ATLANTA,I-20 in Georgia,0
 9620    DATA 2320,CAROLINA Border,I-85 in Georgia,5.75
 9630    DATA 2565,GREENSBORO,I-85 in North Carolina,3.80
 9640    DATA 2680,VIRGINIA Border,I-85 in North Carolina,7.85
 9650    DATA 2775,RICHMOND,I-85 in Virginia,0
 9660    DATA 2880,WASHINGTON D.C.,I-95 in Virginia,0
 9670    DATA 2920,BALTIMORE,I-95 in Maryland,2.30
 9680    DATA 2990,NEW JERSEY Border,I-95 in Delaware,2.25
 9690    DATA 3110,HOLLAND TUNNEL,New Jersey Turnpike,2.40
 9700    DATA 9999,NEW YORK,City Streets,0
```

Streets of the City

Kenneth R. Murray



ABOUT THE GAME

Congratulations! You have been named Transportation Director of River City, Michigan. River City is a central city with a declining population which is now at 185,000 persons. Budget problems over the past decade have resulted in a severely deteriorated road system and inadequate bus service.

Prior to your being hired, the City Commission approved a ten-year transportation improvement plan that will now be your responsibility to complete. In the Street Fund, the plan calls for reconstructing 44 miles of main streets, called primaries, and 16 miles of interstate. At the same time, you have to improve significantly the overall street conditions and traffic safety. For the Transit Authority, an aging bus fleet needs to be expanded and modernized, and ridership must be expanded.

Your success will be measured in two ways. The first is how well you progress each year in meeting the overall goal. Second is your ability to maintain a majority vote of the City Commission. Each influences the other.

Kenneth R. Murray, Deputy City Manager, Grand Rapids, MI.

Goals to be Achieved

In the initialization of the simulation, the initial conditions are randomly set within reasonable limits. This includes the first budgets, street mileage and conditions, the traffic safety index, fleet size and age, and transit performance. The goals that you must achieve are as follows:

Highway Construction: The costs are initially set at random. Each year costs will increase because of inflation. An inadequate maintenance program will also cause the construction costs to rise.

Street Conditions: A street condition index is randomly set; the higher the index, the worse the condition. Each year the index

is adjusted according to street mileage (total streets will be added in relation to inflationary pressures on development) and how well you budget for street maintenance. Your maintenance costs are determined by street mileage, street conditions, labor negotiations, and inflation.

Traffic Safety: A traffic safety index is also set randomly; again, the higher the index, the worse the traffic accident rate. This index is adjusted each year according to changes in the street conditions and how well you meet your maintenance and safety budget. The safety needs are determined by street mileage, the traffic safety index, labor negotiations, and inflation.

Age of Bus Fleet: The size and age of the fleet are randomly set and are incremented each year according to your sale and acquisition of buses. Sale is assumed on the basis of the oldest buses being sold first. Sale and purchase prices are influenced by inflation.

Ridership: Ridership is initially determined randomly. It is then affected by decisions on the number of routes, the hours of service, the days of service, and bus fare. The performance measures of downtime and on-schedule performance (referred to as service delay) and strikes will also affect ridership.

Fleet Downtime: This is measured by an index; the higher the index, the greater the downtime. The index is adjusted according to the age of the fleet and how well you meet your maintenance budget. The maintenance needs are determined by the size and age of the fleet, the level of service, labor negotiations, and inflation.

Service Delay: The higher the service delay index, the poorer is your on-schedule performance. This index is determined by the size of the fleet relative to the number of routes, downtime, and meeting your operational budget. Operating needs are affected by the number of routes, hours and days of operation, labor negotiations, and inflation. You should not let the average number of buses per route drop below three.

Transit Authority Service Decisions

In this phase you determine the level of transit service you will have for the year. Your decisions and ranges are as follows:

SERVICE	INITIAL VALUE	RANGE OF OPTIONS
Routes	6	6 to 25
Hours of Operation Per Day	12	12, 17, or 24
Days of Operation	6	6 or 7
Fare	$.35	$.25 to $1.00

GOAL	STANDARD
Primary Street Reconstruction	Reconstruct 44 Miles
Interstate Highway Construction	Build 16 Miles
Street Condition Index	Reduce 60 Percent
Traffic Safety Index	Reduce 60 Percent
Bus Fleet Age	Reduce 60 Percent
Bus Ridership	Increase 4 Times
Fleet Downtime Index	Reduce 60 Percent
On Schedule Performance Index	Reduce 60 Percent

capital account and vice versa. The percentage that you can shift will change according to the amount of bonds you have issued. Your operating revenue, which includes funds left over from the previous year, gasoline taxes, and tax levy, is automatically adjusted to delete bond payments. Gasoline tax revenue is initially calculated at the start of the simulation based on street mileage and vehicle miles, then adjusted according to mileage changes and inflation. It is not a variable over which you have control. The construction budget, exclusive of bonds, is similarly set.

In making your maintenance and safety decisions, you should remember that the needs shown are the minimum amounts necessary to keep the maintenance and safety indexes approximately the same.

Bonding

In years 3 and 7, you will have the option of seeking authority to borrow money (in the form of bonds) for street construction. In year 3, the bond limit is $1.5 million, and in year 7, it is $2.0 million, each per year. You do not have to request the entire amount. The City Commission will decide what size of a bond issue to put to a vote of Reducing the indexes requires more than the minimum appropriation.

Property Taxes

In this phase you will ask the City Commission to levy up to ten mills of property tax for street and transit operation. The amount that is approved will depend upon your support of the Commission and the size of the levy requested. The tax that is approved must then be divided between streets and transit. If you are too greedy, the chances that the Commission will approve a less-than-adequate property tax increase.

The amount of the property tax base is set at the start of the simulation. Each year it changes according to inflation, street improvements, and bus ridership. The theory is that with streets and more bus riders, property values will increase. Conversely, with poorer streets and fewer riders, property values will decrease.

Street Fund Budget

Once the tax levy is determined, you must decide how much to spend from the Street Fund on maintenance, safety, and construction. You will be able to transfer money from the operating account to the the citizens. The Commission decision will depend upon the size of the bond requested and your support among the Commission members. Once the issue is submitted to a vote, you will be asked to make certain pledges to the Coalition of Neighborhood Associations. Making the pledges will improve the chance of passage; however, if you fail to keep your pledges, you will be penalized severely.

Transit Budget

You have a similar set of decisions to make on the Transit Authority budget. Operating revenues include rider fare (ridership times fare), a federal subsidy which is automatically set at half of the operating and maintenance needs for the year and tax revenues. The capital budget consists of revenues from the sale of buses and from occasional federal grants. You may transfer up to 25% of the operating revenues to acquisition, but you may not use the capital fund for operations. By random determination, you may receive a federal grant for bus acquisition. In those years you cannot transfer funds from the operating account. Your decision whether to buy and/or sell buses depends upon your fleet needs. Remember that buses add to maintenance costs, whether you need them or not. A rule of thumb is that three buses are needed per route. Again, the operating and maintenance needs are minimums necessary to hold the indexes about the same.

Labor Negotiations

The final phase of decision making is labor negotiations for the next year. The outcome of the negotiations directly affects

your operating and maintenance budget for streets and the Transit Authority.

There will be between two and six rounds of negotiations, with the Union making the first offer. Subsequent union offers will depend upon how willing you are to bargain in good faith. If you reach a settlement, excellent. If you do not reach a settlement, you risk a strike. The probability of a strike depends upon the beginning and ending positions of the two parties. Once a strike occurs, the wage decision is out of your hands; and it will be decided by an arbitrator according to the beginning and ending positions of the two parties and how much each has changed its position. A strike negatively affects your performance for the year in which it occurs, so you should not risk one lightly.

Performance Review

Once you have completed the decision process, you will be given a comparison of the effects of your decisions this year against the past year and against the fiscal plan. You will also be shown a graphic display of the status of your street construction. Your general performance will be evaluated and you will be told the strengths and weaknesses of your performance. Depending upon your performance, you can gain or lose support among the Commissioners. You begin the game with the unanimous support of all eleven Commissioners.

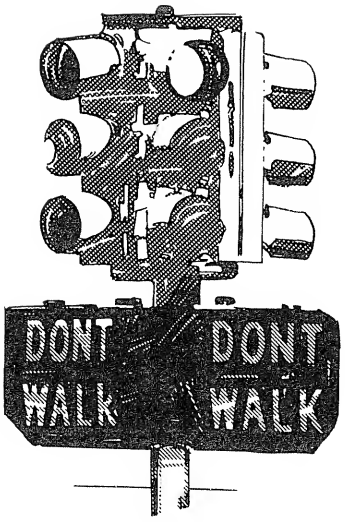
End of the Game

The game can end in one of three ways. The most desirable, and the one requiring the most political acumen, is for you to complete satisfactorily the transportation plan. The second way is to serve out the ten years but not complete the plan, which results in a demotion for you. The third ending is that you will be asked to resign. This will happen if you fail to keep the support of at least six Commissioners. And, it's easier to lose votes than it is to gain them.

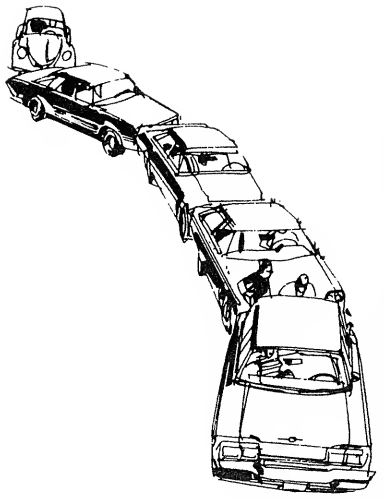
Good luck on your new job! □

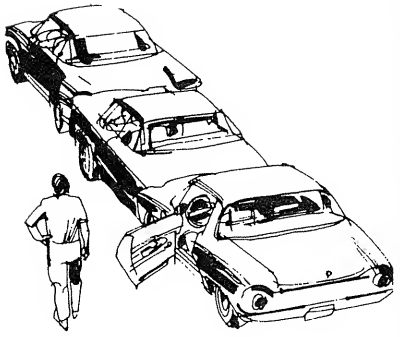
```
1 CLEAR 1500:DEFSTR F:DIM T$(6),T(6,11),TB(9,10),A(8,2),B(2,10),S(5,11),U(10),M(
10):C$=CHR$(207):D$=CHR$(212):E$=CHR$(255)
50 CLS:FOR X=10 TO 115:SET(X,3):NEXT
51 FOR Y=4 TO 44:SET(10,Y):SET(115,Y):NEXT
52 FOR X=10 TO 115:SET(X,44):NEXT
53 PRINT@213,"STREETS OF THE CITY";
54 PRINT@605,"BY";
55 PRINT@661,"KENNETH R. MURRAY";
56 FOR X=1 TO 1500:NEXT X
100 CLS:PRINT"CONGRATULATIONS! YOU HAVE BEEN NAMED TRANSPORTATION"
105 PRINT"DIRECTOR OF RIVER CITY, MICHIGAN, A CENTRAL CITY WITH"
110 PRINT"A DECLINING POPULATION AND WHICH HAS SUFFERED DETERIORATION"
115 PRINT"OF ITS TRANSPORTATION SERVICES OVER THE LAST SEVERAL YEARS.":PRINT
120 PRINT"PRIOR TO YOUR BEING HIRED, THE CITY COMMISSION ADOPTED"
125 PRINT"A TEN-YEAR TRANSPORTATION PLAN TO RESTORE SERVICES FOR"
130 PRINT"BOTH STREETS AND BUSES TO AN ADEQUATE LEVEL. IT WILL BE"
135 PRINT"YOUR RESPONSIBILITY TO CARRY OUT THIS PLAN.":PRINT
140 PRINT"FOR THE STREET FUND, YOU WILL NEED TO CONSTRUCT SEVERAL"
145 PRINT"MILES OF INTERSTATE HIGHWAYS AND RECONSTRUCT MAJOR LOCAL"
150 PRINT"STREETS (CALLED PRIMARIES). YOU WILL ALSO NEED TO IMPROVE"
155 PRINT"STREET CONDITIONS AND TRAFFIC SAFETY.":PRINT
814 S$(1)="RIDERSHIP":S$(2)="FLEET AGE":S$(3)="DOWNTIME":S$(4)="SERVICE DELAY":S
$(5)="FLEET SIZE"
850 T$(2)="PRIMARY ST. MILEAGE":T$(3)="INTERSTATE MILEAGE"
860 T$(4)="STREET CONDITION INDEX":T$(5)="TRAFFIC SAFETY INDEX"
865 T$(1)="LOCAL ST. MILEAGE":T$(6)="VEHICLE MILES"
1000 FA="$$######,###.":FB="######,###":FC="$$##.##"
1002 FOR R=1 TO 8
1003 FOR C=1 TO 2
1004 READ A(R,C)
1005 NEXT C
1006 NEXT R
1007 DATA 128,191,384,431,640,687,704,767,896,959
1008 DATA 15,975,31,991,47,1007
1029 YR=0:CV=11:G1=0:G3=0:B=50:P1=120000:P2=75000:M9=RND(150)*1000
1030 CI=(((RND(250)+250)*1000)+1000000)/2
1040 MI=RND(5000)+35000
1050 T(1,YR)=450+RND(100):T(2,YR)=85+RND(25):T(3,YR)=0
1060 T(4,YR)=RND(50)*.1+6:T(5,YR)=RND(50)*.1+6
1065 XX=3000+RND(3000):XY=7000+RND(3000):XZ=20000+RND(8000)
1070 T(6,YR)=(XX*T(1,YR))+(XY*T(2,YR))
1080 TB(1,YR)=(T(6,YR)/1.6)+300000:PT=TB(1,YR)*(((30+RND(20))*.01)):TB(8,YR)=TB(
1,YR)+PT:TB(2,YR)=1
1090 TB(3,YR)=(RND(500)*1000)+2100000:TB(9,YR)=TB(3,YR)
2000 MN=(T(1,YR)*M1*.16*(T(4,YR)*.1))+(T(2,YR)*MI*.5*(T(4,YR)*.1))+(T(3,YR)*MI)
2010 SN=MN*.04*T(5,YR):TB(6,YR)=MN:TB(7,YR)=SN
2011 S(1,YR)=RND(350)*1000+550000:S(5,YR)=INT(RND(10)+15):M1=INT(RND(3000)+5000)
2012 BF=0:M3=200:FOR X=1 TO S(5,YR)
2013 BF=BF+RND(12)+3
2014 NEXT X
2015 S(2,YR)=INT((BF/S(5,YR))*10)*.1:M2=(M1*S(5,YR))+(M3*BF)
2018 S=(RND(300)+500)*.01:S1=6:S2=12:S3=6:M5=(S1*S2*S3*312*S)+M9
2021 S4=.35:B(1,YR)=(RND(500)*1000)+200000
2022 S(3,YR)=INT(S(2,YR)/3)+6+(RND(50)*.1):S(4,YR)=INT(S(3,YR)/3)+6+(RND(50)*.1)
:S(1,YR)=S(1,YR)-(((S(3,YR)+S(4,YR))*.01)*S(1,YR)):BE=M5:M2=M2+(M5*.1):BD=M2
2030 T(4,11)=INT(T(4,YR)*.4):T(5,11)=INT(T(5,YR)*.4):S(1,11)=S(1,YR)*4:S(2,11)=I
NT(S(2,YR)*.4):S(3,11)=INT(S(3,YR)*.4):S(4,11)=INT(S(4,YR)*.4):T(2,11)=44+T(2,YR
):T(3,11)=16
2040 INPUT"PRESS ENTER";Z:CLS:PRINT"FOR THE TRANSIT AUTHORITY, YOU HAVE TO REPLA
```

```
CE A"
2041 PRINT"DELAPIDATED BUS FLEET, INCREASE RIDERSHIP, REDUCE THE"
2042 PRINT"MAINTENANCE DOWNTIME, AND IMPROVE ON-SCHEDULE PERFORMANCE"
2043 PRINT"(ALSO REFERRED TO AS SERVICE DELAY).":PRINT
2044 PRINT"FOR ALL INDICES USED, THE HIGHER THE INDEX VALUE THE"
2045 PRINT"WORSE THE CONDITION INDICATED.  THE BUDGET NEEDS LISTED"
2046 PRINT"ARE THE MINIMUMS NEEDED TO MAINTAIN THE INDEX AT ITS "
2047 PRINT"PRESENT LEVEL; IMPROVING THE LEVEL REQUIRES BUDGETS THAT"
2048 PRINT"ARE HIGHER THAN THE MINIMUM NEEDS.":PRINT@960,"PRESS ENTER";:INPUT Z
2049 CLS:PRINT:PRINT"YOUR GOALS FOR THE PLAN ARE AS FOLLOWS:":PRINT
2050 PRINT"STANDARD";TAB(30)"PRESENT";TAB(45)"GOAL"
2051 PRINT
2052 FOR X=2 TO 5:PRINT T$(X);TAB(30)T(X,YR);TAB(45)T(X,11):NEXT X
2053 PRINT
2054 FOR X=1 TO 4
2055 IF X=1 THEN 2058
2056 PRINT S$(X);TAB(30)S(X,YR);TAB(45)S(X,11):NEXT X
2057 GOTO 2060
2058 PRINT S$(X);TAB(30);:PRINT USING FB;S(X,YR);:PRINT TAB(45);:PRINT USING FB;
S(X,11):NEXT X
2060 PRINT:PRINT"GOOD LUCK!":PRINT@935,"PRESS ENTER";:INPUT Z
2990 YR=YR+1
2991 FOR X=1 TO 5
2992 S(X,YR)=S(X,YR-1)

2993 NEXT X
3010 FOR X=1 TO 6
3020 T(X,YR)=T(X,YR-1)
3030 NEXT X
3040 FOR X=1 TO 9
3050 TB(X,YR)=TB(X,YR-1)
3051 NEXT X
3052 B(1,YR)=B(1,YR-1):B(2,YR)=B(2,YR-1)::IF RND(10)<5 THEN GG=(RND(6)*P1) ELSE
GG=0
3053 B(2,YR)=B(2,YR)+GG:IF B(2,YR)>P1*10 THEN B(2,YR)=P1*10
3065 IF YR=1 THEN 3071
3070 I=(RND(10)+5)*.01::P1=P1+(P1*I):P2=P2+(P2*I):M9=M9+(M9*I)
3071 B(1,YR)=B(1,YR-1):B(2,YR)=B(2,YR-1):IF RND(10)<5 THEN GG=RND(6)*P1 ELSE GG=
0
3072 B(2,YR)=B(2,YR)+GG:IF YR=1 THEN 3241
3080 MI=MI+(MI*I):CI=CI+(CI*I):TB(1,YR)=TB(1,YR)+(TB(1,YR)*I):IF YR>2 AND T(4,YR
)>T(4,YR-2) THEN CI=CI*1.1
3081 M1=M1+(M1*I):M3=M3+(M3*I):M2=(M1*S(5,YR))+(M3*BF)
3100 IF I>.11 THEN T(1,YR)=T(1,YR)+RND(7)
3110 IF I<=.11 AND I>.08 THEN T(1,YR)=T(1,YR)+RND(15)
3120 IF I<=.08 THEN T(1,YR)=T(1,YR)+RND(22)
3190 T(6,YR)=T(6,YR)+(XX*(T(1,YR)-T(1,YR-1)))+(XY*(T(2,YR)-T(2,YR-2)))+(XZ*(T(3,
YR)-T(3,YR-2)))
3200 TB(1,YR)=((T(6,YR))/1.6)+(TB(1,YR)*I):PT=PT+(PT*(I+.02))+(PT*(((S(5,YR)-S(5
,YR-1))/S(5,YR-1))/2))
3210 TB(3,YR)=TB(3,YR)+(TB(3,YR)*(I-.02)):S(1,YR)=S(1,YR)+(S(1,YR)*.02)
3220 TB(9,YR)=TB(9,YR)+TB(3,YR)+B1:TB(8,YR)=TB(8,YR)+TB(1,YR)
3230 MN=(T(1,YR)*MI*.16*(T(4,YR)*.1))+(T(2,YR)*MI*.5*(T(4,YR)*.1))+T(3,YR)*MI
3235 MN=(MN*.6)+((MN*.4)*(1+U*.01))
3240 SN=MN*.04*T(5,YR):SN=(SN*.6)+((SN*.4)*(1+U*.01))
3241 CLS:PRINT"YOUR TRANSIT AUTHORITY SERVICE OPTIONS ARE:":PRINT
3242 PRINT TAB(5)"1.  ROUTES":PRINT TAB(5)"2.  HOURS OF OPERATION":PRINT TAB(5)"
3.  DAYS OF SERVICE":PRINT TAB(5)"4.  FARE":PRINT TAB(5)"5.  TO CONTINUE"
3243 PRINT:INPUT"WHAT IS YOUR CHOICE";Z:IF INT(Z)<>Z AND Z<1 OR Z>5 THEN 3241
3247 ON Z GOTO 3248,3254,3262,3270,3286
3248 PRINT@640,"PRESENT NUMBER OF ROUTES=";S1
3249 PRINT@704,"NEW NUMBER OF ROUTES (MIN. OF 6, MAX. OF 25)";
3250 INPUT S1(1):PRINT@960,E$;:IF INT(S1(1))<>S1(1) THEN 3280
3251 IF S1(1)<6 THEN 3281
3252 IF S1(1)>25 THEN 3281 ELSE 3241
3254 PRINT@640,"YOUR POSSIBLE HOURS OF OPERATION ARE LISTED BELOW":IF S2=12 THEN
 S2=1
3255 IF S2=17 THEN S2=2
3256 IF S2=24 THEN S2=3
3257 PRINT@704,"CURRENT OPTION=";S2:PRINT@768,"1. 6 AM TO 6 PM":PRINT@808,"2. 6
AM TO 11 PM":PRINT@832,"3. 24 HOURS":PRINT@872,"NEW HOURS";:INPUT S2(1):PRINT@96
0,E$;:IF INT(S2(1))<>S2(1) OR S2(1)<1 OR S2(1)>3 THEN 3282
3258 IF S2(1)=1 THEN S2(1)=12
3259 IF S2(1)=2 THEN S2(1)=17
3260 IF S2(1)=3 THEN S2(1)=24
3261 GOTO 3241
3262 PRINT@640,"YOUR OPTIONS FOR DAYS OF SERVICE ARE AS FOLLOWS"
3263 IF S3=6 THEN S3=1 ELSE S3=2
3264 PRINT@704,"CURRENT OPTION=";S3:PRINT@768,"1. MONDAY THROUGH SATURDAY"
3265 PRINT@832,"2. MONDAY THROUGH SUNDAY"
3266 PRINT@872,"NEW DAYS =";:INPUT S3(1):PRINT@960,E$;
3267 IF INT(S3(1))=S3(1) AND S3(1)=1 OR S3(1)=2 THEN 3268 ELSE 3283
3268 IF S3(1)=1 THEN S3(1)=6 ELSE S3(1)=7
3269 GOTO 3241
3270 PRINT@640,"THE FARE MAY BE CHANGED IN NICKEL UNITS, WITH A"
```

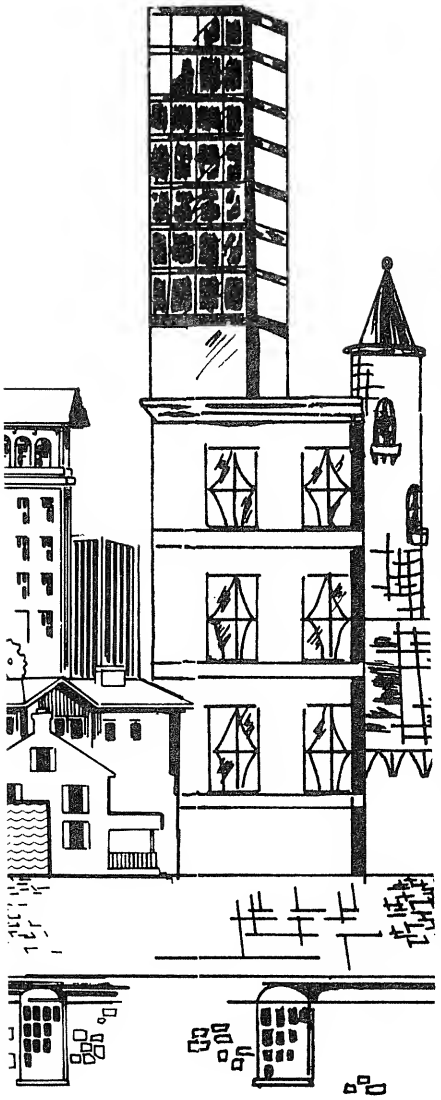



```
3271 PRINT@704,"MINIMUM FARE OF $.25 AND A MAXIMUM OF $1.00"
3272 PRINT@768,"DO NOT ENTER DOLLAR SIGN"
3273 PRINT@832,"CURRENT FARE =";S4:PRINT@872,"NEW FARE =";
3274 INPUT S4$:PRINT@960,E$;:IF RIGHT$(S4$,1)="0" OR RIGHT$(S4$,1)="5" THEN 3275
 ELSE 3285
3275 S4(1)=VAL(S4$):IF S4(1)<.25 OR S4(1)>1 THEN 3284
3277 GOTO 3241
3279 GOTO 3286
3280 PRINT@960,"YOU MUST ENTER A WHOLE NUMBER";:PRINT@704,E$:GOTO 3249
3281 PRINT@960,"YOUR NUMBER IS OUTSIDE THE RANGES";:PRINT@704,E$:GOTO 3249
3282 PRINT@960,"YOUR OPTION MUST BE 1, 2, OR 3";:PRINT@832,E$:GOTO 3257
3283 PRINT@960,"YOUR OPTION MUST BE 1 OR 2";:PRINT@832,E$:GOTO 3265
3284 PRINT@960,"YOUR FARE IS OUTSIDE THE RANGE";:PRINT@832,E$:GOTO 3273
3285 PRINT@960,"THE FARE MUST BE IN NICKEL INCREMENTS";:PRINT@832,E$:GOTO 3273
3286 S6=0:IF S1(1)=0 AND S2(1)=0 AND S3(1)=0 AND S4(1)=0 THEN 3302
3287 IF S3(1)<>0 AND S3(1)-S3=5 THEN S6=S6+((RND(15000)+75000)*S1)
3288 IF S3(1)<>0  AND  S3(1)-S3=7 THEN S6=S6+((RND(25000)+90000)*S1)
3289 IF S3(1)<>0 AND  S3(1)-S3=12 THEN S6=S6+((RND(30000)+110000)*S1)
3291 IF  S3(1)<>0 AND  S3-S3(1)=5 THEN S6=S6-((RND(15000)+75000)*S1)
3292 IF S3(1)<>0 AND  S3-S3(1)=7 THEN S6=S6-((RND(25000)+90000)*S1)
3293 IF S3(1)<>0 AND  S3-S3(1)=12 THEN S6=S6-((RND(30000)+110000)*S1)
3294 IF S1(1)<>0 THEN S6=S6+(((100+RND(120))*500)*(S1(1)-S1))
3295 IF S2(1)<>0 AND S2(1)>0 THEN S6=S6+(RND(10000)+15000)*(S2(1)-S2)
3296 IF S4(1)<>0 AND S4(1)>S4 THEN S6=S6-(((S4(1)-S4)/5)*2000)
3297 IF S1(1)>0 THEN S1=S1(1)
3298 IF S2(1)>0 THEN S2=S2(1)
3299 IF S3(1)>0 THEN S3=S3(1)
3300 IF S4(1)>0 THEN S4=S4(1)
3302 S1(1)=0:S2(1)=0:S3(1)=0:S4(1)=0
3303 IF YR=3 OR YR=7 THEN 3304 ELSE 3680
3304 B9=B1:B8=DS:IF YR=3 THEN B1=1500000 ELSE B1=2000000
3309 CLS:PRINT TAB(15)"STREET FUND BOND PROPOSAL":PRINT:PRINT"YOU MAY PROPOSE BO
NDING UP TO";
3310 PRINT USING FA;B1;
3311 PRINT"SUBJECT TO"
3315 PRINT"APPROVAL OF THE CITY COMMISSION AND A VOTE OF THE"
3320 PRINT"CITIZENS.  HOW MUCH DO YOU WISH TO PROPOSE (IN"
3325 INPUT"THOUSANDS, TYPE '0' IF NONE)";Z
3326 IF Z=0 THEN 3675
3330 Z=Z*1000:IF Z<0 OR Z>B1 THEN 3300
3331 B1=Z
3335 IF CV<.8 THEN B1=B1-(RND(35)*10000)
3340 IF CV<.10 THEN B1=B1-(RND(20)*10000)
3345 PRINT:PRINT"THE COMMISSION HAS APPROVED A BOND REFERENDUM"
3350 PRINT"FOR";
3355 PRINT USING FA;B1;
3356 PRINT"EACH YEAR.":PRINT:PRINT:INPUT"PRESS ENTER";Z
3360 CLS:PRINT:PRINT"THE COALITION OF NEIGHBORHOOD ASSOCIATIONS HAS ASKED"
3365 PRINT"YOU TO MAKE THE FOLLOWING PLEDGES FOR THE NEXT THREE"
3370 PRINT"YEARS.  WILL YOU MAKE ANY OF THEM (Y/N)?":PRINT
3375 IF T(4,YR)>T(4,0) THEN B2=T(4,YR)-2 ELSE B2=T(4,0)-2
3380 IF B2<1 THEN B2=1
3385 IF T(5,YR)>T(5,0) THEN B3=T(5,YR)-2 ELSE B2=T(5,0)-2
3390 IF B3<1 THEN B3=1
3395 IF G1<22 THEN B4=44-G1 ELSE B4=20
3400 IF G3<11 THEN B5=16-G3 ELSE B5=6
3405 PRINT TAB(5)"1.  IMPROVE STREET CONDITION INDEX TO";B2
3410 PRINT TAB(5)"2.  IMPROVE SAFETY INDEX TO";B3
3415 PRINT TAB(5)"3.  CONSTRUCT";B4;"MILES OF PRIMARIES"
3420 PRINT TAB(5)"4.  CONSTRUCT";B5;"MILES OF INTERSTATES"
3423 PRINT:PRINT@768,"PLEDGE 1";TAB(15)"PLEDGE 2";TAB(30)"PLEDGE 3";TAB(45)"PLED
GE 4"
3424 PRINT@832,C$
3425 PRINT@832,;
3426 INPUT Z$
3430 IF Z$<>"Y" AND Z$<>"N" THEN 3405
3435 IF Z$="N" THEN B2=0
3439 PRINT@847,C$
3440 PRINT@847,;
3441 INPUT Z$
3445 IF Z$<>"Y" AND Z$<>"N" THEN 3410
3450 IF Z$="N" THEN B3=0
3454 PRINT@862,C$
3455 PRINT@862,;:INPUT Z$
3460 IF Z$<>"Y" AND Z$<>"N" THEN 3415
3465 IF Z$="N" THEN B4=0
3469 PRINT@877,C$
3470 PRINT@877,;:INPUT Z$
3475 IF Z$<>"Y" AND Z$<>"N" THEN 3420
3480 IF Z$="N" THEN B5=0
3485 PRINT@960,"PRESS ENTER FOR ELECTION RESULTS";
3490 INPUT Z:CLS:PRINT TAB(18)"BOND ELECTION RESULTS"
3492 PRINT:PRINT"WARD";TAB(10)"YES";TAB(20)"NO";TAB(35)"TOT. YES";TAB(50)"TOT. N
O"
```



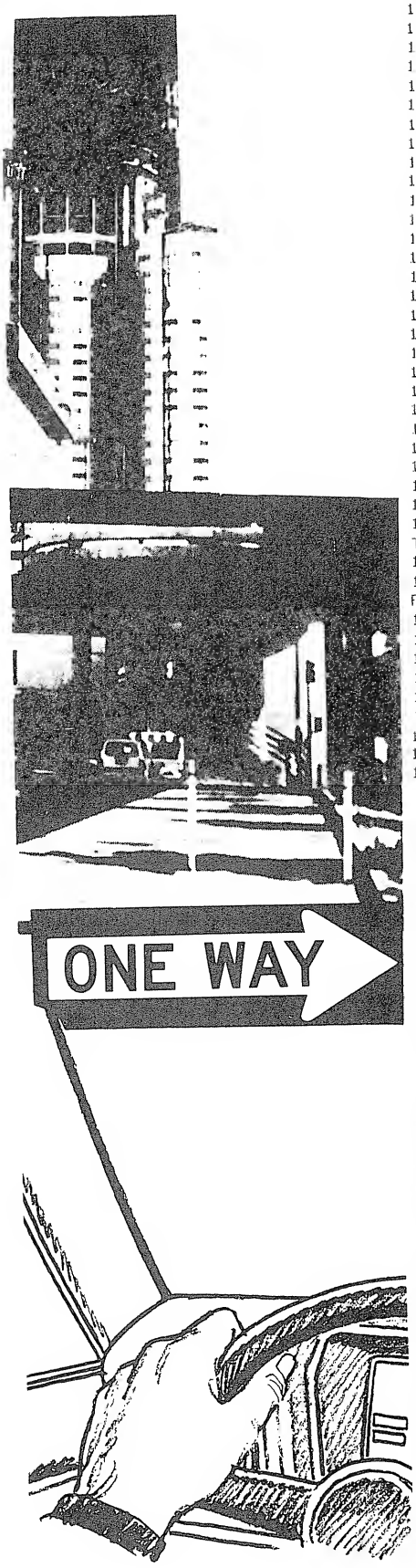


```
3495 PRINT:V5=0:V6=0
3500 IF CV>9 THEN V1=5000 ELSE V1=4000
3505 IF T(4,YR)<T(4,YR-1) AND T(4,YR)<T(4,0) THEN V1=V1+500
3510 IF T(5,YR)<T(5,YR-1) AND T(5,YR)<T(5,0) THEN V1=V1+500
3515 IF YR=7 THEN 3530
3520 IF B1<1100000 THEN V1=V1+500
3525 GOTO 3535
3530 IF B1<1600000 THEN V1=V1+500
3535 IF B2>0 THEN V1=V1+500
3540 IF B3>0 THEN V1=V1+500
3545 IF B4>0 THEN V1=V1+500
3550 IF B4>0 THEN V1=V1+500
3555 V2=RND(5)
3560 FOR X=1 TO 5
3565 IF X=V2 THEN 3585
3575 V3=V1+RND(11000)
3580 GOTO 3590
3585 V3=(V1/2)+RND(12000)
3586 IF V3<3000 THEN V3=4000
3590 V4=21000-V3:V5=V5+V3:V6=V6+V4
3595 PRINT TAB(2)X;TAB(9)V3;TAB(19)V4;TAB(35)V5;TAB(50)V6
3600 FOR Y=1 TO 500:NEXT Y
3605 NEXT X
3610 IF V5<=V6 THEN 3650
3615 PRINT:PRINT"CONGRATULATIONS.  THE BOND ISSUE WAS APPROVED BY"
3620 PRINT"THE VOTERS.  YOUR ANNUAL DEBT PAYMENT WILL BE";:DS=PT*.35
3630 PRINT USING FA;DS
3635 TB(9,YR)=TB(9,YR)+B1:B=B-10:DS=DS+B8:B1=B1+B9
3640 INPUT"PRESS ENTER";Z:GOTO 3680
3650 PRINT"THE REFERENDUM HAS FAILED."
3651 IF V6/(V5+V6)<(55+RND(15))*.01 THEN 3660
3652 PRINT"BECAUSE OF THE MARGIN OF DEFEAT, YOU HAVE LOST"
3653 PRINT"THE VOTE OF A COMMISSIONER.": CV=CV-1
3655 IF CV<6 THEN 6770
3660 B1=B9:B2=0:B3=0:B4=0:B5=0
3675 PRINT:INPUT"PRESS ENTER";Z
3680 CLS:PRINT TAB(18)"PROPERTY TAX LEVY"
3681 M5=(S1*S2*S3*S12*S)+M9:M2=M2+(M5*.1):B(1,YR)=B(1,YR)+(S4*S(1,YR))+((M2+M5)/
2)
3685 PRINT TAB(30)"STREET FUND";TAB(45)"TRANSIT AUTHORITY"
3690 PRINT"OPERATING NEEDS";TAB(30);:PRINT USING FA;MN+SN+DS;:PRINT TAB(45);:PRI
NT USING FA;M2+M5
3695 PRINT"NON-TAX REVENUE";TAB(30);:PRINT USING FA;TB(8,YR);:PRINT TAB(45);:PRI
NT USING FA;B(1,YR)
3699 IF M2+M5-B(1,YR)<0 THEN X1=0 ELSE X1=M2+M5-B(1,YR)
3700 PRINT"PROPERTY TAX NEEDED (MILLS)";:PRINT TAB(30);:PRINT USING FA;MN+SN+DS-
TB(8,YR);:PRINT TAB(45);:PRINT USING FA;X1
3705 TN=INT(((MN+SN+DS+X1-TB(8,YR))/PT)*10)*.1:PRINT:PRINT"YIELD OF ONE MILL =";
:PRINT USING FA;PT:PRINT "TOTAL PROPERTY TAX NEEDED (IN MILLS) =";TN
3710 PRINT@640,"WHAT PROPERTY TAX LEVY (0-10 MILLS) DO YOU PROPOSE";:INPUT TB(C,
YR):PRINT@960,E$;
3715 IF TB(2,YR)>=0 AND TB(2,YR)<=10 THEN 3725
3720 PRINT@960,"YOU CANNOT EXCEED THE LIMITS";:PRINT@640,E$:GOTO 3710
3725 IF TB(2,YR)>=TB(2,YR-1) THEN 3805
3770 X1=0:X2=0
3775 FOR X=1 TO 11
3780 IF CV>9 THEN 3783
3781 IF X<=CV THEN X3=RND(5) ELSE X3=RND(8)
3782 GOTO 3785
3783 IF X<=2 THEN X3=RND(4)
3784 IF X>2 AND X<=CV THEN X3=RND(5):IF X>CV THEN X3=RND(8)
3785 IF X3<=3 THEN X1=X1+1 ELSE X2=X2+1
3790 NEXT X
3795 IF X1>=6 THEN 3801
3800 IF TB(2,YR)<=TN THEN TB(2,YR)=TB(2,YR) ELSE TB(2,YR)=TB(2,YR)-(.1*(X2+1))
3801 IF X1>=6 THEN 3805:IF TB(2,YR)>=TN+2 AND X1>10 THEN TB(2,YR)=TB(2,YR)-(2*(X
2+1))
3805 PRINT@704,"THE CITY COMMISSION HAS APPROVED A LEVY OF";TB(2,YR);"MILLS"
3810 PRINT@768,"HOW MANY MILLS ARE FOR THE STREET FUND";:INPUT T8:PRINT@960,E$;
3815 IF T8<=TB(2,YR) THEN 3825
3820 PRINT@960,"YOU CANNOT ALLOCATE MORE THAN YOU ARE AUTHORIZED";:PRINT@768,E$:
GOTO 3810
3825 TB(8,YR)=TB(8,YR)+(PT*T8)-DS:B(1,YR)=B(1,YR)+(PT*(TB(2,YR)-T8))
4000 CLS:C=0:GOTO 4020
4010 C=1
4020 CLS:PRINT TAB(10)"STREET FUND BUDGET DECISIONS FOR YEAR";YR
4030 PRINT"OPERATIONS:";TAB(33)"CONSTRUCTION:"
4040 PRINT TAB(5)"AVAILABLE:";
4041 PRINT USING FA;TB(8,YR);
4050 PRINT TAB(38)"AVAILABLE:";:PRINT USING FA;TB(9,YR)
4060 PRINT TAB(5)"MAINT. NEED=";:PRINT USING FA;MN;
4070 PRINT TAB(38)"COST PER HALF MILE UNIT:"
4080 PRINT TAB(5)"SAFETY NEED=";:PRINT USING FA;SN;
```

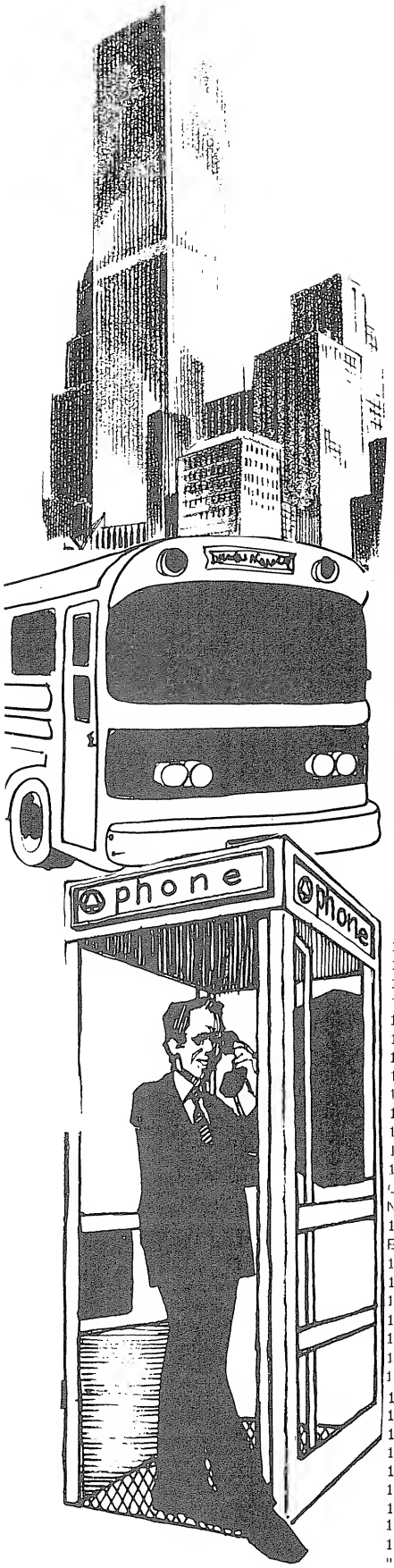


```
4090 PRINT TAB(38)"PRIMARY RDS.=";:PRINT USING FA;CI*.2
4100 PRINT TAB(38)"INTERSTATES=";:PRINT USING FA;CI
4110 PRINT:IF C=1 THEN 4260
4150 PRINT"YOU MAY TRANSFER UP TO ";B;"% FROM AN ACCOUNT"
4160 PRINT TAB(10)"1.  OPERATIONS TO CONSTRUCTION"
4170 PRINT TAB(10)"2.  CONSTRUCTION TO OPERATIONS"
4180 PRINT TAB(10)"3.  NO TRANSFER"
4190 INPUT Z:IF Z<>1 AND Z<>2 AND Z<>3 THEN 4020
4195 IF Z=3 THEN 4010 ELSE 4200
4200 INPUT"HOW MUCH DO YOU WANT TO TRANSFER (IN THOUSANDS, WITHOUT $ SIGN)";T:T=
T*1000
4210 IF Z=1 AND T>TB(8,YR)/(B*.01) THEN 4200
4220 IF Z=2 AND T>TB(9,YR)/(B*.01) THEN 4200
4230 IF Z=2 THEN 4250
4240 TB(8,YR)=TB(8,YR)-T:TB(9,YR)=TB(9,YR)+T:GOTO 4010
4250 TB(8,YR)=TB(8,YR)+T:TB(9,YR)=TB(9,YR)-T:GOTO 4010
4260 PRINT"ENTER CONSTRUCTION BY THE NUMBER OF HALF MILE UNITS;"
4265 PRINT"ENTER MAINTENANCE AND SAFETY BY THOUSAND DOLLAR UNITS."
4270 PRINT"DO NOT USE COMMAS OR DOLLAR SIGNS"
4280 PRINT:PRINT TAB(10)"PRIMARIES";TAB(20);"INTERSTATES";TAB(36)"MAINTENANCE";T
AB(53)"SAFETY"
4290 PRINT"LAST YR";TAB(13)PC;TAB(23)IC;
4300 PRINT TAB(35);
4301 PRINT USING FA;TB(6,YR-1);
4305 PRINT TAB(50);
4306 PRINT USING FA;TB(7,YR-1)
4310 PRINT"THIS YR";
4315 PRINT@844,;
4316 INPUT PC
4317 PRINT@960,E$;
4318 IF INT(PC)<>PC THEN 4390
4319 IF G1+(PC/2)>44 THEN 4392
4320 PRINT@854,;
4321 INPUT IC
4322 PRINT@960,E$;
4323 IF INT(IC)<>IC THEN 4393
4324 IF T(3,YR)+(IC/2)>16 THEN 4395
4325 IF (PC*(CI*.2))+(IC*CI)>TB(9,YR) THEN 4397
4326 PRINT@870,;
4327 INPUT T1:T1=T1*1000:PRINT@960,E$;:IF LEN(STR$(T1))>LEN(STR$(MN))+1 OR T1>10
0000 THEN GOSUB 4399 ELSE 4330
4328 IF Z$="Y" THEN 4330
4329 Z$="Y":PRINT@870,D$:GOTO 4326
4330 PRINT@885,;:INPUT T2:PRINT@960,E$;:T2=T2*1000:IF LEN(STR$(T2))>LEN(STR$(SN)
)+1 OR T2>100000 THEN GOSUB 4399 ELSE 4334
4331 IF Z$="Y" THEN 4334
4332 Z$="Y":PRINT@855,D$:GOTO 4330
4334 IF T1+T2>TB(8,YR) THEN 4405
4335 GOTO 4440
4390 PRINT@960,"YOU MUST ENTER A WHOLE NUMBER.";
4391 PRINT@841,C$:GOTO 4315
4392 PRINT@960,"YOU CANNOT BUILD THAT MANY MORE UNITS";:GOTO 4391
4393 PRINT@960,"YOU MUST ENTER A WHOLE NUMBER.";
4394 PRINT@851,C$:GOTO 4320
4395 PRINT@960,"YOU CANNOT BUILD THAT MANY MORE UNITS";:GOTO 4394
4397 PRINT@960,"YOUR CONSTRUCTION PROGRAM EXCEEDS YOUR BUDGET.";
4398 PRINT@841,C$:PRINT@851,C$:GOTO 4315
4399 PRINT@960,E$;:PRINT@960,"ARE YOU SURE (Y/N)";
4400 INPUT Z$:IF Z$<>"Y" AND Z$<>"N" THEN 4399
4401 RETURN:END
4402 PRINT@960,E$;:PRINT@960,"ARE YOU SURE (Y/N)";
4403 INPUT Z$:IF Z$<>"Y" AND Z$<>"N" THEN 4402:IF Z$="Y" THEN 4332
4404 PRINT@885,D$:GOTO 4330
4405 PRINT@960,"YOUR MAINTENANCE AND SAFETY BUDGET EXCEEDS YOUR FUNDS";
4406 PRINT@870,D$:PRINT@885,D$:GOTO 4326
4440 TB(6,YR)=T1:TB(7,YR)=T2
4445 T(2,YR)=T(2,YR)+PC/2:T(3,YR)=T(3,YR)+IC/2
4450 TB(8,YR)=TB(8,YR)-TB(6,YR)-TB(7,YR)
4460 TB(9,YR)=TB(9,YR)-(PC*CI*.2)-(IC*CI)
4470 PRINT@960,"PRESS ENTER";:INPUT Z
5200 T(4,YR)=T(4,YR)-(INT(((TB(6,YR)-MN)/MN)*10)*.1)
5210 IF T(4,YR)<1 THEN T(4,YR)=1
5240 T(5,YR)=T(5,YR)-(INT(((TB(7,YR)-SN)/SN)*10)*.1)
5249 IF T(4,YR)>T(4,YR-1) THEN T(5,YR)=T(5,YR)+.2
5250 IF T(5,YR)<1 THEN T(5,YR)=1
6000 CLS:C=0:GOTO 6020
6010 C=1
6020 CLS:PRINT TAB(15)"TRANSIT BUDGET FOR YEAR";YR
6030 PRINT"OPERATIONS";TAB(33)"BUS FLEET"
6040 PRINT TAB(5)"AVAILABLE";:PRINT USING FA;B(1,YR);:PRINT TAB(38)"AVAILABLE";:
PRINT USING FA;B(2,YR)
6080 PRINT TAB(5)"MAINT. NEED=";:PRINT USING FA;M2;:PRINT TAB(38)"(COST PER BUS)
:"
6090 PRINT TAB(5)"OPERATIONS NEED=";:PRINT USING FA;M5;:PRINT TAB(38)"ACQUISITIO
```

```
N=";:PRINT USING FA;P1
6100 PRINT TAB(38)"SALE=";:PRINT USING FA;P2
6110 IF C=1 THEN 6340
6120 IF GG>0 THEN PRINT@640,"BECAUSE OF THE FEDERAL GRANT, YOU CANNOT TRANSFER F
ROM":PRINT@704,"OPERATIONS TO THE BUS FLEET":FOR X=1 TO 750:NEXT X:GOTO 6200
6140 PRINT@640,"YOU MAY TRANSFER UP TO 25% FROM OPERATIONS TO ACQUISITION":PRINT
@960,E$;
6150 PRINT@704,"HOW MUCH DO YOU WISH TO TRANSFER (ENTER AMTS. IN THOUSANDS WITHO
UT $ SIGN)";:INPUT Z:Z=Z*1000:IF Z<0 OR Z>B(1,YR)/4 THEN 6180 ELSE 6190
6180 PRINT@640,E$:PRINT@704,E$:GOTO 6140
6190 B(2,YR)=B(2,YR)+Z:B(1,YR)=B(1,YR)-Z
6200 PRINT@640,E$:PRINT@704,E$
6210 PRINT@640,"HOW MANY BUSES DO YOU WISH TO SELL";:INPUT NB:IF NB>S(5,YR) OR N
B<0 OR INT(NB)<>NB THEN 6230 ELSE 6240
6230 PRINT@640,E$:GOTO 6210
6240 IF NB=0 THEN 6010
6250 BF=BF-((S(2,YR)*2)*NB):S(5,YR)=S(5,YR)-NB:M2=(M1*S(5,YR))+(M3*BF)
6331 B(2,YR)=B(2,YR)+(NB*P2):GOTO 6010
6340 PRINT:PRINT"ENTER BUDGETS IN THOUSAND DOLLAR UNITS. DO NOT"
6350 PRINT"USE COMMAS OR DOLLAR SIGNS"
6360 PRINT:PRINT TAB(22)"MAINTENANCE";TAB(36)"OPERATIONS";TAB(50)"NEW BUSES"
6370 PRINT"LAST YEAR";TAB(20);:PRINT USING FA;BD;:PRINT TAB(35);:PRINT USING FA;
BE;:PRINT TAB(53);BN
6375 PRINT"THIS YEAR";
6400 PRINT@793,;:INPUT BD:PRINT@960,E$;:BD=BD*1000
6401 IF BD<0 THEN GOSUB 6690
6402 IF BD<0 THEN 6400
6403 IF BD>100000 THEN GOSUB 4399 ELSE 6410
6404 IF Z$="Y" THEN 6410
6405 Z$="Y":PRINT@841,D$:GOTO 6400
6410 IF LEN(STR$(BD))>LEN(STR$(M2))+1 THEN 6600
6420 PRINT@806,;:INPUT BE:PRINT@960,E$;:BE=BE*1000
6421 IF BE<0 GOSUB 6690
6422 IF BE<0 THEN 6420
6423 IF BE>100000 GOSUB 4399 ELSE 6430
6424 IF Z$="Y" THEN 6430
6425 PRINT@841,D$:GOTO 6420
6430 IF LEN(STR$(BE))>LEN(STR$(M5))+1 THEN 6630
6440 IF BD+BE>B(1,YR) THEN 6660
6450 PRINT@820,;:INPUT BN:PRINT@960,E$;
6460 IF BN>100-S(5,YR) THEN 6680
6470 IF BN<0 THEN GOSUB 6690
6471 IF BN<>0 THEN 6480
6472 PRINT@820,C$:GOTO 6450
6475 IF INT(BN)<>BN THEN 6710
6480 IF BN*P1>B(2,YR) THEN 6700
6490 B(2,YR)=B(2,YR)-(P1*BN):S(5,YR)=S(5,YR)+BN:BF=BF-BN
6500 BF=BF+S(5,YR):S(2,YR)=INT((BF/S(5,YR))*10)*.1
6580 B(1,YR)=B(1,YR)-(BD+BE)
6599 GOTO 6750
6600 PRINT@960,"ARE YOU SURE (Y/N)";:INPUT Z$
6610 IF Z$<>"Y" AND Z$<>"N" THEN 6600:IF Z$="Y" THEN 6420
6620 PRINT@792,D$:GOTO 6400
6630 PRINT@960,"ARE YOU SURE (Y/N)";:INPUT Z$
6640 IF Z$<>"Y" AND Z$<>"N" THEN 6630: IF Z$="Y" THEN 6440
6650 PRINT@806,D$:GOTO 6420
6660 PRINT@960,"YOUR OPERATING AND MAINTENANCE BUDGETS EXCEED YOUR FUNDS";:PRINT
@792,D$:PRINT@806,D$:GOTO 6400
6680 PRINT@960,"YOU CAN PURCHASE ONLY";100-S(5,YR);"BUSES";:GOTO 6450
6690 PRINT@960,"YOU CANNOT ENTER A NEGATIVE NUMBER";:RETURN:END
6700 PRINT@960,"YOUR PROPOSED ACQUISITION EXCEEDS YOUR BUDGET";:PRINT@820,C$:GOT
O 6450
6710 PRINT@960,"YOU MUST ENTER A WHOLE NUMBER";:PRINT@820,C$:GOTO 6450
6750 S(3,YR)=S(3,YR)-(INT(((BD-M2)/M2)*18)*.1):IF S(2,YR)>S(2,YR-1) THEN S(3,YR)
=S(3,YR)+.2
6751 IF S(3,YR)<1 THEN S(3,YR)=1
6755 S(4,YR)=S(4,YR)-(INT(((BE-M5)/M5)*18)*.1):IF S(3,YR)>S(3,YR-1) THEN S(4,YR)
=S(3,YR)+.2
6756 IF S(5,YR)/S1<3 THEN S(4,YR)=S(4,YR)+.2
6757 IF S(4,YR)<1 THEN S(4,YR)=1
6760 S(1,YR)=(S(1,YR)+S6)-((S(3,YR)+S(4,YR)-S(3,YR-1)-S(4,YR-1))*S(1,YR)*.01)
10030 REM SALARY NEGOTIATIONS
10040 NR=RND(4)+2:I=I*100:IF I=0 THEN U(1)=INT(RND(8)+8) ELSE U(1)=INT(RND(1)*8)
10060 CLS:LS=0
10070 PRINT"YOUR PRESENT WAGE IS";S;"DOLLARS PER HOUR"
10110 PRINT"THE UNION'S INITIAL OFFER IS FOR A";U(1);"PERCENT INCREASE"
10120 INPUT"WHAT IS YOUR RESPONSE";M(1)
10130 CLS
10140 PRINT"PRESENT SALARY=$";S
10160 PRINT:PRINT"UNION","MANAGEMENT"
10170 PRINT"POSITION","POSITION"
10180 PRINT
10190 PRINT U(1),M(1)
10200 FOR X=2 TO NR
10210 IF X<>NR THEN 10240
```




```
10220 PRINT"THIS IS THE LAST ROUND OF NEGOTIATIONS.  FAILURE"
10230 PRINT"TO SETTLE COULD RESULT IN A STRIKE"
10240 UO=U(X-1)-M(X-1):IF UO=0 THEN 10250 ELSE 10270
10250 U(X)=M(X-1):U=U(X)
10260 GOTO 10440
10270 IF M(X-1)-M(X-2)>=5 THEN R3=1
10280 IF M(X-1)-M(X-2)<5 THEN R3=2
10290 IF M(X-1)-M(X-2)<3 THEN R3=3
10300 IF M(X-1)-M(X-2)<1 THEN R3=4
10310 IF UO<5 THEN U(X)=U(X-1)-((RND(40)*.1)/R3)
10320 IF UO<10 OR UO>5 THEN U(X)=U(X-1)-((RND(60)*.1)/R3)
10330 IF UO<15 THEN U(X)=U(X-1)-((RND(80)*.1)/R3)
10340 IF UO<20 THEN U(X)=U(X-1)-((RND(100)*.1)/R3)
10345 IF U(X)<=U(X-1) THEN U(X)=U(X-1)-.5
10350 IF U(X)>=M(X-1) THEN U(X)=M(X-1)
10360 U(X)=INT(U(X)*100)*.01
10370 U=U(X)
10380 PRINT U(X),
10390 IF U(X)=M(X-1) THEN 10440
10400 INPUT M(X)
10410 IF M(X)=U(X) THEN 10440
10420 NEXT X
10430 IF M(NR)<U(NR) THEN 10490
10440 S=INT(S*(100+U))*.01
10450 PRINT"YOU HAVE REACHED AGREEMENT ON A ";U;"PERCENT"
10460 PRINT"WAGE INCREASE.  YOUR HOURLY WAGE RATE IS NOW $";S
10480 GOTO 10640
10490 IF (U(NR)-M(NR))*RND(0)>.5 THEN 10502
10500 U(NR)=M(NR):U=U(NR)
10501 GOTO 10440
10502 CLS:PRINT CHR$(23):FOR X=40 TO 80:SET(X,6):NEXT:FOR Y=7 TO 32:SET(40,Y):SE
T(80,Y):NEXT:FOR X=40 TO 80:SET(X,32):NEXT:FOR Y=33 TO 45:SET(60,Y):NEXT
10503 PRINT@216,"WORKERS";:PRINT@282,"LOCAL";:PRINT@346,"10200";
10504 FOR X=1 TO 3:PRINT@540,"ON";:PRINT@600,"STRIKE";:FOR X1=1 TO 300:NEXT X1:I
F X=3 THEN 10530
10505 PRINT@540,CHR$(194);:PRINT@600,CHR$(198);:FOR X1=1 TO 300:NEXT X1:NEXT X
10530 FOR X=1 TO 500:NEXT:M(7)=M(NR)-M(1):U(7)=U(1)-U(NR):DP=U(NR)-M(NR)
10540 IF M(7)<U(7) THEN SS(1)=(DP*(RND(6)*.1)) ELSE SS(1)=(DP*((RND(6)+3)*.1))
10550 SS(2)=M(NR)+SS(1):U=INT(SS(2)*100)*.01
10560 LS=RND(5)+RND(DP+1)
10590 CLS:PRINT"THE STRIKE LASTED FOR";LS;"DAYS.  THE ARBITRATOR"
10600 PRINT"HAS ORDERED A SETTLEMENT OF";U;" PERCENT."
10610 PRINT"THIS RESULTS IN A WAGE OF";
10620 S=INT(S*(100+U))*.01:PRINT USING PC;S

10621 PRINT"AS A RESULT OF THE STRIKE,":PRINT
10623 GOSUB 10634
10624 T(4,YR)=T(4,YR)+X1:PRINT TAB(5) T$(4);" HAS INCREASED BY";X1
10625 GOSUB 10634
10626 T(5,YR)=T(5,YR)+X1:PRINT TAB(5)T$(5);" HAS INCREASED BY";X1
10627 IF PC<2 THEN 10630:IF LS<=7 THE PC=PC-2 ELSE GOTO 10629
10628 PRINT TAB(5)"CONSTRUCTION PROGRAM LOST ONE MILE":GOTO 10630
10629 PC=PC-1:PRINT TAB(5)"CONSTRUCTION PROGRAM LOST 1/2 MILE"
10630 PRINT:GOSUB 10634:S(3,YR)=S(3,YR)+X1:PRINT TAB(5) S$(3);" HAS INCREASED BY
";X1
10631 GOSUB 10634:S(4,YR)=S(4,YR)+X1:PRINT TAB(5)S$(4);" HAS INCREASED BY";X1
10633 PRINT:GOTO 10640
10634 IF LS<=7 THEN X1=RND(7)*.1 ELSE X1=RND(4)*.1
10635 RETURN:END
10640 INPUT"ENTER WHEN READY";Z
11000 CLS:PRINT TAB(10)"STREET FUND PERFORMANCE FOR YEAR";YR
11010 PRINT
11020 PRINT TAB(30)"YEAR";YR;TAB(40)"YEAR";YR-1;TAB(50)"PLAN"
11030 PRINT
11040 FOR X=2 TO 5
11050 PRINT T$(X);TAB(30)T(X,YR);TAB(40)T(X,YR-1);TAB(50)T(X,11)
11060 NEXT X
11070 PRINT:INPUT"TYPE '1' TO REVIEW THE STREET MAP, ELSE PRESS ENTER";Z:CLS:IF
Z=1 THEN 13000 ELSE 13330
13000 G2=0:G4=0:CLS:G1=G1+(PC/2)
13010 FOR X=1 TO 8
13020 IF X>5 THEN G5=64 ELSE G5=2
13030 IF X>5 THEN G6=.5 ELSE G6=.25
13040 IF X=4 OR X=6 THEN 13120
13050 FOR Y=A(X,1) TO A(X,2) STEP G5
13060 G2=G2+G6
13070 IF G2<=G1 PRINT@Y,"+";
13080 IF G2>G1 PRINT@Y,"-";
13090 NEXT Y
13100 NEXT X
13110 GOTO 13170
13120 FOR Y=A(X,1) TO A(X,2) STEP G5
13130 G4=G4+G6
13140 IF G4<=T(3,YR) PRINT@Y,"*";
13150 IF G4>T(3,YR) PRINT@Y,"#";
```



```
13160 GOTO 13090
13170 PRINT@13,"I-196";
13180 PRINT@30,"ASH";
13190 PRINT@46,"OAK";
13200 PRINT@128,"1ST";
13210 PRINT@384,"2ND";
13220 PRINT@640,"3RD";
13230 PRINT@704,"I-465";
13240 PRINT@896,"4TH";
13250 PRINT@242,"PRIMARIES";
13260 PRINT@311,"+=";G1;
13270 PRINT@375,"-=";44-G1;
13280 PRINT@498,"INTERSTATES";
13290 PRINT@567,"*=";T(3,YR);
13300 PRINT@631,"#=";16-T(3,YR);
13310 PRINT@960,"+,* = COMPLETE    -,# = INCOMPLETE";
13320 PRINT@1004,"PRESS ENTER";:INPUT Z
13330 CLS:PRINT TAB(12)"TRANSIT PERFORMANCE REVIEW FOR YEAR";YR:PRINT
13340 PRINT TAB(30)"YEAR";YR;TAB(40)"YEAR";YR-1;TAB(50)"PLAN"
13350 FOR X=1 TO 4
13351 IF X>1 THEN 13360
13352 PRINT S$(X);TAB(30);:PRINT USING FB;S(X,YR);:PRINT TAB(40);:PRINT USING FB
;S(X,YR-1);:PRINT TAB(50);:PRINT USING FB;S(X,11):GOTO 13370
13360 PRINT S$(X);TAB(30)S(X,YR);TAB(40)S(X,YR-1);TAB(50)S(X,11)
13370 NEXT X
13375 PRINT S$(5);TAB(30)S(5,YR);TAB(40)S(5,YR-1)
13380 PRINT:INPUT"PRESS ENTER";Z:CLS
14000 IF T(2,YR)<T(2,11) THEN 15100
14005 IF T(3,YR)<T(3,11) THEN 15100
14010 IF T(4,YR)>T(4,11) THEN 15100
14015 IF T(5,YR)>T(5,11) THEN 15100
14020 IF S(1,YR),S(1,11) THEN 15100
14025 IF S(2,YR)>S(2,11) THEN 15100
14030 IF S(3,YR)>S(3,11) THEN 15100
14035 IF S(4,YR)>S(4,11) THEN 15100
14040 CLS:PRINT CHR$(23):PRINT:PRINT"CONGRATULATIONS!"
14045 PRINT:PRINT"YOU HAVE SUCCESSFULLY COMPLETED":PRINT"THE TRANSPORTATION PLAN
 IN":PRINTYR;" YEARS.":PRINT
14050 IF YR>7 THEN 14060
14055 PRINT"YOUR PERFORMANCE HAS BEEN SO":PRINT"GOOD THAT YOU HAVE BEEN ASKED":P
RINT"TO BECOME THE NEW SECRETARY":PRINT"OF TRANSPORTATION.":GOTO 14200
14060 IF YR=10 THEN 14070
14065 PRINT"BECAUSE OF YOUR PERFORMANCE":PRINT"YOU HAVE BEEN ASKED TO BECOME":PR
INT"THE TRANSPORTATION DIRECTOR OF":PRINT"NEW YORK CITY.":GOTO 14200
14070 PRINT"YOU HAVE BEEN GIVEN A LARGE":PRINT"PAY RAISE AND HAVE BEEN ASKED":PR
INT"TO CONTINUE AS TRANSPORTATION":PRINT"DIRECTOR OF RIVER CITY."
14200 STOP:END
15100 CLS:IF YR=1 THEN 2990

15110 PRINT TAB(15)"PERFORMANCE EVALUATION FOR YEAR";YR:PRINT
15120 IF T(4,YR)<T(4,YR-1)+.1 OR T(4,YR-1)>T(4,YR-2)+.1 THEN 15130
15121 CV=CV-1:PRINT"STREET CONDITIONS HAVE WORSENED FOR TWO STRAIGHT YEARS.
15130 IF T(5,YR)<T(5,YR-1)+.1 OR T(5,YR-1)<T(5,YR-2)+.1 THEN 15140
15131 CV=CV-1:PRINT"TRAFFIC SAFETY HAS WORSENED TWO STRAIGHT YEARS."
15140 IF T(4,YR)<T(4,YR-1)*1.35 THEN 15150
15141 CV=CV-1:PRINT"STREET CONDITIONS HAVE WORSENED BY OVER 35 % THIS YEAR."
15150 IF T(5,YR)<T(5,YR-1)*1.4 THEN 15160
15151 CVICV-1:PRINT"TRAFFIC SAFETY HAS WORSENED BY OVER 40 % THIS YEAR."
15160 IF T(2,YR)>=T(2,YR-2)+5 THEN 15169
15161 CV=CV-1:PRINT"PRIMARY STREET CON;TRUCTION IS NOT PROGRESSING WELL."
15169 IF YR>7 OR YR=10  THEN 15180
15170 IF T(3,YR)>=(YR*2)-4 OR G1>=13+(YR*3) THEN 15180
15171 CV=CV-1:PRINT"THERE IS FEELING YOU WILL NOT COMPLETE THE STREET PLAN."
15180 IF T(4,YR)<T(4,YR-1) OR T(5,YR)<T(5,YR-1) OR T(2,YR)<T(2,YR-2)+4 THEN 1519
0B*USRM;CV=CV-1:PRINT"THERE IS GENERAL DISSATISFACTION WITH STREET FUND PERFORMA
NCE."
15181 CV=CV-1:PRINT"THERE IS GENERAL DISSATISFACTION WITH STREET FUND PERFORMANC
E."
15190 IF S(1,YR)<S(1,YR-1) OR S(1,YR-1)<S(1,YR-2) THEN 15200
15191 CV=CV-1:PRINT"BUS RIDERSHIP HAS DECLINED TWO STRAIGHT YEARS."
15200 IF S(2,YR)<S(2,0)+4 THEN 15210
15201 CV=CV-1:PRINT"THE BUS FLEET HATHEN BEEN ALLOWED TO DETERIORATE."
15210 IF S(3,YR)<S(3,YR-1)+.1 OR S(3,YR-1)<S(3,YR-2)+.1 THEN 15220
15211 CV=CV-1:PRINT"BUS DOWNTIME HAS INCRECSED TWO STRAIGHT YEARS."
15220 IF S(3,YR)>S(3,YR-1)*1.35 THEN 15230
15221 CV=CV-1:PRINT"DOWNTIME HAS INCREASED OVER 35 % THIS YEAR."
15230 IF "(4,YR)<S(4,YR-1)+.1 OR S(4,YR-1)<S(4,YR-2)+.1 THEN 15240
15231 CV=CV-1:PRINT"ON-SCHEDULE PERFORMANCE HAS DECLINED TWO STRAIGHT YEARS."
15177 IF S(4,YR)>S(4,YR-1)*1.35 THEN 15250
15241 CV=CV-1:PRINT"ON-SCHEDULE PERFORMANCE HAS DECLINED OVER 35 % THIS YEAR."
15250 IF S(1,YR)<S(1,YRH1) THEN X1=.3 ELSE X1=0
15260 FOR X=2 TO 4:IF S(X,YR)<S(X,YR-1) THEN X1=X1+.3:NEXT X
15270 IF X1>1 THEN 15280
15271 CV=CV-1:PRINT"THERE IS DISSATISFACTION WITH TRANSIT AUTHORITY PERFORMANCE.
"
```

```
15280 IF TB(2,YR)<7.5 OR RND(3)>2 THEN 15290
15281 CV=CV-2:PRINT"CITIZENS ARE UNHAPPY WITH THE HIGH TAX RAT)."
15290 IF T(2,YR)+T(3,YR)<T(2,YR-2)+T(3,YR-2)+11 OR T(3,YR)<T(3,YR-1)+2 THEN 1530
0
15291 CV=CV+1:PRINT"OVERALL STREET CONSTRUCTION IS PROGJESSING WELL."
15300 IF T(4,YR)>T(4,YR-1)-.1 OR T(4,YR-1)<T(4,YR-2)-.1 THEN 15310
15301 CV=CV+1:PRINT"STREET CONDITIONS HAVE IMPROVED TWO STRAIGHT YEARS."
15310 IF T(4,YR)>T(4,YR-1)*.65 THEN 15320
15311 CV=CV+1:PRINT"STREET CONDITIONS HAVE IMPROVED OVER 35 % THIS YEAR."
15320 IF T(5,BR)>T(5,YR-1)-.1 OR T(5,YR-1)<T(5,YR-2)-.1 THEN 15330
15321 CV=CV+1:PRINT"TRAFFIC SAFETY HAS IMPROVED TWO STRAIGHT YEARS."
15330 IF T(5,YR)<T 5,YR-1)*.6 THEN 15350
15331 CV=CV+1:PRINT"TRAFFIC SAFETY HAS IMPROVED OVER 40 % THIS YEAR."
15350 IF S(2,YR)>S(2,YR-1)-.1 OR S(2,YR-1)<S(2,YH-2)-.1 THEN 15360
15351 CV=CV+1:PRINT"BUS FLEET AGE HAS IMPROVED TWO STRAIGHT YEARS."
15360 IF S(3,YR)>S(3,YR-1)-.1 OR S(3,YR-1)<S(3,YR-2)-.
15361 CV=CV+1:PRINT"DOWNTIME HAS DECREASED TWO STRAIGHT YEARS."
15370 IF S(3,YR)<S(3,YR-1)*.65 THEN 15380
15371 CV=CV+1:PRINT"DOWNTIME H(S BEEN REDUCED OVER 35% THIS YEAR."
15380 IF S(4,YR)>S(4,YR-1)-.1 OR S(4,YR-1)<S(4,YR-2)-.1 THEN 15390
15381 CV=CV+1:PRINT"ON-SCHEDULE PERFO1MANCE HAS IMPROVED TWO STRAIGHT YEARS."
15390 IF S(4,YR)>S(4,YR-1)*.65 THEN 15400
15391 CV=CV+1:PRINT"ON-SCHEDULE PERFORMANCE HAS IMPROVED HVER 35% THIS YEAR."
15400 IF S(1,YR)>S(1,YR-1) THEN X1=.3 ELSE X1=0
15410 FOR X=2 TO 4:IF S(X,YR)<S(X,YR-1) THEN X1=X1+.3:NEXT X
15420 IF X1<1 THEN 15430COSERRTHEN=<CV=CV+1:PRINT"TRANSIT AUTHORITY PERFORMANCE
IS PROGRESSING WELL."
15421 CV=CV+1:PRINT"TRANSIT AUTHORITY PERFORMANCE IS PROGRESSING WELL."
15430 IF TB(2,YR)<3 OR RND(3)>2 THEN 15435
15431 CV=CV+1:PRINT"CITIZENS ARE HAPPY WITH THE LOW TAX RATE."
15435 IF YR=6 OR YR=9 THEN 15436 ELSE 15490
15436 IF B1=0 THEN 15490
15440 IF B2>T(4,YR) AND B2>0 THEN 15480
15450 IF B3>T(5 YR) AND B3>0 THEN 15480
15460 IF T(2,YR)<T(2,YR-3)-B4 AND B4>0 THEN 15480
15470 IF T(3,YR)<T(3,YR-3)-B5 AND B5>0 THEN 15490
15480 CV=CV-2:PRINT:PRINT"YOU HAVERENEGED ON YOUR BOND REFERENDUM PLEDGE."
15490 PRINT@960,"PRESS ENTER";:INPUT Z:CLS:IF CV>11 THEN CV=11
15491 IF CV<0 THEN CV=0
15492 IF CV<6 THEN 15494
15392 IF YR=10 THEN 15498
15494 PRINT:PRINT"AS A RESULT OF YOUR PERFORMANCE THIS YEAR THE CITY"
15495 IF CV>=6 PRINT"COMMISSION HAS VOTED";CV;"-";11-CV;"TO RETAIN YOU ANOTHER Y
EAR."
15496 IF CV<6 PRINT"COMMISSION HAS VOTED TO REQUEST YOUR RESIGNATION."
15497 IF CV>=6 THEN 15500 ELSE 15501
15498 PRINT*CHR$(23):PRINT:PRINT"YOU HAVE NOT SUCCESSFULLY":PRINT"COMPLETED THE
TEN YEAR PLAN.":PRINT:PRINT"HOWEVER, THE CITY COMMISSION":PRINT"HAS VOTED";CV;"-
";11-CGOTO;" TO RETAIN YOU":PRINT"AS ASSISTANT TRANSPORTATION":PRINT"DIRECTOR."
15499 GOTO 15501
15500 PRINT:INPUT"PRESS ENTER";Z:GOTO 2990
15501 PRINT:PRINT"ENTER 'RUN' IF YOU WISH TO":PRINT"TRY AGAIN.":STOP:END
```

Last One Loses

William E. Bailey

Many of you have probably played the ancient intellectual game of "Nim" and found it challenging. "Last One Loses" is a variant of "Nim" for the TRS-80 which provides a new and greater challenge.

The playing board in "Last One Loses" has three rows of circles: three in the top row, five in the middle row, and seven in the bottom row. This creates a board which looks like this:

```
    0 0 0

  0 0 0 0 0

0 0 0 0 0 0 0
```

The object is to cross off the circles in such a way that your opponent is forced to delete the last one. The players alternate crossing off as many circles as they wish so long as the circles are adjacent and in the same row. Each circle may be crossed off only once. For example, assume that player number one decides to cross off all three circles in the top row. The status of the game may be indicated as shown below:

William E. Bailey, 4601 N. Park Ave., Apt. 1811, Chevy Chase, MD 20015.

X X X

0 0 0 0 0

0 0 0 0 0 0 0

Notice that the circles that have been marked off are adjacent and in the same row.

If player number two then crosses off the three middle circles in the bottom row, we have:

X X X

0 0 0 0 0

0 0 X X X 0 0

Note that crossing off circles in the middle of rows is legal.

Then suppose player number one crosses off all the circles in the middle row:

0 0 X X X 0 0

X X X X X

X X X

And player number two counters by crossing off a circle in the bottom row:

0 0 X X X 0 X

X X X X X

X X X

Now player number one can win by crossing off the two circles on the right as follows:

X X X X X 0 X

X X X X X

X X X

Figure 1.

	#1	#2	#3	#4
A.	0 0 0	0 0 0	X X X	0 X 0
B.	0 0 0 0 0	X X X X X	0 0 0 X X	0 0 X X 0
C.	0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 X X 0 0

Since it is the second player's turn and there is only one circle remaining, he is forced to delete the last circle and player number one wins the game.

The strategy in "Last One Loses" is fairly simply but it takes a lot of trial and error to understand it thoroughly. Being a pro at the game is merely a matter of knowing which patterns of circles are winning patterns and which are not.

To understand patterns better, let us take the playing board and label each row as shown in Figure 1.

The pattern is simply the number of circles that are separated from the others by rows or by circles that have been marked off already. For example:

Diagram #1, the initial board has a pattern of [3,5,7], since row A has three circles, row B has five, and row C has seven. An element of a pattern is a single set of circles, so the pattern of diagram #1 has three elements. The first with three circles, the second with five, and the last with seven.

Diagram #2 has a pattern of [3,7] since row A has three circles and row C has seven circles. Row B has no circles and is considered null. Notice that not all patterns have the same number of elements since the pattern of diagram #2 only has two elements whereas the pattern of diagram #1 has three elements.

Diagram #3 also has a pattern of [3,7] since row A is null, row B has three circles and row C has seven. Note that even though diagrams #2 and #3 have different circles crossed off, they can be considered the same game because their patterns are identical.

Diagram #4 has two elements in each row, since they are all separated by circles that have been crossed off in the middle of the row. Row A has two elements each being a single circle, so thus far the pattern is [(1,1)]. Row B has one element with one circle and another with two. This adds to the pattern [(1,1), (2,1)]. (The parentheses indicate elements in different rows, while brackets surround the pattern.) And finally, row C has one element with three circles and another with two, so the pattern of diagram #4 is [(1,1), (2,1), (3,2)]. Now that we have the patterns put together with all of the elements, we can remove the parentheses to make this [1,1, 2,1,3,2] and for convenience, the numbers can be put in least to greatest order, so the pattern for diagram #4 in its final form is [1,1,1,2,2,3].

Now let us go back to the example game. On player number one's second move he crossed off the entire middle row which left this:

A X X X

B X X X X X

C 0 0 X X X 0 0
1 2 3 4

I have labeled the circles that we will be using with the numbers one through four. This has a pattern of [2,2], circles 1 and 2 make up the first element and circles 3 and 4 make up the second element. This is a winning pattern. By a winning pattern I mean that if a player can delete circles so as to make a pattern of [2,2] in any form then that player will be able to win the game. Player number one deleted row B and made a pattern of [2,2] and therefore will be able to win the game. This may not be immediately obvious so let us examine every possibility.

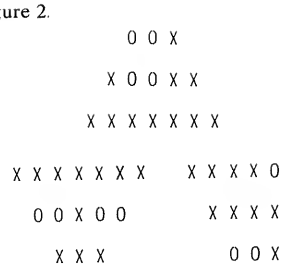
As happened in the example game, if player number two crosses out circle 1 then player number one can countermove by marking off circles 3 and 4 and only 2 will remain. Since it is the turn of player number two he loses the game and player number one wins.

If, instead, player number two marks off circles 1 and 2 then player number one can mark off 3 and only 4 remains, again player number one wins. And vice versa for circles 3 and 4. If player number two decides to cross off circle 4, then player number one can cross off 1 and 2; and if player number two decides to cross off circles 3 and 4, then player number one can cross off circle 1, leaving 2.

Thus, we have examined every possible move for player number two and discovered that no matter where he moves, player number one can always win. This means that the pattern of [2,2] in any form is a winning pattern. I include the "in any form" because there are many ways to fit a pattern of [2,2] on the playing board. Figure 2 shows a few of them:

As you can see, many other combinations are possible and all of them are

Figure 2.



winning patterns. The pattern [2,2] is not the only winning pattern. There are many others, all of which are included in the program.

When we were examining the pattern [2,2] to make sure it was a winning pattern, we discovered that player number one was able to win the game within two moves from the establishment of the pattern. Now let's start a new game in which player number one has just formed the winning pattern of [2,5,7].

It is obvious that player number 1 will not be able to win the game within the next two moves because the [2,5,7] pattern is so much bigger than the [2,2] pattern. In fact, it might take as many as 12 moves before the game is ended.

So how is player number one supposed to know what to cross off on all of his next moves? The answer is quite simple. Since [2,5,7] is a winning pattern, no matter where player number two moves, there will always be a move for player number one that will establish a new and smaller winning pattern. This continues on every move until player number one wins the game. For example, the pattern so far is [2,5,7] which will only fit the board in the way shown below:

```
A       X O O
B     O O O O O
C   O O O O O O O
```

Note that since [2,5,7] *is a winning pattern, this would be a good first move.*

It is player number two's move and let's say that he crosses out three circles in row B. The result is:

```
      X O O
    O O X X X
  O O O O O O O
```

Note that this is one of many possible moves for player number two.

Now there are several places that player number one can move to reestablish a winning pattern. One is to cross off the entire bottom row:

```
  X X X X X X X
    X X X O O
      O O X
```

```
10 DEFINT A-Z
20 DIM BD(3,7),DA(3,7),FA(9),BA(21),SA(25)
30 FOR I=1 TO 21
40    READ BA(I)
50 NEXT I
60 DATA 1,1,1,1,2,3,1,4,5,2,4,6,2,5,7,3,4,7,3,5,6
70 FOR J=1 TO 25
80    READ SA(J)
90 NEXT J
100 DATA 1,1,1,1,1,1,1,1,2,3,1,1,1,4,5,1,2,2,2,3,1,2,3,3,3
110 CLS
120 FOR I=1 TO 3
130    FOR J=1 TO 7
140       BD(I,J)=0
150    NEXT J
160 NEXT I
170 BD(1,1)=1:BD(1,2)=1:BD(1,6)=1:BD(1,7)=1:BD(2,1)=1:BD(2,7)=1
180 REM  "LAST ONE LOSES"
190 PRINT @ 12,CHR$(23)"WELCOME TO THE GAME"
200 PRINT @ 78,CHR$(23)"OF LAST-ONE LOSES"
210 FOR I=1 TO 1000:NEXT I
220 CLS
230 PRINT"DO YOU NEED TO SEE THE INSTRUCTIONS"
240 INPUT ST$
250 IF LEFT$(ST$,1)="Y" THEN GOSUB 1950
260 PRINT @ 0,"WHAT LEVEL OF DIFFICULTY DO YOU WANT(1-2-3-4)"
270 INPUT LE
280 LE=INT(LE)
290 IF LE>4 OR LE<1 THEN PRINT"INVALID":GOTO 260
300 PRINT @ 0,"DO YOU WANT TO GO FIRST                    "
310 INPUT QT$
320 IF LEFT$(QT$,1)="N" THEN GOTO 620
330 GOSUB 1740 :REM DISPLAY BOARD
340 REM PLAYER'S MOVE
350 PRINT @ 256,"DO YOU WANT TO KNOCK OUT MORE THAN ONE CIRCLE"
360 INPUT QZ$
370 IF LEFT$(QZ$,1)="Y" THEN GOTO 410
380 PRINT @ 384,"WHICH CIRCLE":INPUT MZ,NZ
390 IF BD(MZ,NZ)<>0 THEN PRINT"ILLEGAL":GOTO 380
400 BD(MZ,NZ)=2:GOTO 510
410 PRINT @ 384,"FROM":INPUT MZ,NZ
420 PRINT @ 512,"TO":INPUT ZI,ZJ
430 IF MZ<>ZI THEN PRINT"ILLEGAL": GOTO 410
440 IF NZ>ZJ THEN E=NZ:NZ=ZJ:ZJ=E
450 FOR I=NZ TO ZJ
460    IF BD(MZ,I)<>0 THEN PRINT"ILLECAL":GOTO 410
470 NEXT I
480 FOR J=NZ TO ZJ
490    BD(MZ,J)=2
500 NEXT J
510 GOSUB 1740 :REM DISPLAY BOARD
520 FOR OY=1 TO 7:PRINT:NEXT OY
530 GOSUB 1030 : REM FIND PATTERN
540 IF NOT(T=1 AND FA(1)=1) THEN GOTO 670
550 PRINT @ 256,"YOU WON....NICE JOB"
560 FOR D=1 TO 200:NEXT D
570 PRINT"BUT I'D LIKE TO SEE YOU DO IT AGAIN"
580 FOR D=1 TO 200:NEXT D
590 INPUT"CARE FOR ANOTHER GAME";QB$
600 IF LEFT$(QB$,1)="Y" THEN CLS:GOTO 110
610 END
620 REM COMPUTER'S MOVE
630 GOSUB 1740 :REM DISPLAY BOARD
640 IF LE<>4 THEN GOTO 740
650 PRINT @ 256,"THINKING"
660 GOTO 800
670 REM CHECK IF OPPONENT HAS WINNING PATTERN
680 IF T=0 THEN GOTO 1310
690 PRINT @ 256,"THINKING"
700 DV=LE:LE=4
710 GOSUB 1160 :REM IS PATTERN WINNING?
720 LE=DV
730 IF WN=0 THEN GOTO 800
740 REM RANDOMLY MOVE
750 I=RND(3):J=RND(7)
760 IF BD(I,J)<>0 THEN 750
770 BD(I,J)=2
780 GOSUB 1740 :REM DISPLAY BOARD
790 GOTO 340
800 REM PICK CIRCLES TO CROSS OFF
810 FOR I=1 TO 3
820    FOR J=1 TO 7
830       X=-1
840       IF BD(I,J)<>0 THEN GOTO 1000
850       X=X+1
```

Note that the [2,2] *winning pattern reappears.*

And again, no matter where player number two moves, player number one can countermove to form another winning pattern. This is the case with every winning pattern. Each can be made into a smaller winning pattern no matter where the opponent moves. Because of this, if your opponent has a winning pattern, all you can do is move randomly and hope that he makes a mistake.

There are four levels of difficulty in the program. Level 1 is the easiest and Level 4 is the most difficult to defeat. If the player plays Level 4 and allows the computer to move first (the player chooses who makes the first move) then the computer will always win. Indeed, if you watch the computer's moves it will make a winning pattern on each turn. But Levels 3, 2, and 1 are progressively less smart and the computer does not always make optimal moves.

The design of the program can be broken down into two major parts. The player's move can be found in the beginning of the program. It inputs the move, checks it for validity, and then displays the result on the board. The computer's move is much more complex. It has five subparts:

1. Check if opponent has a winning pattern.
2. Pick circles to cross off.
3. Find pattern.
4. Sort pattern.
5. Is pattern winning?

Note: These are all denoted in the program with REM statements.

The computer checks to see if the player has a winning pattern first for time conservation because, as I stated earlier, if the player has a winning pattern, all the computer can do is to hope that the player blows it. In the meantime, the computer moves randomly.

If the player does not have a winning pattern then the computer crosses off circles and checks to see if the pattern it has created is a winning one. It does this by figuring out the pattern through the "find pattern" sub-routine. Then it puts the elements of the pattern into least to greatest order through the "sort pattern" sub-routine and checks the pattern through the "Is pattern winning" sub-routine. When the computer finds a winning pattern it goes back to the player's move and the process continues until the game is over. □

```
860        PRINT @ 265,CHR$(143)
870        IF J+X=8 THEN GOTO 1000
880        IF BD(I,J+X)<>0 THEN GOTO 1000
890        FOR IJ=J TO J+X
900           BD(I,IJ)=2
910        NEXT IJ
920        GOSUB 1030 :REM FIND PATTERN
930        PRINT @ 265,CHR$(32)
940        GOSUB 1160 :REM IS IT WINNING?
950        IF WN=1 THEN GOSUB 1740 :GOTO 340
960        FOR IJ=J TO J+X
970          BD(I,IJ)=0
980        NEXT IJ
990        GOTO 850
1000     NEXT J
1010  NEXT I
1020  GOTO 740
1030  REM SUBROUTINE FIND PATTERN
1040  T=0:JK=0
1050  FOR M=1 TO 3
1060     FOR N=1 TO 7
1070        IF BD(M,N)=0 THEN JK=JK+1 ELSE GOTO 1130
1080        IF N=7 THEN GOTO 1100
1090        IF BD(M,N+1)=0 THEN GOTO 1130
1100        T=T+1
1110        PA(T)=JK
1120        JK=0
1130     NEXT N
1140  NEXT M
1150  RETURN
1160  REM SUBROUTINE IS IT WINNING?
1170  REM SORT PATTERN
1180  FOR F=1 TO T-1
1190     C=T-F+1
1200     FOR G=2 TO C
1210        IF PA(G)>=PA(G-1) THEN GOTO 1250
1220        KL=PA(G)
1230        PA(G)=PA(G-1)
1240        PA(G-1)=KL
1250     NEXT G
1260  NEXT F
1270  REM CHECK FOR WINNING PATTERN
1280  ON T GOTO 1290 ,1360 ,1390 ,1500 ,1580 ,1670 ,1690 ,1710 ,1730
1290  IF PA(1)<>1 THEN WN=0:RETURN
1300  GOSUB 1740 :REM DISPLAY BOARD
1310  PRINT @ 256,"I BEAT YOU THIS TIME!!"
1320  FOR FG=1 TO 200:NEXT FG
1330  INPUT"CARE FOR ANOTHER GAME";GH$
1340  IF LEFT$(GH$,1)="Y" THEN CLS:GOTO 110
1350  END
1360  IF PA(2)=1 THEN WN=0:RETURN
1370  IF PA(1)=PA(2) THEN WN=1:RETURN
1380  WN=0:RETURN
1390  LM=0:MN=0
1400  FOR K=LM+1 TO T+LM
1410     MN=MN+1
1420     IF PA(MN)=BA(K) THEN GOTO 1480
1430     IF K=19 OR K=20 OR K=21 THEN WN=0:RETURN
1440     IF LE=4 THEN GOTO 1460
1450     IF K=LE*3-2 OR K=LE*3-1 OR K=LE*3 THEN WN=0:RETURN
1460     LM=LM+T:MN=0
1470     GOTO 1400
1480  NEXT K
1490  WN=1:RETURN
1500  IF PA(4)=1 THEN WN=0:RETURN
1510  ON LE GOTO 1520 ,1530 ,1540 ,1560
1520  WN=0:RETURN
1530  IF PA(1)=1 AND PA(2)=1 AND PA(3)=PA(4) AND (PA(4)=2 OR
EN WN=1:RETURN ELSE WN=0:RETURN
1540  IF PA(1)=1 AND PA(2)=1 AND PA(3)=PA(4) THEN WN=1:RETURN
1550  WN=0:RETURN
1560  IF PA(1)=PA(2) AND PA(3)=PA(4) THEN WN=1:RETURN
1570  WN=0:RETURN
1580  LM=0:MN=0
1590  FOR K=LM+1 TO T+LM
1600     MN=MN+1
1610     IF PA(MN)=SA(K) THEN GOTO 1650
1620     IF K>=21 AND K<=25 THEN WN=0:RETURN
1630     LM=LM+T:MN=0
1640     GOTO 1590
1650  NEXT K
1660  WN=1:RETURN
1670  IF PA(6)=1 THEN WN=0:RETURN
1680  IF PA(1)=PA(2) AND PA(3)=PA(4) AND PA(5)=PA(6) THEN WN=1:
```

```
 WN=0:RETURN
1690 IF PA(7)=1 THEN WN=1:RETURN
1700 WN=0:RETURN
1710 IF PA(7)=2 THEN WN=1:RETURN
1720 WN=0:RETURN
1730 WN=1:RETURN
1740 REM SUBROUTINE DISPLAY BOARD
1750 FOR BZ=1 TO 3
1760    FOR BX=1 TO 7
1770       IF BD(BZ,BX)=0 THEN DA$(BZ,BX)="O"
1780       IF BD(BZ,BX)=1 THEN DA$(BZ,BX)=" "
1790       IF BD(BZ,BX)=2 THEN DA$(BZ,BX)=CHR$(143)
1800    NEXT BX
1810 NEXT BZ
1820 FOR FT=1 TO 7
1830    PRINT @ (FT-1)*2," ";DA$(1,FT);
1840 NEXT FT
1850 PRINT"                        1,3  1,4  1,5"
1860 FOR FT=1 TO 7
1870    PRINT @ (FT-1)*2+64," ";DA$(2,FT);
1880 NEXT FT
1890 PRINT"                2,2  2,3  2,4  2,5  2,6"
1900 FOR FT=1 TO 7
1910    PRINT @ (FT-1)*2+128," ";DA$(3,FT);
1920 NEXT FT
1930 PRINT"           3,1  3,2  3,3  3,4  3,5  3,6  3,7"
1940 RETURN
1950 CLS
1960 PRINT"    THE OBJECT OF THIS GAME IS TO CROSS OFF CIRCLES IN SUCH A"
1970 PRINT"WAY SO THAT YOUR OPPONENT IS FORCED TO DELETE THE LAST ONE."
1980 PRINT"PERTINENT RULES ARE:"
1990 PRINT
2000 PRINT"     1) YOU CAN CROSS OFF AS MANY CIRCLES AS YOU WISH;"
2010 PRINT"     PROVIDED THAT-"
2020 PRINT"        A)THEY ARE ADAJCENT AND IN THE SAME ROW."
2030 PRINT"        B)AND THE CIRCLES THAT YOU ARE CROSSING OFF HAVE NOT"
2040 PRINT"        BEEN MARKED OFF ALREADY."
2050 PRINT
2060 PRINT"    THE GAME IS ORGANIZED INTO FOUR DIFFERENT LEVELS OF PLAY,"
2070 PRINT"LABELED 1 TO 4.GAME 1 IS THE EASIEST AND GAME 4 IS THE HARDEST
2080 PRINT"THE COMPUTER AND THE PLAYER ALTERNATE IN CROSSING OFF CIRCLES,
2090 PRINT"AND REMEMBER-WHOEVER CROSSES OFF THE LAST-ONE LOSES!!"
2100 PRINT"      'HIT 'C' WHEN YOU ARE READY TO CONTINUE.'"
2110 IF INKEY$="C" THEN CLS:RETURN
2120 GOTO 2110
```

Perquackey

David E. Powers

Perquackey is a word game written in TRS-80 Disk BASIC for a 32K machine. The object of the game is to form as many words as possible from a set of random letters. Scoring depends on the number of letters in the words you form and several other factors which add an extra element of strategy to the game. See the instructions in the program for more details.

Before running the program, you should turn on the display of the realtime clock with the CLOCK command in DOS. Note that the sample runs were done on a Radio Shack line printer using NEWDOS to print the screen display. □

David E. Powers, 10 Wilber Ct., New Hyde Park, NY 11040.

IS THE CLOCK DISPLAYED? YES_ 00.06.47

00 07 22

P E R Q U A C K E Y

D O Y O U N E E D I N S T R U C T I O N S ?

Y E S _

PERQUACKEY 00 07 52

"PERQUACKEY" IS THE DIFFERENT WORD GAME, FUN FOR ALL, ESPECIALLY THOSE WHO LOVE TO HUNT FOR WORDS AND MEET THE CHALLENGE OF AN EVER-TICKING CLOCK

THIS VERSION OF "PERQUACKEY" MAY BE PLAYED BY UP TO FOUR PLAYERS. YOU CAN EVEN PLAY IT SOLITAIRE, ALWAYS TRYING TO BETTER YOUR SCORE FROM PREVIOUS GAMES AND ROUNDS. THE COMPUTER WILL SET UP YOUR GAMES, TALLY YOUR SCORES, AND EVEN MAKE SURE THAT YOU ARE PLAYING FAIRLY.

TO CONTINUE, PRESS ANY KEY

00 08 22

THE OBJECT OF THE GAME IS TO FIND AND SPELL AS MANY WORDS AS POSSIBLE FROM A LIST OF LETTERS THE COMPUTER WILL GENERATE FOR YOU. ALL IN A THREE-MINUTE TIME LIMIT. AT FIRST, THE COMPUTER WILL GIVE YOU TEN LETTERS WITH WHICH TO WORK. AS YOUR SCORE INCREASES AND YOU BECOME "VULNERABLE" YOU WILL BE ALLOTTED THIRTEEN LETTERS. BUT YOU WILL HAVE TO ACHIEVE BETTER SCORES OR BE SET POINTS FOR NON-SUPERIOR PLAY!

TO CONTINUE, PRESS ANY KEY

00 08 53

THE COMPUTER WILL PROMPT YOU AS YOU GO, IN CASE YOU SHOULD NEED ANY HELP IN THE MECHANICS OF THE GAME. BUT FOR YOUR INFORMATION YOU SHOULD KNOW IN ADVANCE THAT ONLY WORDS IN A STANDARD DICTIONARY ARE ACCEPTABLE. ALL THE PLAYERS SHOULD AGREE ON ONE BEFORE PLAY IS BEGUN. OF COURSE, LIKE MOST WORD GAMES, PROPER NAMES, FOREIGN WORDS, ABBREVIATIONS OR CAPITALIZED WORDS ARE NOT ALLOWED. ALSO, YOU MUST RESIST THE TEMPTATION TO USE PUNCTUATION MARKS. THE COMPUTER WILL NOT ALLOW THEM. THEY ARE NOT PART OF THE PERQUACKEY VOCABULARY!

00 09 21

YOU MAY NOT MAKE A WORD ENDING IN "S" IF THAT WORD ALSO APPEARS WITHOUT THE "S" DURING THE SAME TURN

ALL WORDS MUST BE AT LEAST THREE LETTERS LONG

YOU MAY NOT ENTER MORE THAN FIVE WORDS CONTAINING THE SAME NUMBER OF LETTERS IN ANY ONE TURN
TO ENTER A WORD, SIMPLY TYPE IT IN

OF COURSE, YOUR ERRORS CAN BE RECOVERED. TO DELETE THE LAST WORD YOU ENTERED, JUST ENTER ZZ. TO DELETE ANY OTHER WORD, TYPE ZZ FOLLOWED, WITHOUT A SPACE, BY THAT WORD (FOR EXAMPLE, ZZBIGBUG WOULD DELETE THE ENTRY "BIGBUG"
TO CONTINUE, PRESS ANY KEY

00 09 53

SCORING IS A LITTLE COMPLICATED, BUT THE COMPUTER HANDLES IT JUST FINE. YOU'LL GET ALL THE DETAILS RIGHT AWAY, BUT UP FRONT YOU SHOULD KNOW ABOUT THE BONUSES, BECAUSE THEY CAN REALLY ADD UP.

TO CONTINUE, PRESS ANY KEY

00 10 22

REMEMBER, YOU COULD ONLY ENTER FIVE WORDS OF EACH LENGTH? WELL, ONCE YOU DO ENTER FIVE WORDS IN EACH OF TWO ADJOINING CATEGORIES (FOR EXAMPLE FIVE THREE-LETTER WORDS AND FIVE FOUR-LETTER WORDS (AHEM)), YOU GET A RATHER FAT BONUS

300 POINTS FOR 5 THREES AND 5 FOURS
500 POINTS FOR 5 FOURS AND 5 FIVES
800 POINTS FOR 5 FIVES AND 5 SIXES
1200 POINTS FOR 5 SIXES AND 5 SEVENS
1850 POINTS FOR 5 SEVENS AND 5 EIGHTS
2700 POINTS FOR 5 EIGHTS AND 5 NINES

TO CONTINUE, PRESS ANY KEY

00 10 59

NOW, HERE'S THE COMPLICATED PART. SKIP IT IF YOU WISH, BUT IF YOU BECOME A REAL EXPERT, YOU'LL WANT THIS INFORMATION. SO HERE IT IS FOR YOU, ANYWAY

FOR THE FIRST THREE-LETTER WORD YOU GET 60 POINTS, AND 10 MORE FOR EACH THEREAFTER. 60, 70, 80, 90, 100 POINTS TOTAL FOR 1, 2, 3, 4 OR 5 THREE-LETTER WORDS

FOR THE FIRST FOUR-LETTER WORD YOU GET 120 POINTS, AND 20 MORE FOR EACH THEREAFTER. 120, 140, 160, 180, 200 POINTS FOR 1, 2, 3, 4 OR 5 FOUR-LETTER WORDS

TO CONTINUE, PRESS ANY KEY

00:11:28

AS THE FIVE-LETTER CATEGORY GROWS YOU GET 200, 250, 300, 350 AND 400 POINTS

SIX LETTER WORDS BRING 300, 400, 500, 600 OR 700 POINTS FOR ONE THROUGH FIVE ENTRIES

GET SEVEN LETTER WORDS AND YOU'LL WIPE OUT YOUR OPPONENTS. AS THAT CATEGORY FILLS YOU GET 500, 650, 800, 950 AND 1100 POINTS

BUT LOOK AT THE EIGHTS. 750, 1000, 1250, 1500, 1750 POINTS

TO CONTINUE, PRESS ANY KEY

00.11.58

NINE- AND TEN-LETTER WORDS ARE THE SUREST WAY TO DRIVE YOUR OPPONENTS TO DISTPACTION. NINES BRING 1000, 1500, 2000, 2500, OR 3000 POINTS FOR ONE TO FIVE ENTRIES

AND TENS ? ? ? - - - FORGET THE REST OF THE PLAYERS AND LOOK!

^^^^^ 1500, 3000, 5000, 7500 OR 13000 POINTS
FOR ONE THROUGH FIVE ENTRIES]]]]]

TO CONTINUE, PRESS ANY KEY

00.12.33

N O W S O M E V E R Y I M P O R T A N T D E T A I L S

D O N ' T S K I P T H E S E ! !

T O C O N T I N U E , P R E S S A N Y K E Y

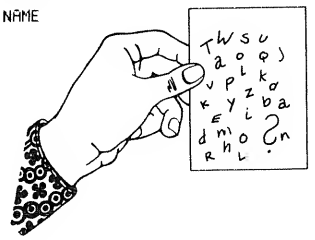
00:13:05

ONCE YOU HAVE ACCUMULATED 2000 POINTS YOU BECOME VULNERABLE! THAT'S FINE, BECAUSE THEN YOU'LL GET 13 LETTERS TO WORK WITH, BUT ALSO YOU MUST SCORE A MINIMUM OF 500 POINTS. IF YOU DON'T SCORE THE MINIMUM, 500 POINTS WILL BE DEDUCTED FROM YOUR SCORE, AND THE POINTS YOU DID MAKE IN THAT ROUND WILL BE DISALLOWED.

WHEN YOU ARE VULNERABLE, YOU MAY NOT MAKE THREE-LETTER WORDS.

```
THE GAME IS OVER AT THE END OF THE ROUND IN WHICH ANY PLAYER
REACHES A TOTAL OF 5000 POINTS

TO CONTINUE, PRESS ANY KEY

                                                    00 13 33
A WORD ABOUT THE DISPLAY

THE DISPLAY IS SELF-PROMPTING AND WILL HELP YOU A LOT
IT IS ALSO SELF-EXPLANATORY, LISTING YOUR WORDS BY LENGTH

WORDS ENTERED AFTER THREE MINUTES WILL AUTOMATICALLY BE
DISALLOWED, AND THE TURN WILL BE ENDED.  TO END YOUR TURN
BEFORE THE TIME LIMIT EXPIRES, ENTER XX

WHEN YOUR TURN IS ENDED, THE COMPUTER WILL EXAMINE ALL THE
ENTRIES AND DISALLOW WORDS MADE BY ADDING S TO OTHER ENTRIES,
DUPLICATE ENTRIES AND WORDS INCONSISTENT WITH THE LETTER LIST

TO CONTINUE, PRESS ANY KEY

                                                    00 14 08
DISALLOWED WORDS WILL BE BRACKETED AS IN THE FOLLOWING EXAMPLES

WORDS MADE BY ADDING S          ++EXAMPLES++S
DUPLICATE WORDS:                ++EXAMPLE++2
WORDS INCONSISTENT WITH LETTERS: ++EXAMPLE++?
```

```
TO CONTINUE, PRESS ANY KEY

                                                    00 14 36
AFTER THE COMPUTER DISALLOWS WORDS, YOUR OPPONENTS MAY DO SO,
TOO.  THEY MAY CHECK WORDS IN A STANDARD DICTIONARY AND THEN
ENTER ANY CHALLENGES WHICH THE COMPUTER WILL BRACKET WITH
++  ++C

AFTER ALL CHALLENGES ARE MADE, ENTER XX, AND THE COMPUTER WILL
CALCULATE AND DISPLAY YOUR SCORE AND THEN DISPLAY A SCOREBOARD
FOR ALL PLAYERS

DURING PLAY, THE LOWER RIGHT CORNER OF THE SCREEN WILL SHOW THE
PLAYER'S SCORE UP TO THE END OF HIS LAST TURN.  THE UPPER RIGHT
WILL DISPLAY THE TIMER

TO CONTINUE, PRESS ANY KEY

                                                    00:15 13
YOU CAN PROBABLY COME UP WITH ALL SORTS OF REFINEMENTS TO THE
BASIC GAME.  WHAT WONDERFUL DEVELOPMENTS THEY COULD BE! LIKE
THEME GAMES.  MAYBE DEVOTE ONE WHOLE GAME ONLY TO COMPUTER-
SCIENCE WORDS.

OR SCI FI
          OR WHO KNOWS
                    WHERE YOUR
                              IMAGINATION
                                        WILL LEAD?


TO CONTINUE, PRESS ANY KEY.

                                                    00:16:17
NOW, IF YOU'D LIKE TO REVIEW THAT, JUST KEY AN R

BUT IF YOU'RE READY TO PLAY, KEY ANYTHING ELSE!

HOW MANY PLAYERS FOR PERQUACKEY (1-4)? 2            00:17:03
TELL ME PLAYER 1 'S NAME
==> STEVE
TELL ME PLAYER 2 'S NAME
==> MR. BILL_
```

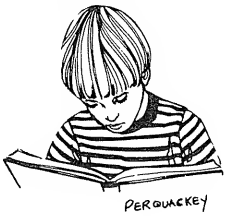


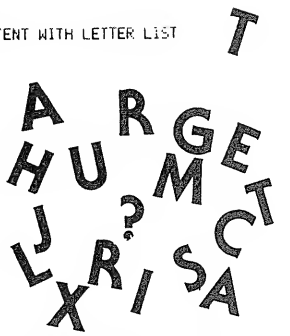
```
                STEVE PLAYING AND NOT VULNERABLE        00:00:02

        YOUR LETTERS ARE   P N S B X O K U E A
  3          4          5          6          7




  8          9          10


                                                            0
                STEVE PLAYING AND NOT VULNERABLE        00:02 44

        YOUR LETTERS ARE   P N S B X O K U E A
  3          4          5          6          7
 BOX        BASK                  SPOKEN
 POX        BONE
 SUN        PUNK
 PUN        SUNK
            BAKE
    8          9          10


                                                            0
                  * *  OVERTIME  * *                    00 03 34
ENTER CHALLENGES, THEN XX _

        YOUR LETTERS ARE   P N S B X O K U E A
  3          4          5          6          7
 BOX        BASK                  SPOKEN
 POX        BONE
 SUN        PUNK
 PUN        SUNK
            BAKE
    8          9          10


                                                            0
                  * *  OVERTIME  * *                    00:04 08
            SCORE FOR STEVE FOR THIS ROUND  590

        YOUR LETTERS ARE   P N S B X O K U E A
  3          4          5          6          7
 BOX        BASK                  SPOKEN
 POX        BONE
 SUN        PUNK
 PUN        SUNK
            BAKE
    8          9          10


                                                            0
                                                        00 04 42
                  TOTAL SCORE, ROUND 1

PLAYER          LAST SCORE      TOTAL SCORE     VULNERABLE?
STEVE              590             590             NO
MR. BILL             0               0             NO

TO CONTINUE, PRESS ANY KEY
```

```
10 REM *** PERQUACKEY, VERSION 2.2  -- 14 MAY 1979
20 REM *** BASED ON "PERQUACKEY"  (C) HOLLINGSWORTH BROS.  1956
30 REM *** AND ON "PERQUACKEY, THE DIFFERENT WORD GAME"
   (C) LEISURE DYNAMICS, INC., 1970
40 REM *** PUBLISHED BY LAKESIDE INDUSTRIES, A DIVISION OF
   LEISURE DYNAMICS, INC., MINNEAPOLIS, MINN
50 REM *** PROGRAM BY DAVID E. POWERS
60 REM ***              10 WILBEN CT
70 REM ***              NEW HYDE PARK, NY 11040
80 REM ***              516 437 8220
90 POKE &H40A9,&HFF
```

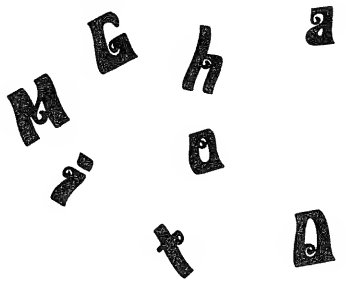
```
100 CLEAR 1000
110 XX$=STRING$(64," ")
120 DEFINT B,E,F,I,L-N,P,R,S,V,W
130 CLS  INPUT "IS THE CLOCK DISPLAYED";A$
140 IF LEFT$(A$,1)="Y" THEN 170
150 PRINT: PRINT "RETURN TO DOS AND ENTER CLOCK COMMAND "
160 CMD"S"
170 CLS
180 PRINT CHR$(23)
190 PRINT @ 274, "PERQUACKEY"
200 PRINT @ 708, "DO YOU NEED INSTRUCTIONS?"
210 PRINT
220 LINEINPUT A$
230 IF LEFT$(A$,1)="N" THEN 940
240 CLS
250 PRINT TAB(28) "PERQUACKEY"
260 PRINT
    PRINT CHR$(34) "PERQUACKEY" CHR$(34) " IS THE DIFFERENT WORD GAME,
FUN FOR ALL, ESPECIALLY THOSE WHO LOVE TO HUNT FOR WORDS AND MEET THE
CHALLENGE OF AN EVER-TICKING CLOCK "
270 PRINT
280 PRINT "THIS VERSION OF " CHR$(34) "PERQUACKEY" CHR$(34) " MAY BE
PLAYED BY UP TO FOUR PLAYERS  YOU CAN EVEN PLAY IT SOLITAIRE, ALWAYS TRYING TO
BETTER YOUR SCORE FROM PREVIOUS GAMES AND ROUNDS  THE COMPUTER"
290 PRINT "WILL SET UP YOUR GAMES, TALLY YOUR SCORES, AND EVEN
MAKE SURE THAT YOU ARE PLAYING FAIRLY "
300 GOSUB 4050
310 PRINT:PRINT"THE OBJECT OF THE GAME IS TO FIND AND SPELL AS MANY WORDS AS
POSSIBLE FROM A LIST OF LETTERS THE COMPUTER WILL GENERATE FOR
YOU, ALL IN A THREE-MINUTE TIME LIMIT.  AT FIRST, THE COMPUTER"
320 PRINT "WILL GIVE YOU TEN LETTERS WITH WHICH TO WORK.  AS YOUR SCORE"
330 PRINT "INCREASES AND YOU BECOME "CHR$(34) "VULNERABLE" CHR$(34) " YOU WILL
BE ALLOTTED THIRTEEN LETTERS.  BUT YOU WILL HAVE TO ACHIEVE BETTER SCORES
OR BE SET POINTS FOR NON-SUPERIOR PLAY!"
340 GOSUB 4050
350 PRINT "THE COMPUTER WILL PROMPT YOU AS YOU GO, IN CASE YOU SHOULD NEED
ANY HELP IN THE MECHANICS OF THE GAME  BUT FOR YOUR INFORMATION
YOU SHOULD KNOW IN ADVANCE THAT ONLY WORDS IN A STANDARD"
360 PRINT "DICTIONARY ARE ACCEPTABLE.  ALL THE PLAYERS SHOULD AGREE
ON ONE BEFORE PLAY IS BEGUN.  OF COURSE, LIKE MOST WORD GAMES,
PROPER NAMES, FOREIGN WORDS, ABBREVIATIONS OR CAPITALIZED WORDS"
370 PRINT "ARE NOT ALLOWED.  ALSO, YOU MUST RESIST THE TEMPTATION TO USE
PUNCTUATION MARKS.  THE COMPUTER WILL NOT ALLOW THEM.  THEY ARE
NOT PART OF THE PERQUACKEY VOCABULARY!"
380 GOSUB 4050
390 PRINT "YOU MAY NOT MAKE A WORD ENDING IN " CHR$(34) "S" CHR$(34) " IF THAT
WORD ALSO APPEARS WITHOUT THE " CHR$(34) "S" CHR$(34) " DURING THE SAME TURN "
400 PRINT
410 PRINT "ALL WORDS MUST BE AT LEAST THREE LETTERS LONG
YOU MAY NOT ENTER MORE THAN FIVE WORDS CONTAINING THE SAME
NUMBER OF LETTERS IN ANY ONE TURN
TO ENTER A WORD, SIMPLY TYPE IT IN
OF COURSE, YOUR ERRORS CAN BE RECOVERED.  TO DELETE THE"
420 PRINT "LAST WORD YOU ENTERED, JUST ENTER ZZ.  TO DELETE ANY OTHER
WORD, TYPE ZZ FOLLOWED, WITHOUT A SPACE, BY THAT WORD (FOR
EXAMPLE, ZZBIGBUG WOULD DELETE THE ENTRY " CHR$(34) "BIGBUG" CHR$(34)" "
430 GOSUB 4050
440 PRINT
450 PRINT "SCORING IS A LITTLE COMPLICATED, BUT THE COMPUTER HANDLES
IT JUST FINE.  YOU'LL GET ALL THE DETAILS RIGHT AWAY, BUT UP
FRONT YOU SHOULD KNOW ABOUT THE BONUSES, BECAUSE THEY CAN
REALLY ADD UP
"
460 GOSUB 4050
470 PRINT "REMEMBER, YOU COULD ONLY ENTER FIVE WORDS OF EACH LENGTH?
WELL, ONCE YOU DO ENTER FIVE WORDS IN EACH OF TWO ADJOINING
CATEGORIES (FOR EXAMPLE FIVE THREE-LETTER WORDS AND FIVE"
480 PRINT "FOUR-LETTER WORDS (AHEM!)), YOU GET A RATHER FAT BONUS.

              300 POINTS FOR 5 THREES AND 5 FOURS
              500 POINTS FOR 5 FOURS AND 5 FIVES
              800 POINTS FOR 5 FIVES AND 5 SIXES"
490 PRINT"            1200 POINTS FOR 5 SIXES AND 5 SEVENS
            1850 POINTS FOR 5 SEVENS AND 5 EIGHTS
            2700 POINTS FOR 5 EIGHTS AND 5 NINES"
500 GOSUB 4050
510 PRINT "NOW, HERE'S THE COMPLICATED PART.  SKIP IT IF YOU WISH, BUT IF
YOU BECOME A REAL EXPERT, YOU'LL WANT THIS INFORMATION, SO HERE
IT IS FOR YOU, ANYWAY "
520 PRINT "
FOR THE FIRST THREE-LETTER WORD YOU GET 60 POINTS, AND 10 MORE
FOR EACH THEREAFTER  60, 70, 80, 90, 100 POINTS TOTAL FOR 1,
2, 3, 4 OR 5 THREE-LETTER WORDS "
530 PRINT "
FOR THE FIRST FOUR-LETTER WORD YOU GET 120 POINTS, AND 20 MORE
FOR EACH THEREAFTER  120, 140, 160, 180, 200 POINTS FOR 1, 2,
3, 4 OR 5 FOUR-LETTER WORDS "
540 GOSUB 4050
550 PRINT "AS THE FIVE-LETTER CATEGORY GROWS YOU GET 200, 250, 300,
350 AND 400 POINTS "
560 PRINT: PRINT"SIX LETTER WORDS BRING 300, 400, 500, 600 OR 700 POINTS
FOR ONE THROUGH FIVE ENTRIES.
GET SEVEN LETTER WORDS AND YOU'LL WIPE OUT YOUR OPPONENTS.  AS
570 PRINT "THAT CATEGORY FILLS YOU GET 500, 650, 800, 950 AND 1100 POINTS.
BUT LOOK AT THE EIGHTS: 750, 1000, 1250, 1500, 1750 POINTS."
580 GOSUB 4050
590 PRINT "NINE- AND TEN-LETTER WORDS ARE THE SUREST WAY TO DRIVE YOUR
OPPONENTS TO DISTRACTION.  NINES BRING 1000, 1500, 2000, 2500,
OR 3000 POINTS FOR ONE TO FIVE ENTRIES "
600 PRINT: PRINT "AND TENS ? ? ? - - - FORGET THE REST OF THE PLAYERS AND LOOK!"
610 PRINT:PRINT TAB(10) STRING$(5,CHR$(94))" 1500, 3000, 5000, 7500 OR 13000 POINTS"
620 PRINT TAB(15) "FOR ONE THROUGH FIVE ENTRIES "
    STRING$(5,CHR$(93))
630 GOSUB 4050
640 PRINT CHR$(2)
650 PRINT: PRINT"NOW SOME VERY IMPORTANT DETAILS
    DON'T SKIP THESE ! !"
660 XX$=STRING$(32," ")
670 GOSUB 4050
680 XX$=STRING$(64," ")
690 PRINT: PRINT "ONCE YOU HAVE ACCUMULATED 2000 POINTS YOU BECOME VULNERABLE!
THAT'S FINE, BECAUSE THEN YOU'LL GET 13 LETTERS TO WORK WITH,
BUT ALSO YOU MUST SCORE A MINIMUM OF 500 POINTS  IF YOU DON'T"
700 PRINT "SCORE THE MINIMUM, 500 POINTS WILL BE DEDUCTED FROM YOUR SCORE,
AND THE POINTS YOU DID MAKE IN THAT ROUND WILL BE DISALLOWED "
710 PRINT: PRINT "WHEN YOU ARE VULNERABLE, YOU MAY NOT MAKE THREE-LETTER WORDS "
720 PRINT"
THE GAME IS OVER AT THE END OF THE ROUND IN WHICH ANY PLAYER
REACHES A TOTAL OF 5000 POINTS "
730 GOSUB 4050
740 PRINT "A WORD ABOUT THE DISPLAY      "
750 PRINT: PRINT"THE DISPLAY IS SELF-PROMPTING AND WILL HELP YOU A LOT
IT IS ALSO SELF-EXPLANATORY, LISTING YOUR WORDS BY LENGTH
WORDS ENTERED AFTER THREE MINUTES WILL AUTOMATICALLY BE"
760 PRINT "DISALLOWED, AND THE TURN WILL BE ENDED.  TO END YOUR TURN
BEFORE THE TIME LIMIT EXPIRES, ENTER XX
WHEN YOUR TURN IS ENDED, THE COMPUTER WILL EXAMINE ALL THE"
770 PRINT"ENTRIES AND DISALLOW WORDS MADE BY ADDING S TO OTHER ENTRIES,
DUPLICATE ENTRIES AND WORDS INCONSISTENT WITH THE LETTER LIST "
780 GOSUB 4050
790 PRINT "DISALLOWED WORDS WILL BE BRACKETED AS IN THE FOLLOWING EXAMPLES "
800 PRINT"
WORDS MADE BY ADDING S.          ++EXAMPLES++S
DUPLICATE WORDS                  ++EXAMPLE++2
WORDS INCONSISTENT WITH LETTERS. ++EXAMPLE++?"
810 GOSUB 4050
820 PRINT "AFTER THE COMPUTER DISALLOWS WORDS, YOUR OPPONENTS MAY DO SO,
TOO.  THEY MAY CHECK WORDS IN A STANDARD DICTIONARY AND THEN
ENTER ANY CHALLENGES WHICH THE COMPUTER WILL BRACKET WITH
++  ++C "
830 PRINT"
AFTER ALL CHALLENGES ARE MADE, ENTER XX, AND THE COMPUTER WILL
CALCULATE AND DISPLAY YOUR SCORE AND THEN DISPLAY A SCOREBOARD
FOR ALL PLAYERS "
840 PRINT"
DURING PLAY, THE LOWER RIGHT CORNER OF THE SCREEN WILL SHOW THE
PLAYER'S SCORE UP TO THE END OF HIS LAST TURN.  THE UPPER RIGHT
WILL DISPLAY THE TIMER."
850 GOSUB 4050
860 PRINT"YOU CAN PROBABLY COME UP WITH ALL SORTS OF REFINEMENTS TO THE
BASIC GAME.  WHAT WONDERFUL DEVELOPMENTS THEY COULD BE! LIKE
THEME GAMES.  MAYBE DEVOTE ONE WHOLE GAME ONLY TO COMPUTER-
SCIENCE WORDS."
870 PRINT"
OR SCI FI
        OR WHO KNOWS
                WHERE YOUR
                        IMAGINATION"
880 PRINT "                                                WILL LEAD?"
890 GOSUB 4050
900 PRINT "NOW, IF YOU'D LIKE TO REVIEW THAT, JUST KEY IN R
BUT IF YOU'RE READY TO PLAY, KEY ANYTHING ELSE! "
910 A$=INKEY$  IF A$="" THEN 910
920 IF A$="R" THEN 240
930 A$=""
940 RANDOM
950 DEF FNA$(A$)=MID$(A$, RND(6),1)+" "
960 DATA FUNIPT, LTORDN, MUSRIG, BYWOLO, VEJQZX, WOPOMC, BRHIKT,
       SRHIFU, AAAEEE, VSYQWS, FHLPBN, JGDKCM
970 READ C1$, C2$, C3$, C4$, C5$, C6$, C7$, C8$, C9$, VA$, VB$,
    VC$
980 CLS
990 INPUT "HOW MANY PLAYERS FOR PERQUACKEY (1-4)";N
1000 IF N<1 OR N>4 THEN 990
1010 DIM W$(7,5), WT(7), WC(7), WD(7), F(7,5), I$(N), V(N), S(),      S1(N)
1020 FOR LP=1 TO N
```

```
1030   PRINT "TELL ME PLAYER" LP "'S NAME
1040 LINEINPUT "==> "; I$(LP)
1050 NEXT
1060 P=1
1070 R=1
1080 CLS
1090 PRINT @ 15, I$(P) " PLAYING AND"
1100 IF V(P)=1 THEN PRINT " VULNERABLE : ELSE PRIN  " N : VULNERABLE";
1110 PRINT @ 1016, S(P);
1120 IF V(P)=1 PRINT @ 323, "NO 3'S"
1130 PRINT @ 261, "  3 ";
1140 PRINT @ 272, "  4 ";
1150 PRINT @ 283, "  5 ";
1160 PRINT @ 296, "  6 ";
1170 PRINT @ 309, "  7 ";
1180 PRINT @ 648, "  8 ";
1190 PRINT @ 663, "  9 ";
1200 PRINT @ 679, " 10 ";
1210 IF V(P)=0 THEN 1260  REM CHECK FOR VULNERABILITY
1220 V1$=FNA$(VA$)
1230 V2$=FNA$(VB$)
1240 V3$=FNA$(VC$)
1250 LY$=V1$+V2$+V3$
1260 L1$=FNA$(C1$)
1270 L2$=FNA$(C2$)
1280 L3$=FNA$(C3$)
1290 L4$=FNA$(C4$)
1300 L5$=FNA$(C5$)
1310 L6$=FNA$(C6$)
1320 L7$=FNA$(C7$)
1330 L8$=FNA$(C8$)
1340 L9$=FNA$(C9$)
1350 L0$=FNA$(C9$)
1360 LY$=LY$+L1$+L2$+L3$+L4$+L5$+L6$+L7$+L8$+L9$+L0$
1370 PRINT @ 128, "PRESS ANY KEY WHEN READY ";
1380 FOR LP=0 TO 95  A$=INKEY$  IF A$<>"" THEN 1420ELSE NEXT
1390 PRINT @ 128, XX$;
1400 FOR LP=0 TO 50  A$=INKEY$  IF A$<>"" THEN 1420ELSE NEXT
1410 GOTO 1370
1420 PRINT @ 128, XX$;
1430 POKE &H4041,0
1440 POKE &H4042,0
1450 POKE &H4043,0
1460 PRINT @ 202, "YOUR LETTERS ARE   " LY$;
1470 CMD"R"
1480 PRINT @ 90,"";
1490 LINEINPUT A$
1500 PRINT @ 64, XX$;
1510 IF PEEK(&H4043)>0 OR PEEK(&H4042)>3 OR (PEEK(&H4042)=3
    AND PEEK(&H4041)>0) THEN PRINT @ 0, XX$; ELSE 1540
1520 PRINT @ 22, "* *  OVERTIME  ** *";
1530 GOTO 1760
1540 IF LEFT$(A$,2)="XX" THEN 1760
1550 IF LEFT$(A$,2)="ZZ" AND LEN(A$)=2 THEN 3320
1560 IF LEFT$(A$,2)="ZZ" THEN 3340
1570 A1$=A$
1580 L=LEN(A$)-3
1590 IF V(P)=1 AND L<1 THEN 1480
1600 IF L<0 THEN 1480
1610 IF L>7 THEN L=7
1620 W$(L,WC(L))=A$
1630 WC(L)=WC(L)+1
1640 IF WC(L)>5 THEN PRINT @ 64, "CATEGORY FULL  DISALLOWED " ELSE GOTO 1670
1650 WC(L)=WC(L)-1
1660 GOTO 1480
1670 B=64*(WC(L)-1)
1680 IF L=0 PRINT @ 325+B,A$;   GOTO 1480
1690 IF L=1 PRINT @ 335+B,A$;   GOTO 1480
1700 IF L=2 PRINT @ 346+B,A$;   GOTO 1480
1710 IF L=3 PRINT @ 358+B,A$;   GOTO 1480
1720 IF L=4 PRINT @ 371+B,A$;   GOTO 1480
1730 IF L=5 PRINT @ 709+B,A$;   GOTO 1480
1740 IF L=6 PRINT @ 724+B,A$;   GOTO 1480
1750 PRINT @ 740+B,A$;  GOTO 1480
1760 PRINT @ 64, XX$;
1770 PRINT @ 86, "INSPECTING ENTRIES";
1780 FOR LP=0 TO 7
1790   WD(LP)=WC(LP)
1800   IF WC(LP)>4 THEN WC(LP)=4
1810 NEXT
1820 REM CHECKS FOR WORDS MADE BY ADDING S
1830 PP$="++"   PQ$="++S"
1840 FOR LP=1 TO 7
1850   GOSUB 4030
1860   FOR LQ=0 TO WC(LP)
1870     IF RIGHT$(W$(LP,LQ),1)<>"S" THEN 1930
1880     FOR LR=0 TO WC(LP-1)
1890       IF W$(LP-1,LR)<>LEFT$(W$(LP,LQ),2+LP) THEN 1920
1900       F(LP,LQ)=1
1910       ON LP GOSUB 3860, 3880, 3900, 3920, 3940, 3960,          3980
1920     NEXT LR
1930   NEXT LQ
1940   GOSUB 4040
1950 NEXT LP
1960 REM ELIMINATES WORDS INCONSISTENT WITH LETTER LIST
1970 PP$="++"   PQ$="++?"
1980 FOR LP=0 TO 7
1990   GOSUB 4030
2000 LX$=LY$
2010   FOR LQ=0 TO WC(LP)
2020     IF W$(LP,LQ)="" THEN 2160
2030     IF F(LP,LQ)=1 THEN 2140
2040     LX$=LY$
2050     FOR LR=1 TO LP+3
2060       T$=MID$(W$(LP,LQ),LR,1)
2070       IN=INSTR(LX$,T$)
2080       IF IN=0 THEN 2110
2090       MID$(LX$,IN,1)=" "
2100       GOTO 2130
2110       F(LP,LQ)=1
2120       ON LP+1 GOSUB 3840, 3860, 3880, 3900, 3920, 3940,  3960, 3980
2130     NEXT LR
2140   NEXT LQ
2150   GOSUB 4040
2160 NEXT LP
2170 REM ELIMINATES DUPLICATE ENTRIES
2180 PP$="++"   PQ$="++2"
2190 FOR LP=0 TO 7
2200   GOSUB 4030
2210   FOR LQ=0 TO WC(LP)
2220   IF W$(LP,LQ)="" THEN 2320
2230     FOR LR=0 TO WC(LP)
2240       IF LQ<>LR AND W$(LP,LQ)=W$(LP,LR) THEN 2270
2250     NEXT LR
2260     GOTO 2300

2270     IF F(LP,LQ)=1 THEN 2300
2280      F(LP,LR)=1
2290     ON LP+1 GOSUB 3840, 3860, 3880, 3900, 3920, 3940,          3960, 3980
2300   NEXT LQ
2310   GOSUB 4040
2320 NEXT LP
2330 A$=""
2340 REM ELIMINATES CHALLENGED ENTRIES
2350 PP$="++"  PQ$="++C"
2360 A$=""
2370 PRINT @ 64, XX$;
2380 PRINT @ 64, "ENTER CHALLENGES, THEN XX";
2390 PRINT @ 90, "";
2400 LINEINPUT A$
2410 PRINT @ 64, XX$;
2420 PRINT @ 99, "";
2430 IF A$="XX"THEN 2600
2440 IF LEN(A$)<3 THEN 2400
2450 L=LEN(A$)-3
2460 FOR LP=0 TO WC(L)
2470   IF A$=W$(L,LP) THEN 2510
2480 NEXT
2490 PRINT @ 64, "CHALLENGED WORD NOT FOUND              "
2500 GOTO 2390
2510 IF F(L,LP)=1 THEN 2580
2520 F(L,LP)=1
2530 LQ=LP
2540 LP=L
2550 ON L+1 GOSUB 3840, 3860, 3880, 3900, 3920, 3940, 3960,          3980
2560 PRINT @ 64,XX$;
2570 GOTO 2390
2580 PRINT @ 64, "ENTRY ALREADY DISALLOWED  "
2590 GOTO 2390
2600 PRINT @ 64,XX$;
2610 FOR LP=0 TO 7
2620   WT(LP)=WT(LP)-WD(LP)
2630 NEXT
2640 REM SCORING
2650 IF WT(0)>0 THEN S=50+10*WT(0)
2660 IF WT(1)>0 THEN S=S+100+20*WT(1)
2670 IF WT(2)>0 THEN S=S+150+50*WT(2)
2680 IF WT(3)>0 THEN S=S+200+100*WT(3)
2690 IF WT(4)>0 THEN S=S+350+150*WT(4)
2700 IF WT(5)>0 THEN S=S+500+250*WT(5)
2710 IF WT(6)>0 THEN S=S+500+500*WT(6)
2720 IF WT(7)=1 THEN S=S+1500
2730 IF WT(7)=2 THEN S=S+3000
2740 IF WT(7)=3 THEN S=S+5000
2750 IF WT(7)=4 THEN S=S+7500
2760 IF WT(7)=5 THEN S=S+13000
2770 IF WT(0)=5 AND WT(1)=5 THEN S=S+300
2780 IF WT(1)=5 AND WT(2)=5 THEN S=S+500
2790 IF WT(2)=5 AND WT(3)=5 THEN S=S+800
```

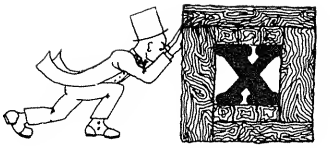


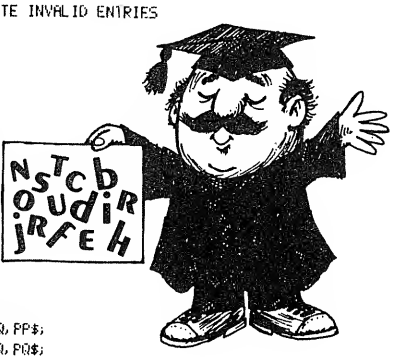


```
2800 IF WT(3)=5 AND WT(4)=5 THEN S=S+1200
2810 IF WT(4)=5 AND WT(5)=5 THEN S=S+1850
2820 IF WT(5)=5 AND WT(6)=5 THEN S=S+2700
2830 IF V(P)=1 AND S<500 THEN S=-500
2840 PRINT @ 79, "SCORE FOR " I$(P) " FOR THIS ROUND." S
2850 S1(P)=S
2860 S(P)=S(P)+S
2870 FOR LP=0 TO 1500 : NEXT
2880 CLS
2890 PRINT
2900 PRINT TAB(20) "TOTAL SCORE, ROUND" R
2910 PRINT
2920 PRINT "PLAYER", "LAST SCORE", "TOTAL SCORE", "VULNERABLE?"
2930 FOR LP=1 TO N
2940   PRINT I$(LP),
2950   PRINT USING "  #####"; S1(LP),
2960   PRINT USING "            #####"; S(LP),
2970   IF S(LP)>=2000 THEN PRINT "              YES" ELSE PRINT "       NO"
2980   IF S(LP)>=2000 THEN V(LP)=1 ELSE V(LP)=0
2990   IF S(LP)>=5000 THEN E=1
3000 NEXT
3010 PRINT : PRINT : PRINT
3020 IF E=1 AND P<N PRINT "GAME OVER AT CONCLUSION OF THIS ROUND"
3030 IF E=1 AND P=N THEN 3250
3040 P=P+1
3050 IF P=N+1 THEN R=R+1
3060 IF P=N+1 THEN P=1
3070 GOSUB 4050
3080 PRINT "STAND BY, PLEASE ";
3090 FOR LP=0 TO 7
3100   WT(LP)=0
3110   WC(LP)=0
3120   WD(LP)=0
3130   FOR LQ=0 TO 4
3140     W$(LP,LQ)=""
3150     F(LP,LQ)=0
3160 NEXT LQ, LP
3170 LX$=""
3180 LY$=""
3190 S=0

3200 W0$=""
3210 X1$=""
3220 S1=0
3230 A1$=""
3240 GOTO 1080
3250 PRINT "GAME NOW OVER"
3260 PRINT "TO PLAY AGAIN KEY A "
3270 Z$=INKEY$
3280 IF Z$="" THEN 3270
3290 IF Z$="A" THEN RUN
3300 END
3310 REM ROUTINE TO DELETE ENTRIES BY PLAYER
3320 A$=A1$
3330 GOTO 3350
3340 A$=RIGHT$(A$,LEN(A$)-2)
3350 PRINT @ 69, "DELETING "A$"                "
3360 IF LEN(A$)<3 THEN 3430
3370 X1$=STRING$(LEN(A$)," ")
3380 L=LEN(A$)-3
3390 IF L>7 L=7
3400 FOR LP=0 TO WC(L)-1
3410  IF A$=W$(L,LP) THEN 3450
3420 NEXT
3430 PRINT @ 90, "              NOT FOUND";
3440 GOTO 1480
3450 W$(L,LP)=""
3460 W$(L,LP)=W$(L,WC(L)-1)
3470 W$(L,WC(L)-1)=""
3480 W0$=W$(L,LP)
3490 ON L+1 GOTO 3500, 3540, 3580, 3620, 3660, 3700,         3740, 3780
```



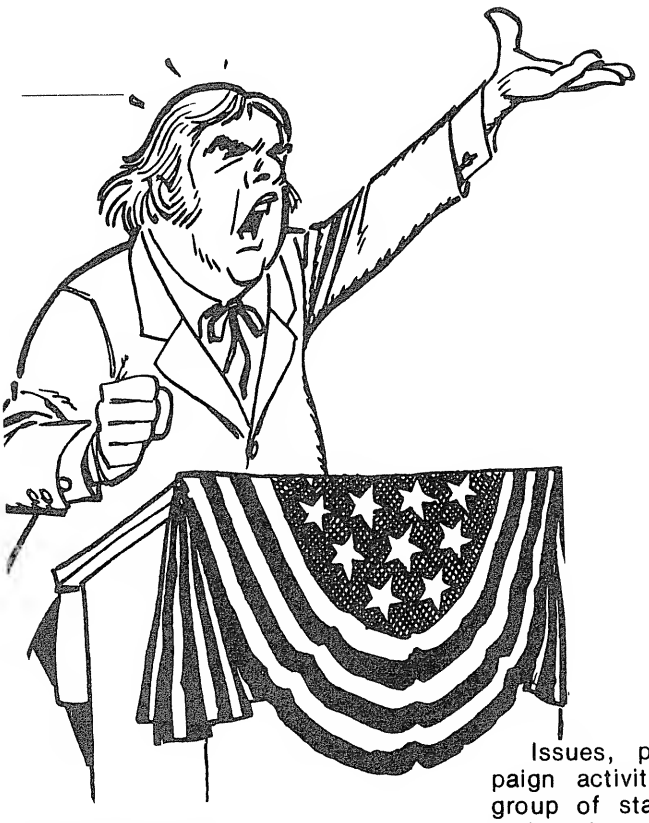
```
3500 PRINT @ 325+64*LP,X1$;
3510 PRINT @ 325+64*LP,W0$;
3520 PRINT @ 325+64*(WC(L)-1),X1$;
3530 GOTO 3810
3540 PRINT @ 335+64*LP,X1$;
3550 PRINT @ 335+64*LP,W0$;
3560 PRINT @ 335+64*(WC(L)-1),X1$;
3570 GOTO 3810
3580 PRINT @ 346+64*LP,X1$;
3590 PRINT @ 346+64*LP,W0$;
3600 PRINT @ 346+64*(WC(L)-1),X1$;
3610 GOTO 3810
3620 PRINT @ 358+64*LP,X1$;
3630 PRINT @ 358+64*LP,W0$;
3640 PRINT @ 358+64*(WC(L)-1),X1$;
3650 GOTO 3810
3660 PRINT @ 371+64*LP,X1$;
3670 PRINT @ 371+64*LP,W0$;
3680 PRINT @ 371+64*(WC(L)-1),X1$;
3690 GOTO 3810
3700 PRINT @ 709+64*LP,X1$;
3710 PRINT @ 709+64*LP,W0$;
3720 PRINT @ 709+64*(WC(L)-1),X1$;
3730 GOTO 3810
3740 PRINT @ 724+64*LP,X1$;
3750 PRINT @ 724+64*LP,W0$;
3760 PRINT @ 724+64*(WC(L)-1),X1$;
3770 GOTO 3810
3780 PRINT @ 740+64*LP,X1$;
3790 PRINT @ 740+64*LP,W0$;
3800 PRINT @ 740+64*(WC(L)-1),STRING$(LEN(W0$)," ");
3810 WC(L)=WC(L)-1
3820 GOTO 1480
3830 REM S/R TO DELETE INVALID ENTRIES
3840 PP=323 : PQ=328
3850 GOTO 3990
3860 PP=333 : PQ=339
3870 GOTO 3990
3880 PP=344   PQ=351
3890 GOTO 3990
3900 PP=356 : PQ=364
3910 GOTO 3990
3920 PP=369 : PQ=378
3930 GOTO 3990
3940 PP=707 : PQ=717
3950 GOTO 3990
3960 PP=722 : PQ=733
3970 GOTO 3990
3980 PP=738   PQ=750
3990 PRINT @ PP+64*LQ,PP$;
4000 PRINT @ PQ+64*LQ,PQ$;
4010 WD(LP)=WD(LP)+1
4020 RETURN
4030 PRINT @ 108, CHR$(143) : RETURN
4040 PRINT @ 108, " " : RETURN
4050 PRINT @ 896, "TO CONTINUE, PRESS ANY KEY.";
4060 FOR LP=0 TO 75 : AA$=INKEY$ : IF AA$<>"" THEN 4100ELSE NEXT
4070 PRINT @ 896, XX$;
4080 FOR LP=0 TO 50 : AA$=INKEY$ : IF AA$<>"" THEN 4100ELSE NEXT
4090 GOTO 4050
4100 CLS
4110 PRINT
4120 RETURN
```





The Presidential Campaign

Ralph G. White



Game Description

The Presidential Campaign allows the user to run for president. The program consumes almost all available memory in a 16K level II TRS-80, so some things could not be included, and there was not enough room to load a line renumbering program.

The states are divided into six groups:

The New England states
The upper midwest and middle Atlantic states
The southern states
The great plains states
The southwest states
The northwest and west coast states

Ralph G. White, 529 South Vermont, Columbus, KS 66725.

Issues, party affiliation, campaign activities, etc. affect each group of states differently. Some actions have an equal effect on all states. So, when faced with political decisions, sometimes some people will be more pleased than others, and sometimes some actions may be highly unpopular in some areas.

The incumbent initially gets a 10% edge. A routine to determine the popularity of the president then adjusts the figure accordingly. Party affiliation of the user also affects the initial conditions.

Not only does the user get to choose whether to be the incumbent or challenger and whether to be a Democrat or Republican, but also to determine which of six different issues will be the most important issue to their campaign and which issue will be the least important. All of these decisions can influence the effectiveness of your decisions. Which issues are chosen for most important and least important do not affect initial conditions.

The user has nine months in which to campaign. Status in an individual state can be improved by either campaigning in the state or spending campaign money in it. The influence you and your money have in each state varies. The major factor is the number of electoral votes. The number of days campaigning or the amount of money spent is also of importance. It costs $1100 per day to visit each state, some of the days you plan to be in a state can be designated for fund raising as well as campaigning. Fund raising does not help your popularity in a state, but it feeds the campaign treasury. Campaigning increases popularity, but depletes the treasury.

Aside from meeting campaign expenses, the money can be spent in each state to finance campaign committees. The maximum that can be spent in each state at one time is $50,000. You are allowed to visit as many states as time and money allow. You can spend as much money each month as can be afforded.

At the end of each month, you will be given a report on balance of the campaign treasury at the beginning of the month and at the end, the contributions and expenditures for the month, and the results of a political poll which will show your popularity status for a state chosen at random.

Before the beginning of the next month a political event will happen. How the event affects you depends upon the conditions you set forth at the beginning of the program. Some of the events require you to make a decision and the course of action taken influences your status.

At the end you receive a state by state account of the results and how

the electoral votes were cast. Who won the electoral votes of each state is displayed, and a running total of the total electoral votes is kept.

Program Information

The Presidential Campaign is written in Radio Shack level II Basic. At least 15K of free memory is needed to run the program.

Following is a table of routines and where they are located in the program:

Title and housekeeping chores	5 - 999
Initial campaign conditions	1000 - 2035
Monthly campaign activities	3000 - 3300
End of month campaign report	4000 - 4999
State by state popularity adjustment	20000 - 21003
Monthly political event	21000 - 21999
Individual state date	30000 - 30510
Scenario	31000 - 31650

□

```
                    THE  PRESIDENTIAL  CAMPAIGN


                              S C E N A R I O

   YOU HAVE DECIDED TO RUN FOR PRESIDENT  AND HAVE OBTAINED
NOMINATION OF YOUR PARTY  THE CAMPAIGN BEGINS NINE MONTHS
BEFORE THE ELECTION  YOU HAVE THE OPTIONS OF DECIDING WHICH
STATES TO VISIT EACH MONTH, HOW MANY DAYS YOU WANT TO SPEND IN
THE STATES YOU CHOOSE TO VISIT, AND WHETHER THE VISIT IS FOR
CAMPAIGNING (WHICH WINS POPULAR VOTES), OR FOR FUND RAISING
(WHICH WINS NO POPULAR VOTES, BUT BRINGS IN CONTRIBUTIONS TO
MEET EXPENSES AND FINANCE CAMPAIGN ACTIVITIES IN OTHER STATES)
THE MONEY THAT IS IN THE CAMPAIGN TREASURY CAN BE SPENT AS YOU
WISH IN ANY STATE
   AT THE BEGINNING OF THE CAMPAIGN, YOU ARE ALLOWED TO MAKE
SOME POLITICAL DECISIONS  THESE WILL AFFECT THE INITIAL
ATTITUDES OF THE VOTERS WITH RESPECT TO YOU AND YOUR OPPONENT
THROUGH OUT THE CAMPAIGN, YOU WILL HAVE TO MAKE ADDITIONAL
POLITICAL DECISIONS THAT WILL INFLUENCE VOTER OPINION  AS
WITH ALL POLITICAL DECISIONS, WHATEVER YOU DECIDE WILL NOT
PLEASE EVERYBODY  IN ADDITION, SOME OF YOUR DECISIONS WILL BE
COMPARED TO DECISIONS YOU HAVE MADE EARLIER TO DETERMINE YOUR
SINCERITY  SO WEIGH THE IMPLICATIONS OF EACH DECISION
CAREFULLY.  IN SOME CASES, CHANGING POSITIONS DURING A
CAMPAIGN CAN BE THE BEST STRATEGY, OTHER TIMES IT MAY BE
DISASTROUS
   AT THE END OF EACH MONTH YOU WILL RECEIVE A REPORT OF THE
FINANCES OF THE CAMPAIGN TREASURY  YOU WILL BE SHOWN THE
      BALANCE AT THE BEGINNING OF THE MONTH
      BALANCE AT THE END OF THE MONTH
      TOTAL CONTRIBUTIONS DURING THE MONTH
      TOTAL EXPENDITURES DURING THE MONTH
   CAMPAIGNING IS EXPENSIVE, NOT JUST THE MONEY YOU DECIDE TO
SPEND IN STATES, BUT ALSO FOR YOUR ACTUAL CAMPAIGN VISITS TO
THE VARIOUS STATES  IT IS HELPFUL TO SPEND TIME FUND RAISING
ON VISITS TO OTHER STATES TO MAINTAIN THE CAMPAIGN TREASURY
WITH CONTRIBUTIONS
   THERE ARE A FEW CAMPAIGN LAWS
   YOU CAN NOT PUT THE CAMPAIGN TREASURY IN DEBT
   A $50,000 MAXIMUM IS PLACED ON EACH TRANSACTION
   UNREPORTED CAMPAIGN CONTRIBUTIONS ARE ILLEGAL
      (YOU MAY BE TEMPTED TO ACCEPT SOME IF THE TREASURY
      GETS LOW ON FUNDS  YOU MAY EVEN GET AWAY WITH IT
      YOU MAY GET AWAY WITH IT MORE THAN ONCE  HOWEVER,
      YOU MIGHT GET CAUGHT  IT MAY COST YOU THE ELECTION,
      IT MAY JUST COST YOU A FEW VOTES )
   YOU CAN CAMPAIGN AS MANY DAYS PER MONTH AS YOU WISH AND
      VISIT AS MANY STATES AS YOU WISH.  THE MAXIMUM DAYS
      AVAILABLE EACH MONTH TO CAMPAIGN IS 30
   AT THE END OF EACH MONTH, YOU WILL BE SHOWN YOUR STATUS
IN ONE STATE--AS OF THE END OF THAT MONTH.  THIS IS THE ONLY
INDICATION YOU WILL RECEIVE ON YOUR PROGRESS
   AT THE END OF THE CAMPAIGN, THE ELECTION IS HELD AND YOU
WILL RECEIVE A STATE BY STATE ACCOUNTING OF THE RESULTS
YOU WILL BE SHOWN THE NUMBER OF ELECTORAL VOTES AWARDED BY
EACH STATE, TO WHOM THEY WERE AWARDED, THE TOTAL ELECTORAL
VOTES YOU HAVE RECEIVED AND THE TOTAL ELECTORAL VOTES YOUR
OPPONENT HAS RECEIVED

1)  BE SURE TO SPELL EACH STATE CORRECTLY
2)  DO NOT USE A DOLLAR SIGN WHEN ENTERING AMOUNTS OF MONEY
3)  DO NOT USE A COMMA WHEN ENTERING NUMBERS

              C O N D I T I O N S

E THE CONDITIONS THAT YOU WISH TO BE TRUE

IN WHAT YEAR WILL THE ELECTION BE  1979
THAT IS NOT AN ELECTION YEAR
IN WHAT YEAR WILL THE ELECTION BE  1980
WHAT IS YOUR NAME JIMMY CARTER
WHAT IS YOUR OPPONENT'S NAME RONALD REAGAN

'1' TO BE THE INCUMBENT
TYPE '2' TO BE THE CHALLENGER
WHAT IS YOUR CHOICE  1

'1' TO BE A DEMOCRAT
TYPE '2' TO BE A REPUBLICAN
WHICH POLITICAL PARTY  1

                    I S S U E S

MY CARTER WHICH OF THE FOLLOWING ISSUES
      (INPUT THE NUMBER  NOT THE PHRASE, PLEASE)
1) UNEMPLOYMENT                4) SOCIAL ADJUSTMENTS
2) INFLATION                   5) DEFENSE
3) ENERGY                      6) FOREIGN AFFAIRS

IS MOST IMPORTANT TO YOUR CAMPAIGN  2
WHICH IS LEAST IMPORTANT TO YOUR CAMPAIGN  4
* * * * * * *     * * * * * * *     * * * * * * *

FEBRUARY  1980              9  MO  BEFORE ELECTION
YOUR CAMPAIGN FUND HAS      $500,000 00
WHAT STATE DO YOU WISH TO VISIT NEW YORK
YOU HAVE  30  UNSCHEDULED DAYS LEFT THIS MONTH
HOW MANY DAYS DO YOU WISH TO SPEND THERE  20
HOW MANY OF THE  20  DAYS WILL BE FOR FUND RAISING AND HOW
MANY DAYS WILL BE FOR CAMPAIGNING
DAYS CAMPAIGNING  10
DAYS FUND RAISING  10
DO YOU WISH TO VISIT ANOTHER STATE (YES/NO) NO

SPEND CAMPAIGN MONEY IN WHAT STATE CALIFORNIA
YOUR CAMPAIGN FUND HAS      $499,866 00
THE MOST YOU CAN SPEND AT ONE TIME IN A STATE IS $50,000
HOW MUCH DO YOU WISH TO SPEND  50000
DO YOU WISH TO SPEND MONEY IN ANOTHER STATE (YES/NO) NO


              MONTHLY REPORT TO THE ELECTION COMMITTEE
CAMPAIGN FUNDS          BEGINNING OF MONTH     END OF MONTH
                            $500,000 00           $449,866 00
CONTRIBUTIONS =   $21,866 00     EXPENDITURES =   $72,000 00

LLS SHOW YOU ARE AHEAD OF RONALD REAGAN
IN TEXAS  YOU HAVE  51 % OF THE VOTE

THE PRESIDENT OF A LARGE UNION PROMISES THE SUPPORT OF THE
UNION'S MEMBERS IF YOU MAKE SOME PRO-UNION CAMPAIGN SPEECHES
WILL YOU ACCEPT HIS HELP YES
* * * * * * *     * * * * * * *     * * * * * * *

 MARCH  1980              8  MO  BEFORE ELECTION
YOUR CAMPAIGN FUND HAS      $449,866 00
WHAT STATE DO YOU WISH TO VISIT NEW JERSEY
YOU HAVE  30  UNSCHEDULED DAYS LEFT THIS MONTH
HOW MANY DAYS DO YOU WISH TO SPEND THERE  15
HOW MANY OF THE  15  DAYS WILL BE FOR FUND RAISING AND HOW
MANY DAYS WILL BE FOR CAMPAIGNING
DAYS CAMPAIGNING  9
DAYS FUND RAISING  6
DO YOU WISH TO VISIT ANOTHER STATE (YES/NO) YES

WHAT STATE DO YOU WISH TO VISIT COLORADO
YOU HAVE  15  UNSCHEDULED DAYS LEFT THIS MONTH
HOW MANY DAYS DO YOU WISH TO SPEND THERE  8
HOW MANY OF THE  8  DAYS WILL BE FOR FUND RAISING AND HOW
MANY DAYS WILL BE FOR CAMPAIGNING
DAYS CAMPAIGNING  8
DAYS FUND RAISING  0
DO YOU WISH TO VISIT ANOTHER STATE (YES/NO) NO


SPEND CAMPAIGN MONEY IN WHAT STATE NEBRASKA
YOUR CAMPAIGN FUND HAS      $430,866 00
THE MOST YOU CAN SPEND AT ONE TIME IN A STATE IS $50,000
HOW MUCH DO YOU WISH TO SPEND  20000
DO YOU WISH TO SPEND MONEY IN ANOTHER STATE (YES/NO) NO


              MONTHLY REPORT TO THE ELECTION COMMITTEE
CAMPAIGN FUNDS          BEGINNING OF MONTH     END OF MONTH
                            $449,866 00           $410,006 00
CONTRIBUTIONS =   $5,440 00     EXPENDITURES =   $45,300 00

LS SHOW YOU ARE AHEAD OF RONALD REAGAN
IN DELAWARE  YOU HAVE  70 % OF THE VOTE

THERE IS A SHORTAGE OF ALL PETROLEUM PRODUCTS--ESPECIALLY
GASOLINE  THE REASONS FOR THE SHORTAGE ARE UNCLEAR AT THIS
TIME
* * * * * * *     * * * * * * *     * * * * * * *

DATE  APRIL  1980              7  MO  BEFORE ELECTION
YOUR CAMPAIGN FUND HAS      $410,006 00
WHAT STATE DO YOU WISH TO VISIT OHIO
YOU HAVE  30  UNSCHEDULED DAYS LEFT THIS MONTH
HOW MANY DAYS DO YOU WISH TO SPEND THERE  20
HOW MANY OF THE  20  DAYS WILL BE FOR FUND RAISING AND HOW
MANY DAYS WILL BE FOR CAMPAIGNING
DAYS CAMPAIGNING  21
DAYS FUND RAISING  4
HOW MANY OF THE  20  DAYS WILL BE FOR FUND RAISING AND HOW
MANY DAYS WILL BE FOR CAMPAIGNING
DAYS CAMPAIGNING  16
DAYS FUND RAISING  4
DO YOU WISH TO VISIT ANOTHER STATE (YES/NO) NO


SPEND CAMPAIGN MONEY IN WHAT STATE TEXAS
YOUR CAMPAIGN FUND HAS      $393,339 00
THE MOST YOU CAN SPEND AT ONE TIME IN A STATE IS $50,000
HOW MUCH DO YOU WISH TO SPEND  50000
DO YOU WISH TO SPEND MONEY IN ANOTHER STATE (YES/NO) NO


              MONTHLY REPORT TO THE ELECTION COMMITTEE
CAMPAIGN FUNDS          BEGINNING OF MONTH     END OF MONTH
                            $410,006 00           $343,339 00
CONTRIBUTIONS =   $5,333 00     EXPENDITURES =   $72,000 00

S SHOW RONALD REAGAN IS AHEAD OF YOU
IN NEVADA  HE HAS  55 % OF THE VOTE

THERE IS A SHORTAGE OF ALL PETROLEUM PRODUCTS--ESPECIALLY
GASOLINE  THE REASONS FOR THE SHORTAGE ARE UNCLEAR AT THIS
TIME
* * * * * * *     * * * * * * *     * * * * * * *

DATE  MAY  1980              6  MO  BEFORE ELECTION
YOUR CAMPAIGN FUND HAS      $343,339 00
WHAT STATE DO YOU WISH TO VISIT PENNSYLVANIA
YOU HAVE  30  UNSCHEDULED DAYS LEFT THIS MONTH
HOW MANY DAYS DO YOU WISH TO SPEND THERE  26
HOW MANY OF THE  26  DAYS WILL BE FOR FUND RAISING AND HOW
MANY DAYS WILL BE FOR CAMPAIGNING
```

```
DAYS CAMPAIGNING  21
DAYS FUND RAISING  5
DO YOU WISH TO VISIT ANOTHER STATE (YES/NO) NO


SPEND CAMPAIGN MONEY IN WHAT STATE MONTANA
YOUR CAMPAIGN FUND HAS       $321,939.00
THE MOST YOU CAN SPEND AT ONE TIME IN A STATE IS $50,000.
HOW MUCH DO YOU WISH TO SPEND  30000
DO YOU WISH TO SPEND MONEY IN ANOTHER STATE (YES/NO) NO



                MONTHLY REPORT TO THE ELECTION COMMITTEE
CAMPAIGN FUNDS         BEGINNING OF MONTH      END OF MONTH
                              $343,339.00            $291,939.00
CONTRIBUTIONS =     $7,200.00      EXPENDITURES =    $58,600.00

POLLS SHOW YOU ARE AHEAD OF RONALD REAGAN
IN NEBRASKA.   YOU HAVE  58 % OF THE VOTE.

THE PRESIDENT OF A LARGE UNION PROMISES THE SUPPORT OF THE
UNION'S MEMBERS IF YOU MAKE SOME PRO-UNION CAMPAIGN SPEECHES.
WILL YOU ACCEPT HIS HELP YES
* * * * * * *       * * * * * * *       * * * * * * *

 JUNE  1980                5  MO.  BEFORE ELECTION
YOUR CAMPAIGN FUND HAS       $291,939.00
WHAT STATE DO YOU WISH TO VISIT IOWA
YOU HAVE  30  UNSCHEDULED DAYS LEFT THIS MONTH.
HOW MANY DAYS DO YOU WISH TO SPEND THERE  15
HOW MANY OF THE  15  DAYS WILL BE FOR FUND RAISING AND HOW
MANY DAYS WILL BE FOR CAMPAIGNING.
DAYS CAMPAIGNING  15
DAYS FUND RAISING  0
DO YOU WISH TO VISIT ANOTHER STATE (YES/NO) NO


SPEND CAMPAIGN MONEY IN WHAT STATE ARIZONA
YOUR CAMPAIGN FUND HAS       $275,439.00
THE MOST YOU CAN SPEND AT ONE TIME IN A STATE IS $50,000.
HOW MUCH DO YOU WISH TO SPEND  40000
DO YOU WISH TO SPEND MONEY IN ANOTHER STATE (YES/NO) NO



                MONTHLY REPORT TO THE ELECTION COMMITTEE
CAMPAIGN FUNDS         BEGINNING OF MONTH      END OF MONTH
                              $291,939.00            $235,439.00
CONTRIBUTIONS =         $0.00      EXPENDITURES =    $56,500.00

S SHOW YOU ARE AHEAD OF RONALD REAGAN
IN MARYLAND.   YOU HAVE  64 % OF THE VOTE.

A POLITICAL BOSS PROMISES TO CONTRIBUTE  11985  DOLLARS TO YOUR
CAMPAIGN IF YOU WILL APPOINT SOME OF HIS FRIENDS TO POWERFUL
POSITIONS IF YOU WIN.  THIS CONTRIBUTION IS NOT LEGAL.
WILL YOU ACCEPT THE CONTRIBUTIONS (YES/NO) NO
* * * * * * *       * * * * * * *       * * * * * * *

DATE: JULY  1980               4  MO.  BEFORE ELECTION
YOUR CAMPAIGN FUND HAS       $235,439.00
WHAT STATE DO YOU WISH TO VISIT ALABAMA
YOU HAVE  30  UNSCHEDULED DAYS LEFT THIS MONTH.
HOW MANY DAYS DO YOU WISH TO SPEND THERE  18
HOW MANY OF THE  18  DAYS WILL BE FOR FUND RAISING AND HOW
MANY DAYS WILL BE FOR CAMPAIGNING.
DAYS CAMPAIGNING  15

DAYS FUND RAISING  3
DO YOU WISH TO VISIT ANOTHER STATE (YES/NO) NO


END CAMPAIGN MONEY IN WHAT STATE KANSAS
YOUR CAMPAIGN FUND HAS       $217,079.00
THE MOST YOU CAN SPEND AT ONE TIME IN A STATE IS $50,000.
HOW MUCH DO YOU WISH TO SPEND  30000
DO YOU WISH TO SPEND MONEY IN ANOTHER STATE (YES/NO) NO



                MONTHLY REPORT TO THE ELECTION COMMITTEE
CAMPAIGN FUNDS         BEGINNING OF MONTH      END OF MONTH
                              $235,439.00            $187,079.00
CONTRIBUTIONS =     $1,440.00      EXPENDITURES =    $49,800.00

OLLS SHOW YOU ARE AHEAD OF RONALD REAGAN
IN MONTANA.   YOU HAVE  77 % OF THE VOTE.

THE U. S. IS THE TARGET OF DEMONSTRATIONS IN SEVERAL MIDDLE
EAST COUNTRIES.
SEVERAL EUROPEAN COUNTRIES HAVE ALSO BEEN CRITICAL OF OUR
FOREIGN POLICY.
* * * * * * *       * * * * * * *       * * * * * * *

AUGUST  1980               3  MO.  BEFORE ELECTION
YOUR CAMPAIGN FUND HAS       $187,079.00
WHAT STATE DO YOU WISH TO VISIT KENTUCKY
YOU HAVE  30  UNSCHEDULED DAYS LEFT THIS MONTH.
HOW MANY DAYS DO YOU WISH TO SPEND THERE  21
HOW MANY OF THE  21  DAYS WILL BE FOR FUND RAISING AND HOW
MANY DAYS WILL BE FOR CAMPAIGNING.
DAYS CAMPAIGNING  19
DAYS FUND RAISING  2
DO YOU WISH TO VISIT ANOTHER STATE (YES/NO) NO


SPEND CAMPAIGN MONEY IN WHAT STATE VIRGINIA
YOUR CAMPAIGN FUND HAS       $164,939.00
THE MOST YOU CAN SPEND AT ONE TIME IN A STATE IS $50,000.
HOW MUCH DO YOU WISH TO SPEND  5500
DO YOU WISH TO SPEND MONEY IN ANOTHER STATE (YES/NO) NO



                MONTHLY REPORT TO THE ELECTION COMMITTEE
CAMPAIGN FUNDS         BEGINNING OF MONTH      END OF MONTH
                              $187,079.00            $159,439.00
CONTRIBUTIONS =       $960.00      EXPENDITURES =    $28,600.00

S SHOW YOU ARE AHEAD OF RONALD REAGAN
IN CALIFORNIA.   YOU HAVE  56 % OF THE VOTE.

THE U. S. IS THE TARGET OF DEMONSTRATIONS IN SEVERAL MIDDLE
EAST COUNTRIES.
SEVERAL EUROPEAN COUNTRIES HAVE ALSO BEEN CRITICAL OF OUR
FOREIGN POLICY.
* * * * * * *       * * * * * * *       * * * * * * *

ATE: SEPTEMBER  1980               2  MO.  BEFORE ELECTION
YOUR CAMPAIGN FUND HAS       $159,439.00
WHAT STATE DO YOU WISH TO VISIT DELAWARE
YOU HAVE  30  UNSCHEDULED DAYS LEFT THIS MONTH.
HOW MANY DAYS DO YOU WISH TO SPEND THERE  15
HOW MANY OF THE  15  DAYS WILL BE FOR FUND RAISING AND HOW
MANY DAYS WILL BE FOR CAMPAIGNING.
DAYS CAMPAIGNING  12
DAYS FUND RAISING  3
DO YOU WISH TO VISIT ANOTHER STATE (YES/NO) NO

SPEND CAMPAIGN MONEY IN WHAT STATE MINNESOTA
YOUR CAMPAIGN FUND HAS       $143,419.00
THE MOST YOU CAN SPEND AT ONE TIME IN A STATE IS $50,000.
HOW MUCH DO YOU WISH TO SPEND  50000
DO YOU WISH TO SPEND MONEY IN ANOTHER STATE (YES/NO) NO



                MONTHLY REPORT TO THE ELECTION COMMITTEE
CAMPAIGN FUNDS         BEGINNING OF MONTH      END OF MONTH
                              $159,439.00            $93,419.00
CONTRIBUTIONS =       $480.00      EXPENDITURES =    $66,500.00

POLLS SHOW YOU ARE AHEAD OF RONALD REAGAN
IN OHIO.   YOU HAVE  73 % OF THE VOTE.

A POLITICAL BOSS PROMISES TO CONTRIBUTE  17359  DOLLARS TO YOUR
CAMPAIGN IF YOU WILL APPOINT SOME OF HIS FRIENDS TO POWERFUL
POSITIONS IF YOU WIN.  THIS CONTRIBUTION IS NOT LEGAL.
WILL YOU ACCEPT THE CONTRIBUTIONS (YES/NO) NO
* * * * * * *       * * * * * * *       * * * * * * *

E  OCTOBER  1980               1  MO.  BEFORE ELECTION
YOUR CAMPAIGN FUND HAS       $93,419.00
WHAT STATE DO YOU WISH TO VISIT LOUISIANA
YOU HAVE  30  UNSCHEDULED DAYS LEFT THIS MONTH
HOW MANY DAYS DO YOU WISH TO SPEND THERE  16
HOW MANY OF THE  16  DAYS WILL BE FOR FUND RAISING AND HOW
MANY DAYS WILL BE FOR CAMPAIGNING.
DAYS CAMPAIGNING  16
DAYS FUND RAISING  0
DO YOU WISH TO VISIT ANOTHER STATE (YES/NO) NO


SPEND CAMPAIGN MONEY IN WHAT STATE TEXAS
YOUR CAMPAIGN FUND HAS       $75,819.00
THE MOST YOU CAN SPEND AT ONE TIME IN A STATE IS $50,000.
HOW MUCH DO YOU WISH TO SPEND  50000
DO YOU WISH TO SPEND MONEY IN ANOTHER STATE (YES/NO) CALIFORNIA



                MONTHLY REPORT TO THE ELECTION COMMITTEE
CAMPAIGN FUNDS         BEGINNING OF MONTH      END OF MONTH
                              $93,419.00             $25,819.00
CONTRIBUTIONS =         $0.00      EXPENDITURES =    $67,600.00

POLLS SHOW YOU ARE AHEAD OF RONALD REAGAN
IN INDIANA.   YOU HAVE  54 % OF THE VOTE.

THE RATE OF INFLATION HAS DROPPED

OO  OO  OO  OO  OO  OO  OO  OO  OO
                ELECTION NIGHT RESULTS
                ELECTORAL VOTES

                                    YOUR      OPPONENT'S
STATE                YOU        OPPONENT  TOTAL     TOTAL
ALABAMA               9                        9         0
ALASKA                          3              9         3
ARIZONA               6                       15         3
ARKANSAS                        6             15         9
CALIFORNIA                     45             15        54
COLORADO                        7             15        61
CONNECTICUT                     8             15        69
DELAWARE              3                       18        69
D.C.                  3                       21        69
FLORIDA                        17             21        86
GEORGIA                        12             21        98
HAWAII                          4             21       102
IDAHO                           4             21       106
ILLINOIS             26                       47       106
INDIANA              13                       60       106
IOWA                  8                       68       106
KANSAS                          7             68       113
KENTUCKY              9                       77       113
LOUISIANA            10                       87       113
MAINE                           4             87       117
MARYLAND             10                       97       117
MASSACHUSSETTS                 14             97       131
MICHIGAN             21                      118       131
MINNESOTA            10                      128       131
MISSISSIPPI                     7            128       138
MISSOURI                       12            128       150
MONTANA               4                      132       150
NEBRASKA                        5            132       155
NEVADA                          3            132       158
NEW HAMPSHIRE                   4            132       162
NEW JERSEY           17                      149       162
NEW MEXICO                      4            149       166
NEW YORK             41                      190       166
NORTH CAROLINA                 13            190       179
NORTH DAKOTA                    4            190       183
OHIO                 25                      215       183
OKLAHOMA                        8            215       191
OREGON                          6            215       197
PENNSYLVANIA         27                      242       197
RHODE ISLAND                    4            242       201
SOUTH CAROLINA                  8            242       209
SOUTH DAKOTA                    4            242       213
TENNESSEE                      10            242       223
TEXAS                          26            242       249
UTAH                            4            242       253
VERMONT                         3            242       256
VIRGINIA                       12            242       268
WASHINGTON                      8            242       276
WEST VIRGINIA                   6            242       282
WISCONSIN            11                      253       282
WYOMING                         3            253       285
RONALD REAGAN IS THE WINNER OF THE  1980  PRESIDENTIAL ELECTION.
RONALD REAGAN HAS  285  ELECTORAL VOTES--MORE THAN HIS OPPONENT,
 JIMMY CARTER
```

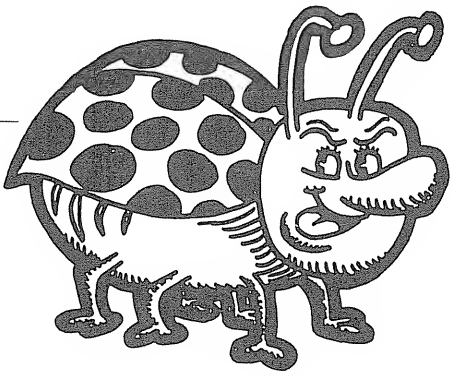
```
5 CLEAR500
20 DIMST(10,15)
400 M$(1)="FEBRUARY" M$(2)="MARCH" M$(3)="APRIL" M$(4)="MAY" M$(5)="JUNE" M$(6)="JULY" M$(7)="AUGUST"
M$(8)="SEPTEMBER" M$(9)="OCTOB
ER"
410 A$="YOUR CAMPAIGN FUND HAS $###,###,### ##"
411 B$="                    $###,###,### ##       $###,###,### ##
412 D$="CONTRIBUTIONS = $$###,### ##     EXPENDITURES = $$###,### ##"
500 FOR1=1TO6.ST(1,I)=50 ST(5,I)=50.NEXT
510 FOR1=1TO13 ST(2,I)=50 ST(4,I)=50 NEXT
520 FOR1=1TO7 ST(3,1)=50 ST(6,I)=50 NEXT
530 F=0 A#=500000
600 CLS PRINTCHR$(23) PRINT PRINT PRINT"THE PRESIDENTIAL CAMPAIGN"
615 FOR1=1TO2000.NEXT
620 GOSUB31000
1000 CLS
1010 PRINTTAB(10);"C O N D I T I O N S" PRINT PRINT
1020 PRINT"CHOOSE THE CONDITIONS THAT YOU WISH TO BE TRUE " PRINT
1030 INPUT"IN WHAT YEAR WILL THE ELECTION BE ";EY
1031 IFEY/4=INT(EY/4)GOTO1040
1032 PRINT"THAT IS NOT AN ELECTION YEAR" GOTO1030
1040 INPUT"WHAT IS YOUR NAME ";N$
1045 INPUT"WHAT IS YOUR OPPONENTS NAME ";O$
1050 PRINT PRINT"TYPE '1' TO BE THE INCUMBENT"
1055 PRINT"TYPE '2' TO BE THE CHALLENGER"
1060 INPUT"     WHAT IS YOUR CHOICE ";P1 IFP1<1ORP1>2GOTO1060
1065 PRINT.PRINT"TYPE '1' TO BE A DEMOCRAT"
1070 PRINT"TYPE '2' TO BE A REPUBLICAN"
1075 INPUT"     WHICH POLITICAL PARTY ";P2:IFP2<1ORP2>2GOTO1075
1100 PR=RND(100)
1110 IFPR<30THENPA=-8
1120 IFPR>=30ANDPR<40THENPA=-5
1130 IFPR>=40ANDPR<55THENPA=2
1140 IFPR>=55ANDPR<65THENPA=6
1150 IFPR>=65THENPA=10
1160 IFP1=2THENPA=-PA
1170 FOR1=1TO6.C(I)=PA NEXT
1180 GOSUB20000
1200 C(1)=-8 C(2)=10 C(3)=-15.C(4)=12:C(5)=6 C(6)=-9
1210 IFP2=1GOTO1290
1220 FOR1=1TO6.C(I)=-C(I):NEXT
1290 GOSUB20000
2000 CLS PRINTTAB(20);"I S S U E S" PRINT
2010 PRINTN$;", WHICH OF THE FOLLOWING ISSUES
2011 PRINTTAB(8);"(INPUT THE NUMBER, NOT THE PHRASE, PLEASE)
2015 PRINT"1) UNEMPLOYMENT";TAB(32);"4) SOCIAL ADJUSTMENTS"
2020 PRINT"2) INFLATION";TAB(32);"5) DEFENSE"
2025 PRINT"3) ENERGY";TAB(32);"6) FOREIGN AFFAIRS"
2030 PRINT:INPUT"IS MOST IMPORTANT TO YOUR CAMPAIGN ";I1
2035 INPUT"WHICH IS LEAST IMPORTANT TO YOUR CAMPAIGN ";I2
3000 FORT=1TO9
3010 CLS.PRINT"DATE.  ";M$(T);" ";EY;TAB(32);(10-T);" MO. BEFORE ELECTION"
3015 CM#=0 TS#=0
3020 PRINTUSING#$;A#,AB#=A#:MD=0
3030 INPUT"WHAT STATE DO YOU WISH TO VISIT ";V$
3035 PRINT"YOU HAVE ";(30-MD);" UNSCHEDULED DAYS LEFT THIS MONTH."
3040 INPUT"HOW MANY DAYS DO YOU WISH TO SPEND THERE ";DV
3045 IFMD+DV>30GOTO3040
3046 MD=MD+DV
3050 PRINT"HOW MANY OF THE ";DV;" DAYS WILL BE FOR FUND RAISING AND HOW":PRINT"MANY DAYS WILL BE FOR CAMPF
3060 INPUT"DAYS CAMPAIGNING ";DC
3070 INPUT"DAYS FUND RAISING ";DF
3080 IFDV<DC+DFGOTO3050
3090 RESTORE
3100 READST$,EV,I,J
3105 IFST$<>"END"GOTO3110
3106 PRINT"YOU DID NOT SPELL THE STATE CORRECTLY.  TRY AGAIN."
3107 MD=MD-DV GOTO3030
3110 IFST$<>V$GOTO3100
3120 CC#=INT(EV*1600*(DF/30)).CE=DV*1100:A#=A#+CC#-CE CM#=CM#+CC# TS#=TS#+CE
3130 ST(I,J)=ST(I,J)+INT((100-ST(I,J))*DC/30)
3140 IFMD=30GOTO3160
3150 INPUT"DO YOU WISH TO VISIT ANOTHER STATE (YES/NO) ";C$
3152 IFC$="YES"GOTO3030
3160 CLS
3200 INPUT"SPEND CAMPAIGN MONEY IN WHAT STATE ";SP$
3210 PRINTUSING#$;A#
3215 PRINT PRINT"THE MOST YOU CAN SPEND AT ONE TIME IN A STATE IS $50,000."
3220 INPUT"HOW MUCH DO YOU WISH TO SPEND ";AS
3225 IFAS>50000GOTO3215
3230 IFAS>A#GOTO3210
3240 A#=A#-AS:TS#=TS#+AS
3245 RESTORE
3250 READST$,EV,I,J
3253 IFST$<>"END"GOTO3260
3255 PRINT"YOU DID NOT SPELL THE STATE CORRECTLY  TRY AGAIN."
3256 A#=A#+AS:GOTO3200
3260 IFST$<>SP$GOTO3250
3270 ST(I,J)=ST(I,J)+INT((100-ST(I,J))*AS/10000*(1/EV))
3280 IFA#<=0GOTO4000
3290 INPUT"DO YOU WISH TO SPEND MONEY IN ANOTHER STATE (YES/NO)";C$
3300 IFC$="YES"GOTO3200
4000 CLS:PRINTTAB(15);"MONTHLY REPORT TO THE ELECTION COMMITTEE"

4010 PRINT"CAMPAIGN FUNDS       BEGINNING OF MONTH     END OF MONTH
4020 PRINTUSING#$;AB#;A# PRINT
4030 PRINTUSING#$;CM#,TS#
4040 PL=RND(51) RESTORE
4050 FOR2=1TOPL:READST$,EV,I,J.NEXT
4060 IFST(I,J)>50GOTO4100
4070 IFST(I,J)<50GOTO4120
4080 PRINT"POLLS SHOW YOU ARE EVEN WITH ";O$
4090 PRINT"IN ";ST$;" " GOTO4140
4100 PRINT"POLLS SHOW YOU ARE AHEAD OF ";O$
4110 PRINT"IN ";ST$;"   YOU HAVE ";ST(I,J);"% OF THE VOTE " GOTO4140
4120 PRINT"POLLS SHOW ";O$;" IS AHEAD OF YOU"
4130 PRINT"IN ";ST$;"   HE HAS ";(100-ST(I,J));"% OF THE VOTE "
4140 INPUT"PRESS 'ENTER' TO BEGIN NEXT MONTH ";Z$
4900 CLS GOSUB21000
4910 FORY=1TO6 ST(1,Y)=ST(1,Y)-2 ST(5,Y)=ST(5,Y)-2.NEXT
4911 FORY=1TO13 ST(2,Y)=ST(2,Y)-2 ST(4,Y)=ST(4,Y)-2 NEXT
4912 FORY=1TO7.ST(3,Y)=ST(3,Y)-2 ST(6,Y)=ST(6,Y)-2 NEXT
4998 INPUT"       PRESS 'ENTER'";C$
4999 NEXTT
5000 CLS PRINTTAB(20);"ELECTION NIGHT RESULTS" PRINT.PRINTTAB(23);"ELECTORAL VOTES"
5001 PRINTTAB(40);"YOUR";TAB(50);"OPPONENT'S"
5002 PRINT"STATE";TAB(20);"YOU";TAB(30);"OPPONENT";TAB(40);"TOTAL";TAB(50);"TOTAL"
5100 RESTORE
5105 FORK=1TO51
5110 READST$,EV,I,J
5130 IFST(I,J)>50GOTO5140
5132 OT=OT+EV X=30 GOTO5150
5140 YT=YT+EV X=20
5150 PRINTST$;TAB(X);EV;TAB(40);YT;TAB(50);OT
5160 FORTM=1TO750 NEXTTM
5190 NEXT
5200 IFVT>OTGOTO5240
5205 W$=O$ L$=N$ WT=OT GOTO5250
5240 W$=N$ L$=O$ WT=YT
5250 PRINTW$;" IS THE WINNER OF THE ";EY;" PRESIDENTIAL ELECTION."
5260 PRINTW$;" HAS ";WT;" ELECTORAL VOTES--MORE THAN HIS OPPONENT, ";L$;"."
9999 END
20000 FOR1=1TO6 ST(1,I)=ST(1,I)+C(1).ST(5,I)=ST(5,I)+C(5) NEXT
20001 FOR1=1TO13.ST(2,I)=ST(2,I)+C(2) ST(4,I)=ST(4,I)+C(4):NEXT
20002 FOR1=1TO7.ST(3,I)=ST(3,I)+C(3).ST(6,I)=ST(6,I)+C(6).NEXT
20003 RETURN
21000 PE=RND(8)
21010 IFPE>1GOTO21100
21020 PRINT"THE U. S. IS THE TARGET OF DEMONSTRATIONS IN SEVERAL MIDDLE" PRINT"EAST
      COUNTRIES " PRINT"SEVERAL EUROPEAN COUNTRIES HAV
E ALSO BEEN CRITICAL OF OUR" PRINT"FOREIGN POLICY " C=0
21030 IFP1=1THENC=C-1
21040 IFP1=2THENC=C+1
21050 IFI1=6THENC=C+1
21060 IFI2=6THENC=C-1
21065 FORY=1TO6 C(Y)=C NEXT
21070 GOSUB20000
21080 GOTO21999
21100 IFPE>2GOTO21200
21110 IN=RND(2)
21120 PRINT"THE RATE OF INFLATION HAS ";
21125 IFIN=1THEN IN$="DROPPED"
21130 IFIN=2THENIN$="RISEN"
21135 PRINTIN$ C=0
21140 IFP1=1GOTO21175
21145 IFI1=1THENC=C+1
21150 IFI2=1THENC=C-1
21160 IFIN=1THENC=C+1
21175 IFIN=1THENC=C+1
21180 IFIN=2THENC=C-1
21185 FORY=1TO6 C(Y)=C NEXT
21190 GOTO21999
21200 IFPE>3GOTO21300
21205 PRINT"THERE IS A SHORTAGE OF ALL PETROLEUM PRODUCTS--ESPECIALLY" PRINT"GASOLINE
      THE REASONS FOR THE SHORTAGE ARE UNCLEAR AT
THIS" PRINT"TIME " C=1
21210 IFI1=3THENC=C+1
21220 IFI2=3THENC=C-1
21230 IFP1=1THENC=-C
21240 FORY=1TO6 C(Y)=C NEXT
```

```
21250 GOSUB20000
21287 GOTO21999
21300 IFPE>4GOTO21400
21305 PB=RND(10000)+10000
21310 PRINT"A POLITICAL BOSS PROMISES TO CONTRIBUTE ";PB;" DOLLARS TO YOUR" PRINT
     "CAMPAIGN IF YOU WILL APPOINT SOME OF HIS FRIENDS T
O POWERFUL" PRINT"POSITIONS IF YOU WIN.  THIS CONTRIBUTION IS NOT LEGAL "
21315 INPUT"WILL YOU ACCEPT THE CONTRIBUTIONS (YES/NO) ";C$
21320 IFC$="NO"GOTO21999
21325 IFC$="YES"GOTO21335
21330 GOTO21315
21335 F=F+1 A#=A#+PB
21340 GOTO21999
21400 IFPE>5GOTO21500
21405 PRINT"ALLEGATIONS HAVE BEEN MADE THAT YOU HAVE ACCEPTED ILLEGAL" PRINT
     "CAMPAIGN FUNDS.  YOU ARE PRESENTLY UNDER INVESTIGATION.
21410 IFF=8GOTO21450
21415 IFF>5GOTO21417
21416 PRINT"YOU HAVE BEEN FOUND GUILTY AND YOU LOSE ";"INT(100/(G-F));PRINT"PERCENT OF YOUR SUPPORT IN EACH STATE."
21417 PRINT"YOU HAVE BEEN FOUND GUILTY AND THROWN IN THE FEDERAL PEN AT":PRINT"LEAVENWORTH, KANSAS FOR THENTY YEARS."
                                                                                                                    END
21420 FORY=1TO6 ST(1,Y)=INT((1/(G-F))*ST(1,Y)) ST(5,Y)=INT((1/(G-F))*ST(5,Y)) NEXT
21421 FORY=1TO13 ST(2,Y)=INT((1/(G-F))*ST(5,Y)):ST(4,Y)=INT((1/(G-F))*ST(4,Y)):NEXT
21422 FORY=1TO7 ST(3,Y)=INT((1/(G-F))*ST(3,Y)):ST(6,Y)=INT((1/(G-F)):ST(6,Y)):NEXT
21425 GOTO21999
21450 PRINT"YOU HAVE BEEN FOUND INNOCENT " GOTO21999
21500 IFPE>6GOTO21600
21505 PRINT"YOU AND ";O$;" AGREE TO A TELEVISED DEBATE."
21510 IFI1>1GOTO21520
21515 FORY=1TO12 ST(2,Y)=ST(2,Y)+INT( 07*(100-ST(2,Y)))
21520 IFI1<5GOTO21530
21525 FORY=1TO13 ST(4,Y)=ST(4,Y)+INT(.06*(100-ST(4,Y)))
21530 IFI2>1GOTO21540
21535 FORY=1TO12 ST(2,Y)=ST(2,Y)-INT(.05*ST(2,Y))
21540 IFI2<5GOTO21550
21545 FOORY=1TO13:ST(4,Y)=ST(4,Y)-INT(.02*ST(2,Y))
21550 DB=RND(3)
21555 IFDB=2GOTO21586
21560 IFDB=1GOTO21581
21570 FORY=1TO6 ST(1,Y)=ST(1,Y)+INT( 02*(100-ST(1,Y))):ST(5,Y)=ST(5,Y)+INT( 02*(100-ST(5,Y))):NEXT
21571 FORY=1TO13 ST(2,Y)=ST(2,Y)+INT( 01*(100-ST(2,Y))) ST(4,Y)=ST(4,Y)+INT( 01*(100-ST(4,Y))):NEXT
21572 FORY=1TO7 ST(3,Y)=ST(3,Y)+INT( 01*(100-ST(3,Y))):ST(6,Y)=ST(6,Y)+INT(.01*(100-ST(6,Y))):NEXT
21580 GOTO21585
21581 FORY=1TO6:ST(1,Y)=ST(1,Y)-INT( 02*ST(1,Y)):ST(5,Y)=ST(5,Y)-INT(.02*ST(5,Y)) NEXT
21582 FORY=1TO13 ST(2,Y)=ST(2,Y)-INT( 01*ST(2,Y)).ST(4,Y)=ST(4,Y)-INT( 01*ST(4,Y)):NEXT
21583 FORY=1TO7:ST(3,Y)=ST(3,Y)-INT( 01*ST(3,Y)):ST(6,Y)=ST(6,Y)-INT( 01*ST(6,Y)):NEXT
21585 IFDB=1PRINT"YOU LOST THE DEBATE "
21586 IFDB=2PRINT"THE DEBATE WAS A DRAW."
21587 IFDB=3PRINT"YOU WON THE DEBATE "
21590 GOTO21999
21600 IFPE>7GOTO21700
21602 PRINT"THE PRESIDENT OF A LARGE UNION PROMISES THE SUPPORT OF THE"
      PRINT"UNION'S MEMBERS IF YOU MAKE SOME PRO-UNION CAMPAIGN SP
EECHES "
21610 INPUT"WILL YOU ACCEPT HIS HELP ";C$
21615 IFC$="YES"GOTO21630
21620 IFC$="NO"GOTO21640
21625 GOTO21610
21630 C(1)=0 C(2)=2 C(3)=-2 C(4)=-1 C(5)=1 C(6)=-1:GOSUB20000
21635 GOTO21999
21640 C(1)=0 C(2)=-2 C(3)=3 C(4)=1 C(5)=0 C(6)=0:GOSUB20000
21645 GOTO21999
21700 PRINT"FARMERS AND RANCHERS WANT YOU TO CAMPAIGN THAT THEY SHOULD"
      PRINT"RECEIVE HIGHER PRICES FOR THEIR PRODUCTS.  KEEP IN MIN
D THE" PRINT"CONSUMERS WILL NOT LIKE THIS.":C=0
21705 INPUT"WILL YOU SUPPORT THE FARMERS AND RANCHERS (YES/NO)";C$
21710 IFC$="YES"GOTO21725
21715 IFC$="NO"GOTO21730
21720 GOTO21705
21725 C(1)=-1 C(2)=-1 C(3)=3 C(4)=2 C(5)=-1 C(6)=2 GOTO21840
21730 C(1)=2 C(2)=3 C(3)=-2 C(4)=-2 C(5)=1:C(6)=-2
21840 GOSUB20000
21999 RETURN
30000 DATAALABAMA,9,4,9
30010 DATAALASKA,3,5,4
30020 DATAARIZONA,6,6,4
30030 DATAARKANSAS,6,4,12
30040 DATACALIFORNIA,45,5,6
30050 DATACOLORADO,7,6,7
30060 DATACONNECTICUT,8,1,5
30070 DATADELAWARE,3,2,3
30080 DATAD C.,3,2,12
30090 DATAFLORIDA,17,4,7
30100 DATAGEORGIA,12,4,8
30110 DATAHAWAII,4,5,5
30120 DATAIDAHO,4,5,1
30130 DATAILLINOIS,26,2,9
30140 DATAINDIANA,13,2,8
30150 DATAIOWA,8,3,2
30160 DATAKANSAS,7,3,5
30170 DATAKENTUCKY,9,4,3
30180 DATALOUISIANA,10,4,11
30190 DATAMAINE,4,1,1
30200 DATAMARYLAND,10,2,5
30210 DATAMASSACHUSETTS,14,1,5
30220 DATAMICHIGAN,21,2,7
30230 DATAMINNESOTA,10,2,11
30240 DATAMISSISSIPPI,7,4,10
30250 DATAMISSOURI,12,4,13
30260 DATAMONTANA,4,3,6
30270 DATANEBRASKA,5,3,4
30280 DATANEVADA,3,6,5
30290 DATANEW HAMPSHIRE,4,1,2
30300 DATANEW JERSEY,17,2,4
30310 DATANEW MEXICO,4,6,3
30320 DATANEW YORK,41,2,1
30330 DATANORTH CAROLINA,13,4,5
30340 DATANORTH DAKOTA,4,3,1
30350 DATAOHIO,25,2,6

30360 DATAOKLAHOMA,8,6,2
30370 DATAOREGON,6,5,3
30380 DATAPENNSYLVANIA,27,2,2
30390 DATARHODE ISLAND,4,1,6
30400 DATASOUTH CAROLINA,8,4,6
30410 DATASOUTH DAKOTA,4,3,3
30420 DATATENNESSEE,10,4,4
30430 DATATEXAS,26,6,1
30440 DATAUTAH,4,6,6
30450 DATAVERMONT,3,1,3
30460 DATAVIRGINIA,12,4,2
30470 DATAWASHINGTON,8,5,2
30480 DATAWEST YIRGINIA,6,4,1
30490 DATAWISCONSIN,11,2,10
30500 DATAWYOMING,3,3,7
30510 DATAEND,0,0,0
31000 CLS:PRINTTAB(25);"S C E N A R I O" PRINT
31010 PRINT"   YOU HAVE DECIDED TO RUN FOR PRESIDENT.  AND HAVE OBTAINED"
31020 PRINT"NOMINATION OF YOUR PARTY.  THE CAMPAIGN BEGINS NINE MONTHS"
31030 PRINT"BEFORE THE ELECTION.  YOU HAVE THE OPTIONS OF DECIDING WHICH"
31040 PRINT"STATES TO VISIT EACH MONTH, HOW MANY DAYS YOU WANT TO SPEND IN"
31050 PRINT"THE STATES YOU CHOOSE TO VISIT, AND WHETHER THE VISIT IS FOR"
31060 PRINT"CAMPAIGNING (WHICH WINS POPULAR VOTES), OR FOR FUND RAISING"
31070 PRINT"(WHICH WINS NO POPULAR VOTES, BUT BRINGS IN CONTRIBUTIONS TO"
31080 PRINT"MEET EXPENSES AND FINANCE CAMPAIGN ACTIVITIES IN OTHER STATES)."
31090 PRINT"THE MONEY THAT IS IN THE CAMPAIGN TREASURY CAN BE SPENT AS YOU"
31100 PRINT"WISH IN ANY STATE "
31110 PRINT:INPUT"PRESS 'ENTER' TO CONTINUE SCENARIO ";C$
31120 CLS:PRINT"   AT THE BEGINNING OF THE CAMPAIGN, YOU ARE ALLOWED TO MAKE"
31130 PRINT"SOME POLITICAL DECISIONS.  THESE WILL AFFECT THE INITIAL"
31140 PRINT"ATTITUDES OF THE VOTERS WITH RESPECT TO YOU AND YOUR OPPONENT "
31150 PRINT"THROUGH OUT THE CAMPAIGN, YOU WILL HAVE TO MAKE ADDITIONAL"
31160 PRINT"POLITICAL DECISIONS THAT WILL INFLUENCE VOTER OPINION.  AS"
31170 PRINT"WITH ALL POLITICAL DECISIONS, WHATEVER YOU DECIDE WILL NOT"
31180 PRINT"PLEASE EVERYBODY.  IN ADDITION, SOME OF YOUR DECISIONS WILL BE"
31190 PRINT"COMPARED TO DECISIONS YOU HAVE MADE EARLIER TO DETERMINE YOUR"
31200 PRINT"SINCERITY.  SO WEIGH THE IMPLICATIONS OF EACH DECISION"
31210 PRINT"CAREFULLY.  IN SOME CASES, CHANGING POSITIONS DURING A"
31220 PRINT"CAMPAIGN CAN BE THE BEST STRATEGY, OTHER TIMES IT MAY BE"
31230 PRINT"DISASTROUS "
31240 PRINT:INPUT"PRESS 'ENTER' TO CONTINUE SCENARIO ";C$

31250 CLS:PRINT"   AT THE END OF EACH MONTH YOU WILL RECEIVE A REPORT OF THE"
31260 PRINT"FINANCES OF THE CAMPAIGN TREASURY.  YOU WILL BE SHOWN THE"
31270 PRINT"     BALANCE AT THE BEGINNING OF THE MONTH"
31280 PRINT"     BALANCE AT THE END OF THE MONTH"
31290 PRINT"     TOTAL CONTRIBUTIONS DURING THE MONTH"
31300 PRINT"     TOTAL EXPENDITURES DURING THE MONTH"
31310 PRINT"   CAMPAIGNING IS EXPENSIVE, NOT JUST THE MONEY YOU DECIDE TO"
31320 PRINT"SPEND IN STATES, BUT ALSO FOR YOUR ACTUAL CAMPAIGN VISITS TO"
31330 PRINT"THE VARIOUS STATES.  IT IS HELPFUL TO SPEND TIME FUND RAISING"
31340 PRINT"ON VISITS TO OTHER STATES TO MAINTAIN THE CAMPAIGN TREASURY"
31350 PRINT"WITH CONTRIBUTIONS."
31370 PRINT:INPUT"PRESS 'ENTER' TO CONTINUE SCENARIO ";C$
31380 CLS:PRINT"   THERE ARE A FEW CAMPAIGN LAWS:"
31390 PRINT"   YOU CAN NOT PUT THE CAMPAIGN TREASURY IN DEBT "
31400 PRINT"   A $50,000 MAXIMUM IS PLACED ON EACH TRANSACTION."
31410 PRINT"   UNREPORTED CAMPAIGN CONTRIBUTIONS ARE ILLEGAL "
31420 PRINT"      (YOU MAY BE TEMPTED TO ACCEPT SOME IF THE TREASURY"
31430 PRINT"      GETS LOW ON FUNDS.  YOU MAY EVEN GET AWAY WITH IT "
31440 PRINT"      YOU MAY GET AWAY WITH IT MORE THAN ONCE.  HOWEVER,"
31450 PRINT"      YOU MIGHT GET CAUGHT.  IT MAY COST YOU THE ELECTION."
```

```
31460 PRINT"      IT MAY JUST COST YOU A FEW VOTES )
31470 PRINT"   YOU CAN CAMPAIGN AS MANY DAYS PER MONTH AS YOU WISH AND"
31480 PRINT"      VISIT AS MANY STATES AS YOU WISH; THE MAXIMUM DAYS"
31490 PRINT"      AVAILABLE EACH MONTH TO CAMPAIGN IS 30 "
31500 PRINT INPUT"PRESS 'ENTER' TO CONTINUE SCENARIO ";C$
31510 CLS PRINT"   AT THE END OF EACH MONTH, YOU WILL BE SHOWN YOUR STATUS"
31520 PRINT"IN ONE STATE--AS OF THE END OF THAT MONTH.  THIS IS THE ONLY"
31530 PRINT"INDICATION YOU WILL RECEIVE ON YOUR PROGRESS "
31540 PRINT"   AT THE END OF THE CAMPAIGN, THE ELECTION IS HELD AND YOU"
31550 PRINT"WILL RECEIVE A STATE BY STATE ACCOUNTING OF THE RESULTS "
31560 PRINT"YOU WILL BE SHOWN THE NUMBER OF ELECTORAL VOTES AWARDED BY"
31570 PRINT"EACH STATE, TO WHOM THEY WERE AWARDED, THE TOTAL ELECTORAL"
31580 PRINT"VOTES YOU HAVE RECEIVED AND THE TOTAL ELECTORAL VOTES YOUR"
31590 PRINT"OPPONENT HAS RECEIVED "
31600 PRINT INPUT"PRESS 'ENTER' FOR SPECIFIC INSTRUCTIONS ";C$
31610 PRINT"1)  BE SURE TO SPELL EACH STATE CORRECTLY "
31620 PRINT"2)  DO NOT USE A DOLLAR SIGN WHEN ENTERING AMOUNTS OF MONEY "
31630 PRINT"3)  DO NOT USE A COMMA WHEN ENTERING NUMBERS "
31640 PRINT INPUT"PRESS 'ENTER' TO BEGIN THE CAMPAIGN ";C$
31650 RETURN
```

Twonky Revisited

G.R. Hertel



My TRS-80 arrived early in 1978. Unable to wait for Level II, I had "taken the plunge" with a 16K Level I machine. I spent hours learning and experimenting with the limitations and surprising capabilities of Level I.

One of the first "real" programs I entered was a game from *Creative Computing* (May-June, 1977). The game, by Mark Capella, was called Twonky.

From the description it looked like an interesting game and the challenge of translating it into Level I Basic made it doubly appealing. Once the conversion of the two-dimensional matrix into the single available one-dimensional array was worked out, the translation went smoothly and the program worked. It proved to be an interesting game.

In case you have misplaced your May-June 1977 issue of *Creative Computing*, let me briefly explain Twonky. In the game, you are trapped in a 15 x 15 square maze which contains blocked squares, relocation squares (which randomly relocate you), an objective, a super-maze square (which starts the game over by setting up a new maze), and a Twonky.

To win you must reach the objective square before the Twonky reaches you. The Twonky chases you relentlessly, oblivious to all obstacles, and your only options are to run from him or shoot him. If hit, he is relocated randomly somewhere in the maze.

The only real problem is that you play this whole game in the dark. After every move, the computer tells you the distance between you and the objective and between you and the Twonky.

Level I proved to be a more powerful programming language than I had expected, but, when Level II chips became available, I was ready. I had my TRS-80 upgraded and took off into a multi-dimensional, multi-functional world. I eventually returned to Twonky and typed in the program essentially as it was originally written.

It was the same game, of course, except that it ran much faster. I dressed the game up with some graphics and gave the player a "malfunctioning tricorder which was activated by the relocation field." It printed the maze and flashed your position, the Twonky, and the objective for a few seconds (I always did want to see the maze) whenever you were relocated.

Twonky was eventually stored away and almost forgotten.

Recently, I was introduced to the game of Chase (I admit I haven't been keeping up on computer games) on a computer that wasn't really meant to play games. As games go, it did not impress me.

The program was rather cumbersome; movement was slow and not at all exciting. It occurred to me, however, that it would be rather easy to put TRS-80 graphics into the game and make it really interactive.

(For those readers, like me, who are not familiar with computer games, Chase plunks you down in a field containing electrified posts and surrounded by an electrified fence. There are several robots who continuously move toward you. Both you and the robots are annihilated upon hitting a post, so the object of the game is to maneuver the robots into posts as they chase you through the field.)

I wrote a version of Chase for my TRS-80 to see how fast it would run. I used the INKEY$ function and the four arrow keys to control direction and movement in real time. The game uses surprisingly little memory and is fun to play.

You are represented by a single TRS-80 graphics location (rectangle), and the electrified posts and robots are double locations or squares. The robots come at you relentlessly, trying to catch you. You destroy them by maneuvering them into one of the posts or into each other.

In my version, a robot-post encounter annihilates both. You lose if caught or if you run into a post or the border. There are three rounds of play in a game. The number of robots increases from 4 to 8 to 12 while the number of posts decreases from 80 to 60 to 40. (See Listing 1.)

While writing and playing this version of Chase, the similarity between robots chasing you and the Twonky chasing you occurred to me and I modified the game to conform more to the characteristics of Twonky.

G.R. Hertel, 1500 Hibiscus Ave., Winter Park, Florida 32789.

In the new version, called Pursuit (Listing 2), there is one Twonky who chases you relentlessly as you try to thread your way through the maze to the flashing objective square. The Twonky homes in on you and lets nothing stand in his way.

Running into a wall either stops you cold (the blocked squares of the original game) or flips you randomly to some other spot in the maze (the relocation squares). Movement is controlled by the four arrow keys and can get frantic as the Twonky draws near or cuts off your escape route.

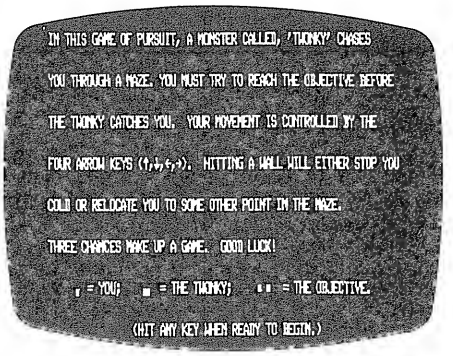
The darkness of the original Twonky game is gone (I always did want to see that maze) and so is the zap gun—you just don't have time to shoot in this version. It's real-time fun.

The game is set up to give you three chances, but the maze is more difficult each time. The goal (objective), Twonky, and you are identified in the game of Pursuit before the maze is drawn the first time.

There are enough differences between these two games to give each its own flavor. Chase runs initially at a slower pace since movement of several robots and you must be looped through successively.

As robots are destroyed, the game speeds up until, when you and just one robot remain, you are really moving.

In Pursuit, the cycle through which the program loops moves you and the Twonky, and flashes the objective. The pace is fast and constant. In order to keep the pace fast, "relocation squares" are not specific sites. Relocation may occur (30% chance) whenever you hit a wall. Also, your destination upon relocation is not checked.



As a result, you may find yourself inside a wall.

Don't give up, just keep trying to move out. Eventually that random number generator will come up in your favor and you will be relocated—if the Twonky doesn't get there first!

Instructions for Chase are omitted from the listing, but could be added if desired. There are brief instructions included in the Pursuit listing. Obviously, these are games designed specifically for the TRS-80. They are based upon TRS-80 graphics and functions. □

Program Notes for Chase

Lines 60-90 draw the border.

Lines 100-110 position posts.

Lines 120-150 select and store robot positions.

Lines 160-170 select an empty position for your starting location.

Line 180 moves you.

Lines 190-230 scan the keyboard for instruction and convert any received into horizontal or vertical instruction.

Lines 240-330 control robot movement.

The remainder of the program includes the beginning and ending routines.

Upon depressing an arrow key, you will move in that direction — one unit per move. Depressing the same key again will double your speed to two locations per move. By depressing the opposite direction key, you can slow down, stop, and reverse direction. The maximum speed is limited to two units per move.

The robots move at the constant speed of one unit per move. Thus, you can outrun the robots and even leap over posts. Diagonal movement is possible for both you and the robots.

Program Notes for Pursuit

Lines 60-90 draw the border.

Lines 100-140 pick sites for you, the objective, and the Twonky; 140 checks to be sure the Twonky isn't sitting on top of you or your goal.

Lines 170-190 set up the maze.

Lines 200-220 move you, checking to see if you have hit a wall. If so, you are stopped and possibly relocated.

Lines 230-270 check for instruction from keyboard and convert it into a vertical or horizontal direction command.

Line 280 flashes the goal.

Lines 290-330 determine which way to move the Twonky.

Line 340 checks to see if you have been caught.

Line 360 checks to see if you have entered the goal.

Line 370 flashes the goal again and starts the process over.

The remainder of the program includes the beginning and ending routines.

Upon depressing an arrow, you will move in that direction until hitting a wall or depressing the opposite arrow key. Diagonal movement is possible for you, but not for the Twonky (you must have some advantage).

Listing 1. Chase.

```
10 CLEAR 100
20 DEFINT A-Z
30 DIM X(30),Y(30)
40 GOSUB 390
50 W=0:R=4:ON ERROR GOTO 380
60 H=0:V=0:CLS:PRINT@1,STRING$(62,131)
70 FOR I=15360 TO 16320 STEP 64
80 POKE I,191:POKE I+63,191:NEXT
90 PRINT@961,STRING$(62,176);
100 FOR I=1 TO 80-5*R:X=RND(124)+1:Y=RND(44)+1
110 SET(X,Y):SET(X+1,Y):NEXT
120 FOR I=1 TO R
130 X(I)=RND(123)+1:Y(I)=RND(45)+1
140 IF POINT(X(I),Y(I)) OR POINT(X(I)+1,Y(I)) THEN 130
150 SET(X(I),Y(I)):SET(X(I)+1,Y(I)):NEXT
160 X=RND(124)+1:Y=RND(44)+1
170 IF POINT(X,Y)THEN 160 ELSE SET(X,Y)
180 RESET(X,Y):X=X+H:Y=Y+V:IF POINT(X,Y) THEN 340 ELSE SET(X,Y)
190 I$=INKEY$:IF I$="" THEN 240
200 IF ASC(I$)<10 THEN H=H+2*ASC(I$)-17 ELSE 220
210 IF ABS(H)>2 THEN H=SGN(H)*2:GOTO 240 ELSE 240
220 IF ASC(I$)=10 THEN V=V+1 ELSE V=V-1
230 IF ABS(V)>2 THEN V=SGN(V)*2
240 C=0:FOR I=1 TO R:IF X(I)=0 THEN 330
250 C=C+1:RESET(X(I),Y(I)):RESET(X(I)+1,Y(I))
260 X(I)=X(I)+SGN(X-X(I))
270 Y(I)=Y(I)+SGN(Y-Y(I))
280 IF (X(I)=X OR X(I)+1=X) AND Y(I)=Y THEN 370
290 IF POINT(X(I),Y(I)) OR POINT(X(I)+1,Y(I)) THEN 310
300 SET(X(I),Y(I)):SET(X(I)+1,Y(I)):GOTO 330
310 RESET(X(I),Y(I)):RESET(X(I)+1,Y(I)):RESET(X(I)-1,Y(I))
320 RESET(X(I)+2,Y(I)):X(I)=0
330 NEXT:IF C=0 THEN 350 ELSE 180
340 CLS:PRINT CHR$(23):PRINT@524,"Y O U   L O S E !":GOTO360
350 CLS:PRINTCHR$(23):PRINT@522,"Y O U   W I N ! ! !":W=W+1
360 FOR I=1 TO 1000:NEXT:R=R+4:IF R<13 THEN 60 ELSE 420
370 CLS:PRINT CHR$(23):PRINT@530,"G O T C H A ! ! !":GOTO360
```

```
380 RESUME 340
390 CLS:PRINTCHR$(23):PRINT@470,"C H A S E"
400 PRINT@800,"BY G. R. HERTEL"
410 FOR I=1 TO 1500:NEXT:RETURN
420 CLS:PRINTCHR$(23):IF W<3 THEN 450
430 PRINT@200,"Y O U   D I D   I T ! !":PRINT@454,"YOU WON ALL THREE TIMES!"
440 PRINT@704,"C O N G R A T U L A T I O N S !":GOTO 480
450 IF W=2 PRINT@396,"TWO OUT OF THREE":PRINT@592,"NOT TOO BAD!":GOTO 480
460 IF W=1 PRINT@390,"YOU WON ONCE - BIG DEAL."
470 IF W=0 PRINT@386,"HA-HA-HA -- YOU LOST THEM ALL."
480 A$=INKEY$:FOR I=1 TO 500:NEXT:PRINT@900,"CARE TO TRY AGAIN (Y/N)?";
490 A$=INKEY$:IF A$="" THEN 490
500 IF A$="Y" THEN 50
510 IF A$<>"N"THEN 480
```

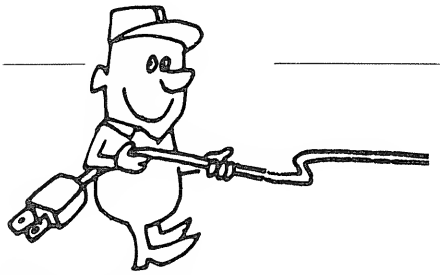
Listing 2. Pursuit.

```
10 CLEAR 100
20 DEFINT A-Z
30 GOSUB 430
40 W=0:R=0
50 RANDOM:ON ERROR GOTO 420
60 H=0:V=0:CLS:PRINT@1,STRING$(62,131)
70 FOR I=15360 TO 16320 STEP 64
80 POKE I,191:POKE I+63,191:NEXT
90 PRINT@961,STRING$(62,176);
100 X=RND(123)+1:Y=RND(41)+3:SET(X,Y)
110 GX=RND(122)+2:GY=RND(41)+3
120 SET(GX,GY):SET(GX+1,GY):SET(GX-1,GY):SET(GX+2,GY)
130 TX=RND(123)+1:TY=RND(41)+3
140 IF POINT(TX,TY) OR POINT(TX+1,TY) THEN 130
150 A$=INKEY$:RESET(GX,GY):RESET(GX+1,GY)
160 SET(TX,TY):SET(TX+1,TY):GOSUB 490
170 FOR I=1 TO 100+R*50:POKE 15360+RND(1023),191
180 A=RND(13)*64+RND(60):POKE 15424+A,140
190 POKE 15425+A,140:POKE 15426+A,140:NEXT
200 RESET(X,Y):X=X+H:Y=Y+V:IF NOT POINT(X,Y) THEN 230
210 X=X-H:Y=Y-V:H=0:V=0
220 IF RND(0)>.7 THEN X=RND(124)+1:Y=RND(45)+1
230 SET(X,Y):I$=INKEY$:IF I$="" THEN 280
240 IF ASC(I$)<10 THEN H=H+2*ASC(I$)-17 ELSE 260
250 IF ABS(H)>1 THEN H=SGN(H):GOTO 280 ELSE 280
260 IF ASC(I$)=10 THEN V=V+1 ELSE V=V-1
270 IF ABS(V)>1 THEN V=SGN(V)
280 SET(GX,GY):SET(GX+1,GY):RESET(GX-1,GY):RESET(GX+2,GY)
290 RESET(TX,TY):RESET(TX+1,TY):IF TX=X OR TX+1=X THEN 330
300 IF TY=Y THEN 320
310 ON RND(2) GOTO 320,330
320 TX=TX+SGN(X-TX):GOTO 340
330 TY=TY+SGN(Y-TY)
340 IF (TX=X OR TX+1=X) AND TY=Y THEN 410
350 SET(TX,TY):SET(TX+1,TY)
360 IF (X=GX OR X=GX+1)AND Y=GY THEN 390
370 RESET(GX,GY):RESET(GX+1,GY):SET(GX-1,GY):SET(GX+2,GY):GOTO 200
380 CLS:PRINT CHR$(23):PRINT@524,"Y O U   L O S E !":GOTO400
390 CLS:PRINTCHR$(23):PRINT@522,"Y O U   W I N ! ! !":W=W+1
400 FOR I=1 TO 1000:NEXT:R=R+1:IF R<3 THEN 50 ELSE 580
410 CLS:PRINT CHR$(23):PRINT@530,"G O T C H A ! ! !":GOTO400
420 RESUME 380
430 CLS:PRINTCHR$(23):PRINT@400,"P U R S U I T"
440 PRINT@672,"BY G. R. HERTEL"
450 PRINT@896,"DO YOU WISH INSTRUCTIONS (Y/N)?"
460 A$=INKEY$:IF A$=""THEN 460
470 IF A$="N" RETURN
480 IF A$="Y" THEN 690 ELSE 450
490 IF R>0 THEN 550
500 IF X>110 THEN P1=64*INT(Y/3)+(X-10)/2 ELSE P1=64*INT(Y/3)+(X+4)/2
510 IF GX>100 THEN PG=64*INT(GY/3)+(GX-22)/2ELSE PG=64*INT(GY/3)+(GX+6)/2
520 IF TX>100 THEN PT=64*INT(TY/3)+(TX-24)/2 ELSE PT=64*INT(TY/3)+(TX+6)/2
530 PRINT@P1,"YOU!";:PRINT@PG,"THE GOAL!";:PRINT@PT,"THE TWONKY!";
540 FOR I=1TO2500:NEXT
550 FOR I=1 TO 1500:NEXT:IF R>0 RETURN
560 PRINT@P1,"     ";:PRINT@PG,"          ";:PRINT@PT,"            ";
570 RETURN
580 CLS:PRINTCHR$(23)
590 IF W<3 THEN 620
600 PRINT@200,"Y O U   D I D   I T ! !":PRINT@454,"YOU WON ALL THREE TIMES!"
610 PRINT@704,"C O N G R A T U L A T I O N S !":GOTO 650
620 IF W=2 PRINT@396,"TWO OUT OF THREE":PRINT@592,"NOT TOO BAD!":GOTO 650
630 IF W=1 PRINT@390,"YOU WON ONCE - BIG DEAL."
640 IF W=0 PRINT@386,"HA-HA-HA -- YOU LOST THEM ALL."
650 A$=INKEY$:FOR I=1 TO 500:NEXT:PRINT@900,"CARE TO TRY AGAIN (Y/N)?";
660 A$=INKEY$:IF A$="" THEN 660
670 IF A$="Y" THEN 40
680 IF A$="N"THEN END ELSE 650
690 CLS:PRINT"IN THIS GAME OF PURSUIT, A MONSTER CALLED, 'TWONKY' CHASES":PRINT
700 PRINT"YOU THROUGH A MAZE. YOU MUST TRY TO REACH THE OBJECTIVE BEFORE":PRINT
710 PRINT"THE TWONKY CATCHES YOU.  YOUR MOVEMENT IS CONTROLLED BY THE":PRINT
720 PRINT"FOUR ARROW KEYS (^,"CHR$(92);",";CHR$(93);",";CHR$(94):").  HITTING A
```

```
WALL WILL EITHER STOP YOU":PRINT
730 PRINT"COLD OR RELOCATE YOU TO SOME OTHER POINT IN THE MAZE.":PRINT
740 PRINT"THREE CHANCES MAKE UP A GAME.  GOOD LUCK!":PRINT
750 PRINTTAB(5);CHR$(132);" = YOU;     ";CHR$(140);"  = THE TWONKY;";TAB(42);"= T
HE OBJECTIVE.":PRINT
760 PRINTTAB(15);"(HIT ANY KEY WHEN READY TO BEGIN.)"
770 PRINT@806,CHR$(132)+CHR$(136);:A$=INKEY$:FOR I=1TO50:NEXT
780 IF A$="" THEN PRINT@806,CHR$(136)+CHR$(132);:FOR I=1TO 50:NEXT:GOTO 770
790 RETURN
```

The Electric Company

Ralph White



Game Description

THE ELECTRIC COMPANY is an economic game/simulation. The user is made president of the company for ten years — unless mismanagement forces earlier retirement.

At the beginning of year one, the company has one million dollars, 1600 tons of coal, 2500 barrels of fuel oil, one power plant and 700 customers. All power plants have a capacity of one million kilowatts. The user may choose which type of fuel to use at each power plant he owns — the one he starts with as well as each one that is built. Coal plants are more expensive to build than fuel oil power plants, but they are cheaper to operate.

If more than one power plant is owned, the amount of power generated can be divided between or among the plants in any fashion desired — as long as no plant generates more than one million kilowatts. The only requirement is that enough power be generated to meet demand. Demand is determined by the number of customers. In a normal year in the program each customer consumes one thousand kilowatts.

The kilowatt demand is displayed for the user. If more than one plant is being operated, the display will subtract the power you chose to generate from previous plants and displays only the remaining kilowatts needed to meet demand.

If the amount of power generated falls short of the customers' demand then a brown-out occurs. This does not terminate the game, but damage is done to the generators and the company is charged an amount of money for repairs. The more the generated amount of electricity is below the demand, the more it will cost to repair the damage.

If the kilowatt demand outstrips the generating capacity of the company, then the generators go "poof!" and the game is terminated.

As power plants get older, they become increasingly inefficient. Each year the president of the company has to decide whether or not to remodel the plant(s). To aid in making the decision, a print out of cost for each plant, the percent of increase in efficiency for each plant, and total cost is shown. If the decision is to remodel, all plants are remodeled at the same time. The longer one waits to remodel, the more it will cost. The cost of remodeling is in part based on the increase of efficiency. Remodeling does not increase each plant's capacity. It makes each plant more efficient; it will use less fuel to make a kilowatt of electricity.

Very little in the program is allowed to remain constant. The population does not remain stable throughout the game. Each year, there is growth in the number of customers. Usually, the number of customers will increase at a rate that will require the building of a second power plant before the ten year term is completed.

Not only is the population not stable, but neither is the economy. An inflationary spiral is built into the price of all goods and services except one — the rate the company charges per kilowatt of electricity. As the program advances through each year, fuel, power plants, damage charges, etc. become more expensive.

Although the price per kilowatt the company gets for electricity does not automatically increase with inflation, a provision is made for applying for a rate increase each year. Rate increases are not allowed in two consecutive years. By waiting two years a rate increase request will be accepted if it is not too large. If it exceeds the upper limit defined by the program (which is tied to the Consumer Price Index), the request will be denied. If three or more years elapse between rate increases not only will the increase be accepted if it falls within the program's guide lines, but if it is too large, the program will yield a lesser compromise rate increase. Since an increase is not allowed every year, the upper limit for an allowable increase exceeds the Consumer Price Index.

Each year there is a 50 percent chance of some unforeseen event. These are not always bad or disastrous. Demand may rise a little faster or slower than usual, the company may be charged to repair storm damage to equipment, customers may be lost, additional customers may want to be hooked into the system (it costs the company a one-time fee to add these customers so the option of refusing service to them is given). If there are any costs involved in any of the above events, they will be influenced by

Ralph G. White, 529 South Vermont, Columbus, KS 66725.

inflation also.

Not all decisions become effective immediately. When ordering coal and fuel oil there is a lead time of one year. The fuel does not become available for use until the following year. Enough fuel must be on hand to supply the current year's needs. The same lead time of one year applies to rate increases also.

Two reports are shown. The first report, at the beginning of each year, shows the cash reserve, coal and fuel oil amounts, number of customers and number of plants owned. The second report, at the end of each year, displays the gross income, the maintenance costs for the year (maintenance costs are influenced by number of plants being operated, number of customers being served and the Consumer Price Index), percent of generating capacity that is being used, the inflation rate for the past year and the consumer price index.

Program Notes

THE ELECTRIC COMPANY is written in Radio Shack Level II Basic and requires 11K of memory. The instructions and scenario (not shown in the sample run) are located in lines 90-500.

The left hand brackets in lines 1820 and 1830 are the line printer's representation of an exponential arrow. If the Basic you have access to does not support exponentiation, substitute R*PI*SQR(PI). This yields the same result.

The "#" after some of the variables such as CR# indicates double precision. This prevents the output from appearing in scientific notation if the value of the variable exceeds six digits.

The strings defined in lines 10 and 20 are used in conjunction with the PRINTUSING statements to format the print-out of data by inserting dollar signs, commas and decimal points, or restricting numbers to two decimal places. If the PRINTUSING command is not available to you, the lines may be rewritten with only PRINT statements.

For those of you who wish to modify the degree of difficulty of the program, line 540 defines the initial values of the variables that are to be managed. □

```
10 D$="$$###,###,###.##":X$="   ###,###,###":C$="$$###.##"
20 E$="$$###,###.##"
30 CLS:PRINTCHR$(23):PRINT:PRINT:PRINT
40 A$="+--++--++--++--++--++--++--++--++--+"
50 PRINTA$:PRINT:PRINT"        ELECTRIC COMPANY":PRINT:PRINT:PRINTA$
60 FORJ=1TO1200:NEXT
70 CLS:INPUT"DO YOU NEED INSTRUCTIONS (YES/NO) ";Z$
80 IFZ$="NO"GOTO540
90 CLS:PRINT"YOU HAVE BEEN HIRED AS PRESIDENT OF AN ELECTRIC COMPANY. "
100 PRINT"YOU WILL BE PRESIDENT FOR 10 YEARS BEFORE BEING NAMED TO THE"
110 PRINT"BOARD OF DIRECTORS--ASSUMING A SUCCESSFUL TERM AS PRESIDENT. "
120 PRINT"     YOU WILL START WITH ONE MILLION DOLLARS IN CASH, 1 POWER"
130 PRINT"PLANT AND STOCK PILES OF COAL AND FUEL OIL--THE TWO FUELS YOU"
140 PRINT"MAY USE TO POWER A GENERATING PLANT.  YOU MAY CHOOSE EITHER"
150 PRINT"FUEL YOU WISH TO POWER THE PLANT YOU START WITH AND EACH"
160 PRINT"SUCCEEDING PLANT YOU BUILD THEREAFTER.  ALL PLANTS HAVE A"
170 PRINT"GENERATING CAPACITY OF 1 MILLION KILOWATTS.  AS THE PLANTS"
180 PRINT"BECOME OLDER, THEY BECOME INCREASINGLY INEFFICIENT.  THAT IS,"
190 PRINT"THEY REQUIRE MORE FUEL TO PRODUCE EACH KILOWATT OF ELECTRICITY. "
200 PRINT"THEY CAN BE BROUGHT BACK TO PEAK EFFICIENCY BY REMODELING THEM. "
210 PRINT"THE LONGER YOU WAIT TO REMODEL, THE MORE IT WILL COST.  AND IF"
220 PRINT"MORE THAN ONE PLANT IS OWNED, ALL PLANTS MUST BE REMODELED AT"
230 PRINT"THE SAME TIME. "
240 INPUT"PRESS 'ENTER' TO CONTINUE INSTRUCTIONS ";Z$
250 CLS:PRINT"YOU HAVE TWO FUELS FROM WHICH TO CHOOSE: COAL AND FUEL OIL. "
260 PRINT"COAL TENDS TO YIELD MORE KILOWATTS PER DOLLAR OF FUEL; HOWEVER, "
270 PRINT"COAL PLANTS COST ABOUT 50% MORE TO BUILD THAN FUEL OIL PLANTS. "
280 PRINT"     MANAGEMENT OF FUEL SUPPLIES IS IMPORTANT BECAUSE THERE IS"
290 PRINT"A LEAD TIME OF 1 YEAR BEFORE IT ARRIVES.  COAL ORDERED IN YEAR"
300 PRINT"4 IS NOT AVAILABLE FOR USE UNTIL YEAR 5, SO A CLOSE EYE MUST BE"
310 PRINT"KEPT ON FUEL RESERVES. "
320 PRINT"     INFLATION AFFECTS THE COST OF ALMOST EVERYTHING YOU"
330 PRINT"ENCOUNTER, FROM FUEL COSTS TO DAMAGE CAUSED BY DISASTER OR"
340 PRINT"JUDGEMENT ERROR.  IF YOU FIND THAT A RATE INCREASE IS REQUIRED"
350 PRINT"TO MAINTAIN A PROFITABLE BUSINESS AND KEEP THE STOCK HOLDERS"
360 PRINT"HAPPY, YOU MAY APPLY FOR A RATE INCREASE EACH YEAR.  ALL YOU DO"
370 PRINT"IS SPECIFY WHAT YOU WANT THE NEW RATE TO BE (THE INITIAL RATE"
380 PRINT"IS .05 PER KILOWATT). "
390 PRINT:INPUT"PRESS 'ENTER' TO CONTINUE INSTRUCTIONS ";Z$
400 CLS:PRINT"     TEMPER YOUR RATE REQUESTS WITH MODERATION.  GREED WILL"
410 PRINT"GET YOU NOWHERE.  RATE INCREASES CAN BE REJECTED AS WELL AS"
420 PRINT"ACCEPTED.  BY WAITING LONGER BETWEEN INCREASE REQUESTS, YOU"
430 PRINT"CAN ASK FOR A LARGER INCREASE AND HAVE IT ACCEPTED.  ALSO BY"
440 PRINT"WAITING LONGER BETWEEN REQUESTS, A COMPROMISE RATE INCREAS MAY"
450 PRINT"BE GRANTED--WHERE AS FREQUENT REQUESTS MAY RESULT IN A DENIED"
460 PRINT"INCREASE.  AS WITH FUEL ORDERS, THERE IS A LEAD TIME OF 1 YEAR"
470 PRINT"FOR RATE HIKES.  A RATE INCREASE GRANTED IN YEAR 6 DOES NOT"
480 PRINT"TAKE AFFECT UNTIL YEAR 7. "
490 PRINT:PRINT"     NOW LIKE ALL GOOD EXECUTIVES, YOU MUST GET TO WORK. "
500 PRINT"              G O O D    L U C K ! ! !"
510 DEFDBLB-D,K-L
520 DEFDBLB-D,K-L
530 PRINT:PRINT:INPUT"PRESS 'ENTER' TO BEGIN ";Z$
540 YR=0:PI=1:GF=1:CS=700:CC=20:CF=25:CL=1600:FO=2500:CR#=1000000:R=.05:RT=
550 CLS:PRINT"WHAT FUEL DO YOU WANT TO POWER THE GENERATORS OF YOUR PLANT?"
560 INPUT"1=COAL   2=FUEL OIL";F(1)
570 IFF(1)<1ORF(1)>2GOTO550
580 EF(1)=1
590 YR=YR+1
600 I=0
610 E=(100+RND(10))/100
620 I=I+1:EF(I)=EF(I)*E:IFI<GFGOTO610
630 CLS:PRINT"   ***** FINANCES FOR BEGINNING OF YEAR ";YR;" *****":PRINT
640 PRINT"GENERATING PLANTS OWNED          ";:PRINTUSINGX$;GF
650 PRINT"NUMBER OF CUSTOMERS              ";:PRINTUSINGX$;CS
660 PRINT"TONS OF COAL IN STOCK            ";:PRINTUSINGX$;CL
670 PRINT"BARRELS OF FUEL OIL IN STOCK     ";:PRINTUSINGX$;FO
680 PRINT"CASH RESERVE                     ";:PRINTUSINGD$;CR#
690 PRINT:INPUT"PRESS 'ENTER' TO BEGIN THE YEAR'S TRANSACTIONS";Z$
700 IFCR#>0GOTO780
710 CLS:PRINT"YOU HAVE BUNGLED THE JOB.  THE COMPANY IS BROKE. "
720 PRINT"YOU HAVE BEEN SENT TO A SIBERIAN SALT MINE TO COOL YOUR HEELS. "
730 PRINT:GOTO2220
740 PRINT"YOU HAVE MISMANAGED YOUR FUEL INVENTORY.  THAT IS TOO BAD, FOR"
750 PRINT"YOU WERE DOING FINE IN OTHER AREAS OF MANAGEMENT.  BUT THE"
760 PRINT"BOARD OF DIRECTORS WERE UNSYMPATHETIC, YOU WERE CANNED. "
770 PRINT:GOTO2220
780 IN=(100+RND(12))/100:PI=PI*IN:PG=(100+RND(12))/100
790 MN=((GF*5000)+(CS*10))*PI:CR#=CR#-MN
800 CS=INT(CS*PG):KD#=CS*1000:GC=1000000*GF
810 CC=INT(2000*PI)/100:SC=INT(100*PI*(RND(30)+20)/10)/100
820 OC=INT(2500*PI)/100:SO=INT(100*PI*(RND(10)+10)/10)/100
830 EV=RND(10):IFEV>5GOTO1000
840 ONEVGOTO850  ,920  ,940  ,960  ,980
850 CA=RND(100):HU=INT((500+RND(500)+(1/CA)*RND(500))*PI)
```

```
860 CLS:PRINTCA;" PEOPLE HAVE REQUESTED TO BE ADDED TO THE CUSTOMER LIST."
870 PRINT"IT WILL COST $";HU;" TO HOOK THE CUSTOMERS TO YOUR SYSTEM."
880 PRINT"DO YOU WISH TO SELL POWER TO THEM?":INPUT"1=YES   2=NO";DC
890 IFDC<1ORDC>2GOTO860
900 IFDC=2GOTO1000
910 CS=CS+CA:GOTO1000
920 CA=RND(100):CLS:PRINTCA;" PEOPLE HAVE DROPPED AS CUSTOMERS"
930 CS=CS-CA:GOTO990
940 SD=(RND(1000)+1000)*PI:PRINT"A STORM CAUSED $";SD;" DAMAGE."
950 CR#=CR#-SD:GOTO990
960 PRINT"PEOPLE PROMISE TO CONSERVE.  DEMAND WILL NOT RISE AS SHARPLY"
970 PRINT"THIS YEAR.":KD#=INT(KD#*.9):GOTO990
980 PRINT"DEMAND WILL RISE FASTER THAN USUAL":KD#=INT(KD#*1.1)
990 FORJ=1TO2000:NEXT
1000 KG=0:NC=0:NF=0:I=0
1010 I=I+1
1020 CLS:PRINTUSINGX$;(KD#-KG);
1030 PRINT" KILOWATTS ARE NEEDED.  YOU HAVE ";GF;" PLANTS."
1040 PRINT"HOW MANY KILOWATTS FROM PLANT ";I;:INPUT" ";K
1050 IFK<=1000000GOTO1070
1060 PRINT"ONE PLANT CAN NOT PRODUCE THAT MUCH.":GOTO1040
1070 KG=KG+K
1080 IFF(I)=2GOTO1120
1090 F=INT((K/600*EF(I))+.5):NC=NC+F:IFNC<CLGOTO1110
1100 NC=NC-F:KG=KG-K:PRINT"YOU DO NOT HAVE ENOUGH COAL.":GOTO740
1110 PRINT"PLANT ";I;" NEEDS ";F;" TONS OF COAL.":GOTO1150
1120 F=INT((K/400*EF(I))+.5):NF=NF+F:IFNF<FOGOTO1140
1130 NF=NF-F:KG=KG-K:PRINT"YOU DID NOT HAVE ENOUGH FUEL OIL.":GOTO740
1140 PRINT"PLANT ";I;" NEEDS ";F;" BARRELS OF OIL."
1150 FORJ=1TO1500:NEXT
1160 IFI<GFGOTO1010
1170 IFKD#>GCGOTO1200
1180 IFKD#>KGGOTO1240
1190 GOTO1280
1200 CLS:PRINT"ALL GENERATORS WERE OVERLOADED!  DEMAND WAS GREATER THAN"
1210 PRINT"CAPACITY.  A MAJOR BLACKOUT OCCURRED.  SEVERE DAMAGE HAS BEEN"
1220 PRINT"CAUSED TO THE GENERATORS AND YOUR CAREER.  YOU HAVE BEEN SENT"
1230 PRINT"TO ANTARCTICA TO MANAGE AN ICE CREAM PARLOR.":GOTO2220
1240 DM=(RND(1000)+2000)*PI*KD#/KG:CR=CR-DM
1250 CLS:PRINT"YOU DID NOT GENERATE ENOUGH TO MEET DEMANDS."
1260 PRINT"$";DM;" DOLLARS WORTH OF DAMAGE HAS BEEN CAUSED TO THE"
1270 PRINT"EQUIPMENT.  FORTUNATELY THIS WAS NOT A FATAL MISTAKE."
1280 PRINT"FUEL REQUIREMENTS"
1290 PRINTNC;" TONS OF COAL":PRINTNF;" BARRELS OF FUEL OIL":PRINT
1300 PRINT"COST PER TON OR BARREL";TAB(30);"SHIPPING";TAB(45);"TOTAL COST"
1310 PC=CC+SC:PO=OC+SO
1320 PRINT"COAL";TAB(10);:PRINTUSINGC$;CC;
1330 PRINTTAB(30);:PRINTUSINGC$;SC;
1340 PRINTTAB(50);:PRINTUSINGC$;PC
1350 PRINT"FUEL OIL";TAB(10);:PRINTUSINGC$;OC;
1360 PRINTTAB(30);:PRINTUSINGC$;SO;
1370 PRINTTAB(50);:PRINTUSINGC$;PO
1380 INPUT"HOW MANY TONS OF COAL DO YOU WISH TO BUY ";CP
1390 INPUT"HOW MANY BARRELS OF FUEL OIL DO YOU WISH TO BUY ";OP
1400 CB=CP*PC:OB=OP*PO
1410 CLS:PRINT"FUEL COSTS";TAB(20);"COAL";TAB(35);"FUEL OIL";TAB(50);"TOTA
1420 PRINTTAB(15);:PRINTUSINGE$;CB;:PRINTTAB(30);:PRINTUSINGE$;OB;
     :PRINTTAB(45);:PRINTUSINGE$;(CB+OB)
1430 ER#=KG*RT:CR#=CR#-CB-OB+ER#:CL=CL+CP-NC:FO=FO+OP-NF
1440 FORJ=1TO2000:NEXT
1450 CLS:PRINT"REMODELING DOES NOT INCREASE THE GENERATING CAPACITY OF A"
1460 PRINT"PLANT.  IT WILL RESTORE ALL PLANTS TO PEAK EFFICIENCY.  ALL"
1470 PRINT"PLANTS MUST BE DONE AT THE SAME TIME.":PRINT
1480 PRINT"PLANT";TAB(20);"COST";TAB(40);"INCREASE IN EFFICIENCY"
1490 RM=0
1500 FORI=1TOGF
1510 RC=5000*EF(I)*PI:SV=((EF(I)-1)/EF(I))*100
1520 RM=RM+RC
1530 PRINTI;TAB(15);:PRINTUSINGE$;RC;:PRINTTAB(50);SV;" %"
1540 NEXT
1550 PRINT"TOTAL COST     ";:PRINTUSINGE$;RM
1560 PRINT:PRINT"DO YOU WISH TO REMODEL?":INPUT"1=YES   2=NO";DC
1570 IFDC<1ORDC>2GOTO1560
1580 IFDC=2GOTO1610
1590 CR#=CR#-RM
1600 FORI=1TOGF:EF(I)=1:NEXT
1610 CM=(RND(5000)+45000)*PI:OM=(RND(3000)+30000)*PI
1620 CLS:PRINT"A COAL GENERATING PLANT WILL COST     ";:PRINTUSINGD$;CM
1630 PRINT"A FUEL OIL GENERATING PLANT WILL COST ";:PRINTUSINGD$;OM
1640 PRINT:PRINT"DO YOU WISH TO BUILD A NEW PLANT?"
1650 PRINT"1=COAL   2=FUEL OIL   3=NEITHER"
1660 INPUT"WHAT IS YOUR DECISION ";DC
1670 IFDC<1ORDC>3GOTO1640
1680 IFDC>1GOTO1700
1690 CR#=CR#-CM:GF=GF+1:F(GF)=1:EF(GF)=1:GOTO1720
1700 IFDC>2GOTO1720
1710 CR#=CR#-OM:GF=GF+1:F(GF)=2:EF(GF)=1
1720 CLS:PRINT"DO YOU WISH TO APPLY FOR A RATE INCREASE?"
1730 INPUT"1=YES   2=NO ";DC
1740 IFDC<1ORDC>2GOTO1720
1750 IFDC=1GOTO1770
1760 GOTO1920
1770 PRINT:PRINT"IT HAS BEEN ";TR;" YEARS SINCE YOU HAD A RAISE."
1780 PRINT"CUSTOMERS ARE PRESENTLY PAYING ";RT;"  PER KILOWATT."
1790 PRINT"HOW MUCH DO YOU WISH TO CHARGE PER KILOWATT?"
1800 INPUT"ENTER REQUEST AS A DECIMAL (.07 OR .075 ETC.)";PR
1810 IFTR<2GOTO1890
1820 IFPR>R*PI1.5ANDTR<3GOTO1890
1830 IFPR<R*PI1.5GOTO1870
1840 RT=RT+((PR-RT)*.8):TR=0
1850 PRINT"YOU ARE ALLOWED TO INCREASE YOUR RATE TO ";RT;"
     PER KILOWATT."
1860 GOTO1910
1870 PRINT"YOUR RATE INCREASE HAS BEEN STUDIED AND ACCEPTED
     AS REQUESTED"
```

```
 COAL GENERATING PLANT WILL COST           $62,720.40
A FUEL OIL GENERATING PLANT WILL COST       $40,719.80
DO YOU WISH TO BUILD A NEW PLANT?  1=COAL  2=FUEL OIL  3=NEITHER  3

DO YOU WISH TO APPLY FOR A RATE INCREASE?  1=YES  2=NO  1
IT HAS BEEN  2  YEARS SINCE YOU HAD A RAISE.
CUSTOMERS ARE PRESENTLY PAYING  .05  PER KILOWATT.
HOW MUCH DO YOU WISH TO CHARGE PER KILOWATT?  .06
YOUR RATE INCREASE HAS BEEN STUDIED AND ACCEPTED AS REQUESTED


      ***** END OF YEAR  3  REPORT *****
INCOME FROM SALE OF POWER                $40,679.95
MAINTAINENCE COSTS                       $30,427.50
YOU ARE GENERATING AT   27.12 % OF CAPACITY
THE INFLATION RATE FOR THE YEAR WAS      11 %
THE CONSUMER PRICE INDEX NOW STANDS AT   1.31835


+--++--++--++--++--++--++--++--+

 ***** FINANCES FOR BEGINNING OF YEAR  4  *****
GENERATING PLANTS OWNED                      3
NUMBER OF CUSTOMERS                        904
TONS OF COAL IN STOCK                    1,685
BARRELS OF FUEL OIL IN STOCK             3,113
CASH RESERVE                          $813,676.90

     976,000 KILOWATTS ARE NEEDED.  YOU HAVE  3  PLANTS.
HOW MANY KILOWATTS FROM PLANT  1  900000
PLANT  1  NEEDS  1650  TONS OF COAL.

      76,000 KILOWATTS ARE NEEDED.  YOU HAVE  3  PLANTS.
HOW MANY KILOWATTS FROM PLANT  2  50000
PLANT  2  NEEDS  136  BARRELS OF OIL.

      26,000 KILOWATTS ARE NEEDED.  YOU HAVE  3  PLANTS.
HOW MANY KILOWATTS FROM PLANT  3  26000
PLANT  3  NEEDS  68  BARRELS OF OIL.
FUEL REQUIREMENTS
 1650  TONS OF COAL
 204  BARRELS OF FUEL OIL
COST PER TON OR BARREL        SHIPPING       TOTAL COST
COAL       $27.42               $5.89             $33.31
FUEL OIL   $34.27               $2.60             $36.87

HOW MANY TONS OF COAL DO YOU WISH TO BUY?  1000
HOW MANY BARRELS OF FUEL OIL DO YOU WISH TO BUY?  500

FUEL COSTS          COAL          FUEL OIL       TOTAL
                 $33,310.00     $18,435.00     $51,745.00
```

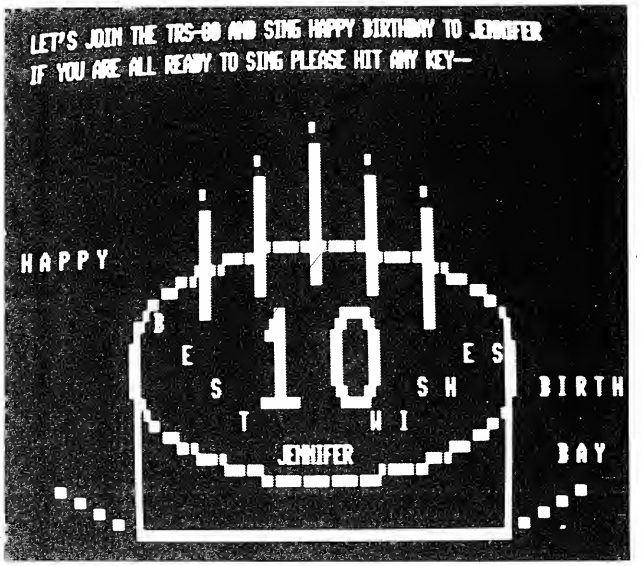
```
REMODELING DOES NOT INCREASE THE GENERATING CAPACITY OF A
PLANT.  IT WILL RESTORE ALL PLANTS TO PEAK EFFICIENCY.  ALL
PLANTS MUST BE DONE AT THE SAME TIME.
PLANT              COST                  INCREASE IN EFFICIENCY
 1               $7,540.94                        9.09091  %
 2               $7,472.39                        8.25688  %
 3               $7,129.62                        3.84615  %
TOTAL COST      $22,143.00
DO YOU WISH TO REMODEL?  1=YES  2=NO  2

AL GENERATING PLANT WILL COST          $68,067.30
A FUEL OIL GENERATING PLANT WILL COST      $41,763.10
DO YOU WISH TO BUILD A NEW PLANT?  1=COAL  2=FUEL OIL  3=NEITHER  3

DO YOU WISH TO APPLY FOR A RATE INCREASE?  1=YES  2=NO  2

     ***** END OF YEAR  4  REPORT *****
INCOME FROM SALE OF POWER              $58,560.00
MAINTAINENCE COSTS                     $32,960.80
YOU ARE GENERATING AT   32.5333 % OF CAPACITY
THE INFLATION RATE FOR THE YEAR WAS      4 %
THE CONSUMER PRICE INDEX NOW STANDS AT   1.37108


--++--++--++--++--++--++--++--+

   ***** FINANCES FOR BEGINNING OF YEAR  5  *****
GENERATING PLANTS OWNED                     3
NUMBER OF CUSTOMERS                       976
TONS OF COAL IN STOCK                   1,035
BARRELS OF FUEL OIL IN STOCK            3,409
CASH RESERVE                         $787,531.12

    1,054,000 KILOWATTS ARE NEEDED.  YOU HAVE  3  PLANTS.
HOW MANY KILOWATTS FROM PLANT  1  900000
YOU DO NOT HAVE ENOUGH COAL.
YOU HAVE MISMANAGED YOUR FUEL INVENTORY.  THAT IS TOO BAD, FOR
YOU WERE DOING FINE IN OTHER AREAS OF MANAGEMENT.  BUT THE
BOARD OF DIRECTORS WERE UNSYMPATHETIC, YOU WERE CANNED.
```

Chapter VII
Graphics & Music

A Cake For Your TRS-80

Birthday Celebration

Vida Harper



The next time someone in your family or a friend has a birthday, why not have your TRS-80 join in the celebration? The following program combines graphics and music to produce a unique display which will delight old and young.

The graphics include a cake decorated with best wishes, blinking candles, the person's age, and the name of the one who is celebrating. For an added extra, the computer plays the age-old favorites "Happy Birthday" and "How Old Are You?"

The numbers on the cake are constructed by using the 64 possible TRS-80 graphics characters plus the standard ASCII Control Codes. Each one is a block of graphics which are three characters high and six characters wide. By drawing the numbers on a graphics worksheet, one can easily see similarities in the ten digits. In line 370 of the program you will notice that the top third of the 0, 2, 8, and 9 are identical. Since each digit is a block of 18 graphics, it will be an advantage if you recognize similiarities and then define these strings, thus resulting in short cuts. Lines 350 to 550 are the concatenating of the strings which result in producing these ten digits. (Notice that the numerals are named R (0) through R (9) for ease in calling these from memory for display purposes.) In lines 570, 580 and 590 the numerals are placed on the cake with PRINT@ statements. The three positions used depends on whether the birthday kid is one digit, two digits, or three digits (???) old. The blinking of the candles is done by using graphic blocks 128 and 132, and timer loops combined with subroutines.

Now an explanation about the music in the last part of the program: The original tone generator program used here was published in the Nov. 1978 issue of *TRS-80 Users Group Newsletter* of Fayetteville

Vida Harper, 1807 Shore Dr., Holland, MI 49423.

```
100 REM *  *                                          *  *
110 REM *  *          BIRTHDAY CELEBRATION            *  *
120 REM *  *             BY VIDA A. HARPER            *  *
130 REM *  *                JUNE  1980                *  *
140 REM *  *          WRITTEN  FOR  TRS-80            *  *
150 REM *  *             Level II 16K & 32K           *  *
160 REM *  *                                          *  *
170 CLS
180 CLEAR 1000
190 DEFSTR Q,R,S
200 INPUT"DO YOU HAVE YOUR SPEAKER CONNECTED  (Y/N) -- ";QA
210 IF QA ="Y" THEN240
220 PRINT"IF YOU DO NOT HAVE A SPEAKER THEN PLACE AN AM RADIO"
230 PRINT"NEAR THE KEYBOARD SO YOU CAN HEAR THE MUSIC";
    :FORX=1TO2000:NEXTX
240 CLS:PRINT@384,"   B I R T H D A Y     C E L E B R A T I O N"
    :FOR X=1TO500:NEXT
250 PRINT:PRINT
260 INPUT"PLEASE GIVE THE FIRST NAME OF THE BIRTHDAY KID";Q7
270 INPUT"GIVE THE AGE OF THE ONE WHO IS CELEBRATING";Q8
280 L=LEN(Q8):IFL=1THENB=VAL(Q8)
290 IFL=2:C=VAL(LEFT$(Q8,1)):D=VAL(RIGHT$(Q8,1))
300 IFL=3:E=VAL(LEFT$(Q8,1)):F=VAL(MID$(Q8,2,1)):G=VAL(RIGHT$(Q8,1))
310 CLS
320 IFL>3THENPRINT@64,"  Y O U     M U S T     B E     K I D D I N G ! ! ! "
    :FOR X=1TO500:NEXT:GOTO270
330 CLS
340 PRINT@448,"H A P P Y":PRINT@759,"B I R T H":PRINT@889,"D A Y":PRINT@859,Q7
350 ' THE NUMBERS ARE BLOCKS (6 ACROSS AND 3 DOWN) OF GRAPHIC
      CHARACTERS
360 Q1=STRING$(5,24)+CHR$(26)   'BACKSPACE 5 AND DOWN FEED
370 'Q2 IS TOP THIRD OF 0,2,8,&9
380 Q2=CHR$(184)+CHR$(135)+CHR$(131)+CHR$(139)+CHR$(180)
390 'Q3 IS BOTTOM THIRD OF 0,6,& 8
400 Q3=CHR$(139)+CHR$(180)+CHR$(176)+CHR$(184)+CHR$(135)
410 'Q4 IS BOTTOM THIRD OF 3 AND 5
420 Q4=CHR$(172)+STRING$(2,CHR$(176))+CHR$(184)+CHR$(135)
430 'Q5 IS TWO BLANK SPACES, Q6 IS 2 OF CHR 140, AND Q9 ERASES
      TO END OF LINE
440 Q5=STRING$(2,CHR$(128)):Q6=STRING$(2,CHR$(140)):Q9=CHR$(30)
450 'BUILDING THE NUMBERS ON THE CAKE BY USING GRAPHICS
460 R(0)=Q2+Q1+CHR$(191)+STRING$(3,CHR$(128))+CHR$(191)+Q1+Q3
470 R(1)=CHR$(128)+CHR$(184)+CHR$(191)+Q5+Q1+Q5+CHR$(191)+Q5+Q1+CHR$(160)
    +CHR$(176)+CHR$(191)+CHR$(176)+CHR$(144)
480 R(2)= Q2+Q1+CHR$(128)+CHR$(176)+CHR$(156)+CHR$(135)+CHR$(129)+Q1
    +CHR$(190)+CHR$(179)+STRING$(3,CHR$(176))
490 R(3)=CHR$(140)+ STRING$(2,CHR$(131))+CHR$(139)+CHR$(180)+Q1+CHR$(128)
    +Q6+CHR$(166)+CHR$(145)+Q1+Q4
500 R(4)=CHR$(160)+CHR$(158)+CHR$(131)+CHR$(191)+CHR$(128)+Q1+CHR$(143)
    +Q6+CHR$(191)+CHR$(140)+Q1+Q5+CHR$(128)+CHR$(191)
510 R(5)=CHR$(190)+STRING$(3,CHR$(131))+CHR$(129)+Q1+CHR$(139)+Q6
    +CHR$(172)+CHR$(144)+Q1+Q4
520 R(6)= CHR$(128)+CHR$(160)+CHR$(158)+CHR$(131)+CHR$(128)+Q1+CHR$(184)
    +CHR$(143)+Q6+CHR$(176)+Q1+Q3
530 R(7)= CHR$(142)+STRING$(2,CHR$(131))+CHR$(163)+CHR$(159)+Q1+Q5
    +CHR$(184)+CHR$(135)+CHR$(128)+Q1+CHR$(160)+CHR$(158)+CHR$(129)
540 R(8)=Q2+Q1+CHR$(162)+CHR$(153)+CHR$(140)+CHR$(166)+CHR$(145)+Q1+Q3
550 R(9)=Q2+Q1+CHR$(131)+Q6+CHR$(188)+CHR$(135)+Q1+CHR$(128)+CHR$(176)
    +CHR$(158)+CHR$(129)
560 'POSITIONING THE NUMBERS ON THE CAKE ACCORDING TO WHETHER
      THE PERSON IS ONE DIGIT, TWO DIGITS, OR THREE DIGITS (???) OLD
```

NC. Dean McCulloch, the author, uses a machine program to produce a single frequency of short duration. (Mr. McCulloch has given full consent of the use of his tone generator program, which is incorporated into my program.) His original music program of "O Come All Ye Faithful" uses the cassette recorder as the note generator. You first remove the cassette tape and push down the play and record buttons simultaneously, then connect a speaker into the ear plug.

The Birthday Celebration program listed here uses this tone generator program of Mr. McCulloch with a few minor changes. The machine language program was expanded to a 16K location, and to speed up the tempo a .6 factor of TM (time) was inserted in line 1000. Of course the pairs of data in lines 1240 to 1310 had to be changed to coincide with the notes and timing of "Happy Birthday" and "How Old Are You?" The following notes and identified frequencies are suggested for use:

B (215) Middle C (233) D (264)
D# (286) E (297) F (319) F# (330)
G (352) A (396) B (440)
High C (465) D (528) and E (600).

To modify the cassette program for Disk Basic it was necessary to change or add the following lines:

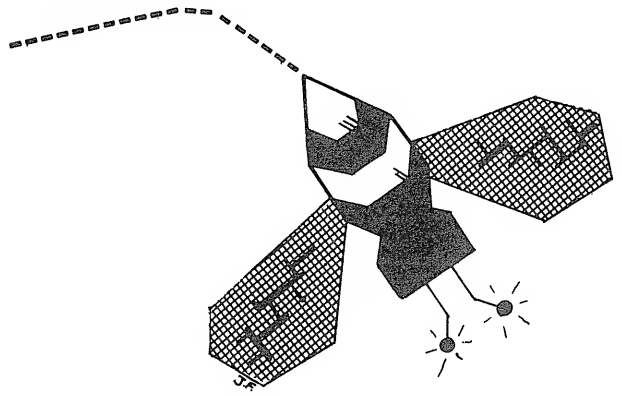
```
870 FOR I = -16528 TO -16490
910 DATA 205,127,10,14,255,6,1,
    205,136,191,6,2,205,136,191
920 DATA  43,62,0,180,181,194,117,
    191,201,237,65,237,91,159,191
935 ON ERROR GOTO 939:DEFUSR=
    -16528:ON ERROR GOTO 0: GOTO
    960
939 ON ERROR GOTO 0: RESUME 940
950 POKE 16527,191
1060 POKE -16481, DE%: POKE
     -16480,D%
```

Improvements are left to the musically minded readers. Lines 960-980 and 1090-1200 were added to the original music program, so that the additional graphics, including blowing out the candles, would be executed between the playing of the two songs.

Here's hoping that your TRS-80 will be invited to attend many birthday celebrations.

Personal Note: The inspiration behind this program was the desire to have something special to help celebrate my father's 90th birthday last fall. Since he claims that he will live to be 100, the program has the capabilities of positioning three digit numbers on the cake. □

```
570 IFL=1THENPRINT@605,R(B)
580 IFL=2THENPRINT@601,R(C):PRINT@609,R(D)
590 IFL=3THENPRINT@599,R(E):PRINT@605,R(F):PRINT@611,R(G)
600 'DECORATING THE CAKE WITH BEST WISHES
610 N=0:F$="BEST":G=LEN(F$):FORZ=1TOG:G$=MID$(F$,Z,1):PRINT@590+N,G$;
    :N=N+67:NEXT
620 N=0:H$="W  IS  HE  S ":I=LEN(H$):FORX=1TOI/4:I$=MID$(H$,4*X-3,4)
    :PRINT@805-N,I$;:N=N+59:NEXT
630 'BUILDING THE SIDES AND BOTTOM OF THE CAKE
640 X=7:Y=43:FORJ=0TO12 STEP4:SET(X+J,Y+J/4):NEXT
650 X=8:Y=43:FORJ=0TO12 STEP4:SET(X+J,Y+J/4):NEXT
660 FOR X=24TO102:SET(X,47):NEXT
670 X=106:Y=46:FOR J=0TO12STEP4:SET(X+J,Y-J/4):NEXT
680 X=107:Y=46:FORJ=0TO12STEP4:SET(X+J,Y-J/4):NEXT
690 FORY=29TO47:SET(24,Y):NEXT:FORY=30TO34:SET(23,Y):NEXT
700 FORY=29TO47:SET(103,Y):NEXT:FORY=30TO34:SET(104,Y):NEXT
710 'TOP OF THE CAKE USING AN ALTERED CIRCLE FORMULA
720 FOR X=-10TO10STEP.3
730 Y=SQR(110-X*X):SET(4*X+64,Y+32):NEXT
740 FOR X=-10TO10 STEP.3
750 Y=-SQR(110-X*X):SET (4*X+64,Y+32):NEXT
760 ' WHAT'S A CAKE WITHOUT CANDLES
770 FORX=38TO39:FORY=18TO27:SET(X,Y):NEXT:NEXT
780 FORX=50TO51:FORY=15TO25:SET(X,Y):NEXT:NEXT
790 FORX=62TO63:FORY=12TO24:SET(X,Y):NEXT:NEXT
800 FOR X=74TO75:FORY=15TO25:SET(X,Y):NEXT:NEXT
810 FOR X=86TO87:FORY=18TO28:SET(X,Y):NEXT:NEXT
820 ' THE GRAPHICS NECESSARY TO BLINK THE CANDLES
       1= ON AND 2=OFF
830 S(1)=CHR$(132):S(2)=CHR$(128)
840 PRINT@339,S(1);:PRINT@281,S(1);:PRINT@223,S(1);:PRINT@293,S(1);
    :PRINT@363,S(1);:FORX=1TO500:NEXT
850 FOR J=1 TO6:GOSUB 1330:NEXT J
860 'ORIGINAL MUSIC PROGRAM (MACHINE 4K-RAM) WRITTEN BY
    DEAN MCCULLOCH AND PUBLISHED IN TRS-80 USERS GROUP
    NEWSLETTER--NOV. 1978 --FOLLOWING PROGRAM IS FOR 16K
870 FORI=32624TO32662     '870-950 LOADS MACHINE PROGRAM & SETS UP USR
880 READ A
890 POKE I,A
900 NEXT I
910 DATA 205,127,10,14,255,6,1,205,136,127,6,2,205,136,127
920 DATA 43,62,0,180,181,194,117,127,201,237,65,237,91,159,127
930 DATA 27,62,0,178,179,194,142,127,201
940 POKE 16526,112
950 POKE 16527,127
960 PRINT@0,"LET'S JOIN THE TRS-80 AND SING HAPPY BIRTHDAY TO ";Q7
970 PRINT@64,"IF YOU ARE ALL READY TO SING PLEASE HIT ANY KEY--"
980 J$=INKEY$:IF J$=""THEN980ELSE1210
990 FOR J= 1 TO D         'FQ IS FREQUENCY--TM IS THE TIME DESIRED
1000 READ FQ,TM:CY=INT(FQ*.6*TM):IF FQ=0ANDTM=0 THEN 1320
1010 IF FQ>5000 THEN FQ=5000       'CHECK FOR FREQUENCY OUT OF RANGE
1020 IF FQ<50 THEN FQ=50
1030 D%=0:DE%=29480/FQ          'FIND AND STORE HALF WAVE TIME DURATION
1040 IF DE%>512 THEN D%=2:DE%=DE%-512
1050 IF DE%>256 THEN D%=D%+1:DE%=DE%-256
1060 POKE 32671,DE%:POKE 32672,D%
1070 X=USR(CY)            'CALL MACHINE PROGRAM
1080 NEXT J
1090 PRINT@0,"MAKE A WISH ";Q7;" AND BLOW OUT YOUR CANDLES!!! ";:PRINTQ9;
1100 PRINT@64,Q9
1110 FOR J= 1 TO 3:GOSUB 1330:NEXTJ
1120 PRINT@64,"OH COME ON YOU CAN BLOW HARDER THAN THAT!!!";:PRINTQ9
1130 FORJ=1TO2:GOSUB1330:NEXTJ
1140 PRINT@339,S(2);:PRINT@281,S(2);:PRINT@223,S(2);:PRINT@293,S(2);
     :PRINT@363,S(2);
1150 PRINT@128,"CONGRATULATIONS YOU WILL GET YOUR WISH!!!";
1160 FORJ=1TO1000:NEXTJ
1170 PRINT@0,"HOW ABOUT A FINAL CHORUS OF --- HOW OLD ARE YOU---";:PRINTQ9;
1180 PRINT@64,"HIT ANY KEY WHEN THE GANG IS READY TO SING!!!"
1190 PRINT@192,"HAVE A GREAT BIRTHDAY--";Q7;
1200 J$=INKEY$:IFJ$=""THEN1200ELSE1220
1210 AD=39:D=25:RESTORE:GOTO 1230
1220 AD=89:D=23:RESTORE
1230 FORW=1TOAD:READA:NEXTW:GOTO990
1240 DATA 264,.5,264,.5,297,.5,264,.5,352,.5,330,1
1250 DATA 264,.5,264,.5,297,.5,264,.5,396,.5,352,1
1260 DATA 264,.5,264,.5,528,.5,440,.5,352,.5,330,.5,297,1
1270 DATA 465,.5,465,.5,440,.5,352,.5,396,.5,352,2
1280 DATA 264,.5,297,.5,264,.5,352,.5,330,1
1290 DATA 264,.5,297,.5,264,.5,396,.5,352,1
1300 DATA 264,.5,528,.5,440,.5,352,.5,330,.5,297,1
1310 DATA 465,.6,440,.6,352,.6,396,.6,352,2,0,0
1320 FOR J=1 TO 1000:NEXTJ:RESTORE:GOTO 280
1330 PRINT@339,S(2);:PRINT@223,S(2);:PRINT@363,S(2);:FORX=1TO8:NEXT
1340 PRINT@339,S(1);:PRINT@223,S(1);:PRINT@363,S(1);:FORX=1TO600:NEXT
1350 PRINT@281,S(2);:PRINT@293,S(2);:FORX=1TO8
1360 PRINT@281,S(1);:PRINT@293,S(1);:FORX=1TO600:NEXT:RETURN
```



Can you find the shortest path?

Bee Amazed

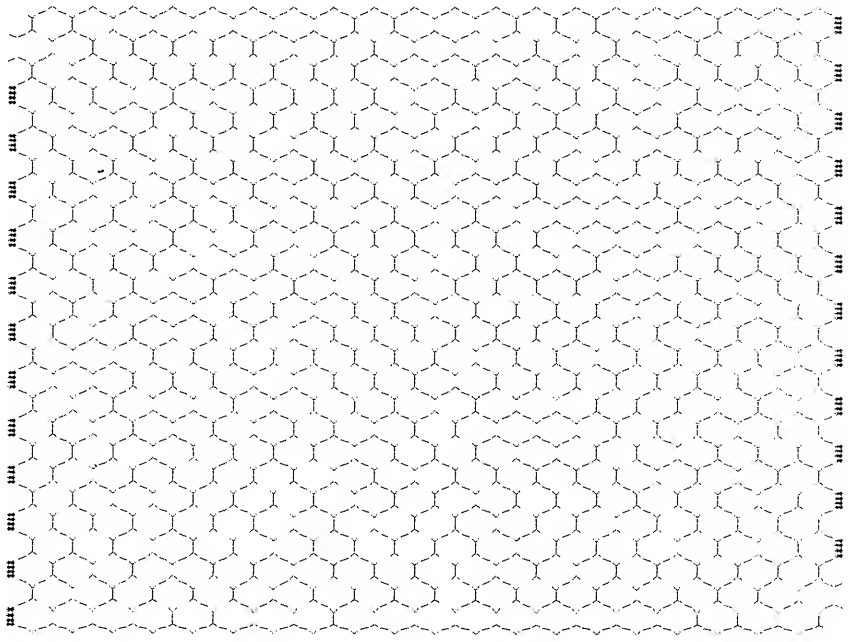
Dan Rollins

There is a certain fascination in the way a computer can create and solve a maze. This article explains the algorithms I used in Beemaze, a program which generates and finds the shortest path through a maze of hexagonal cells.

The program is written in TRS-80 Level 2 Basic, and parts of it use the TRS-80 graphics. However, all the routines used in creating the printout are compatible with similar Basics.

For TRS-80 users, the program provides a special treat. The maze is created graphically on the screen. When the maze is completed, you'll watch as the bee searches for the shortest route. Then man is pitted against insect in a race through the maze. If you choose the print output option, a much larger maze may be created. A 16K machine will handle a maze with 2700 elements. My 48K disk system turned out a maze with 6500 elements—a printout over ten feet long!

There are two options for the printer output. You may make copies of the maze with or without the solution. Distribute the unsolved versions among your friends, and when they claim that the monster is unsolvable, give them a brief peek at the answer.



Example 1a. Printout without solution.

Generating the Maze

The maze is represented in computer memory as a two-dimensional integer array. Each array element (or cell) is defined by a numerical value of 0 to 63 in the following manner. Every SIDE of a cell has been assigned a number (see Figure 1a). The overall value of a cell is determined by the sum of the values for all sides of the cell which have a door. For example, a cell with no doors is a zero cell. A cell with doors valued 1 and 16 will be given a value of 17. If a cell has three doors, e.g. 1, 4, and 16, it is valued at 21.

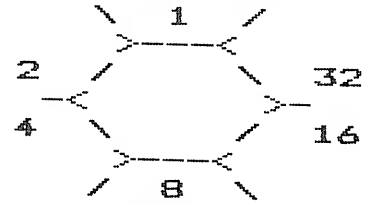


Figure 1a. Values representing the doors of each cell. Cells with two or more doors will be the sum of the respective doors.

Dan Rollins, 370 N. Cerritos, Apt. 15A, Azusa, CA 91702.

These numbers correspond to bit positions in a byte of memory. (More on that in a minute.) The variables X and Y are pointers to a vertical column and horizontal row respectively. Initialized randomly, these pointers are manipulated within the array, moving from cell to cell.

Note a peculiar property of this hexagonal array representation. In a normal array of "square" cells, pointers are manipulated north or south by simply adjusting the Y variable. East and west motion is performed by bumping the X pointer. But each cell of this array neighbors *six* other cells and motion is possible in as many directions (N,S,NE, NW,SE,SW). Examine Figure 2. Motion

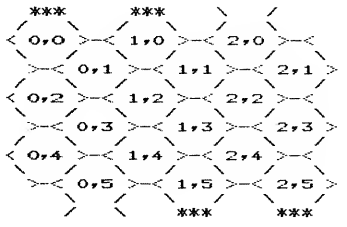


Figure 2. How the cells are numbered in the two-dimensional integer array.

north is accomplished by subtracting 2 from Y. Likewise, 2 is added to Y for motion south. However, to adjust the pointers in a diagonal motion (NE,NW, SE,SW), special attention must be paid to the current Y coordinate. For example, when Y is an even number (evenly divisible by 2), then moving SE requires that Y be incremented and X be left alone. If Y is an odd number, the same motion requires that both X and Y be incremented.

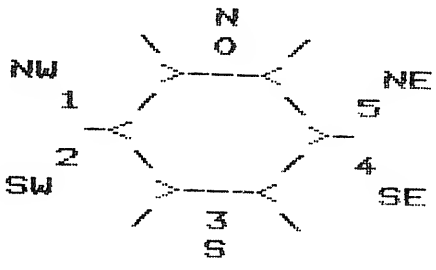


Figure 1b. Possible directions for the bee to turn in the maze solution algorithm.

The variable Z is used to account for this variation. Several lines in the program evaluate Z as -((Y AND 1)=1). TRS-80 Basic will return Z=0 when Y is even and Z=1 when Y is odd. Your Basic may require something like:

```
IF Y/2=INT(Y/2) THEN Z=0 ELSE Z=1
```

The direction of motion is determined in lines 120-200. Six neighboring cells are checked for a zero (doorless) condition. A short list, held in T(Q), is compiled from the possible directions of travel (see Figure 1b). One of these directions is chosen randomly, and the ON..GOTO command sends control to the line for the correct action. Here, the current cell is updated with a value indicating a door in that direction and the X,Y variables are adjusted to point to the new cell. That cell is then given the value for its new door and, if the game option was chosen, the screen is updated.

When the program finds itself trapped, e.g. when none of the neighboring cells is still zero, the pointers are moved down and across the array until a doorless cell is located which is adjacent to a cell with at least one door. Thus all branches are linked—there are no "islands" of unconnected cells in the completed maze. Any individual cell may be reached from any other. Indeed, though I chose to place them at opposite corners, the entrace and exit may be randomly created after the entire maze is completed. Another advantage is that the solution, the direct path from entrance to exit, will be unique.

Solving the Maze

When the IEEE (Institute of Electrical and Electronic Engineers) holds its Micromouse Maze contest, tiny mechanical robots compete in finding their way through a maze. They are given three tries to obtain the best possible time. The maze they run may have many solutions, only one of which is the fastest. One method of finding this path is called the "leftmost/rightmost" algorithm. On the first traversal, whenever an opening to the left is sensed, the "mouse" rotates and follows that path. On the second run, doors to the right are taken. Coordinates of each position are saved in on-board memory. After the two runs, the paths are compared for common intersections. Superfluous loops are eliminated and the shortest path is computed from the stored data. Some other factors are considered such as rotating speed—a longer, straighter path may be quicker than a path shorter in distance but having many turns. Finally, the optimized third run is made.

The "bee" in this program uses a similar system. The algorithm is simplifed by the fact that a single walk-through will determine the best route. It must be extended, however, to allow for the six possible exits from each cell. In this "clockwise" algorithm, the correct cell exit will be the first door clockwise of the door used as the entrance.

Consider the course of action as the bee traverses the maze. It enters a cell, records the coordinates, and begins rotating in clockwise direction. The first opening it sees determines the direction of travel and it moves in that direction into the adjacent cell. The bee moves from cell to cell until it reaches a cul-de-sac (dead end). In that case, rotation brings it back to the door thouh which it entered. The next cell it enters is the one it just vacated. Now the bee is facing in a direction *different* from it previous visit. Rotation brings it to another exit, possibly the one by which it originally entered the cell, but never back into the cul-de-sac. The bee searches through its memory and notes that it has been there before. It then deletes the cul-de-sac from its best-path list and continues until the exit is reached.

Think of the X,Y pointers in the program as the bee. Its current direction is kept in the variable D. As the bee enters a cell, the last door it should check is the one directly behind it. If it entered by going south then that door is now to its north. The program reinitializes the direction by subtracting 3 from D, then adjusting for underflow (adding 6 if the result is less than zero). The actual rotation is performed by repeatedly decrementing D (again adjusting for underflow). At every turn, the cell is tested for a door in the current direction.

Refering back to Figures 1a and 1b, notice the relationship between the values of a direction and a door in that direction. Each door corresponds to the power of 2 defined by the direction. This system is set up to use the powerful bit-manipulation of the AND command. When two numbers are ANDed they are compared on a bit by bit basis. The result, shown in Figure 3, is a value composed of only the ON bits common to both.

decimal	binary	doors
19 =	00010011 =	xxxSExxNWN
AND 16 =	00010000 =	xxxSExx x x
result 16=	00010000	SE

Figure 3.

ANDing two numbers which have no ON bits in common will return a zero as the result. The program lines asking ....IF var1 AND var2 THEN...are checking for common bits. The IF fails and excution falls through when the logical AND results in a non-zero value.

When the maze was created, certain bits of each cell were turned ON, defining the doors. Now, by testing the state of a

given bit, the maze runner can sense the presence of a door. The program computes 2 ↑ D, then ANDs this value with the value of the cell. A non-zero result indicates that there is indeed a door so the bee, taking that direction, enters the new cell.

The real value of this sytem is not only the ease with which the maze data is handled, but in the compactness of that data. A single integer defines both the number of doors and their locations. Also, a cell may take on additional aspects just by turning on new bits. After the best route has been determined, the cells along that path are increased by 64. The printer routine senses this for marking the solution. By adding 128, a new factor could be introduced to certain cells. Perhaps a queen bee or some honey might be found there.

The maze runner algorithm always yields a list of the coordinates which make up the shortest route from the start of the maze to the end. But as described, it's too slow. For example, a cul-de-sac would require six complete turns to locate the exit. There are a couple of shortcuts I used to speed the bee along.

First, that cul-de-sac may be deduced by the fact that the value of the cell is exactly equal to that of the door by which it entered. No sense in rotating here, so the bee simply reverses direction and exits.

Secondly, a test is made for a two-doored cell. The door is subtracted from the cell. If the difference is exactly a power of 2 then there is only one other exit and it is determined as in line 670.

Only when there are more than two doors (the least likely occurrence) is the clockwise algorithm invoked.

All three possibilities return a new direction in the variable D. This value is used in the ON...GOTO sequence to adjust the X,Y array pointers to the next cell. Execution resumes by saving the new position and repeating the action of finding the maze exit.

The multiple-door cells also triggers the routine which updates the best-route list. It is here that the selective "forgetting" of deadend paths is performed. A FOR... NEXT loop checks backwards through the best-route list comparing each saved coordinate against the current one. If a match is found then the best-path list counter, PTR, is set to the loop counter and the loop is exited. Coordinates of the next cells visited will be written over the superfluous steps, in effect erasing them from memory.

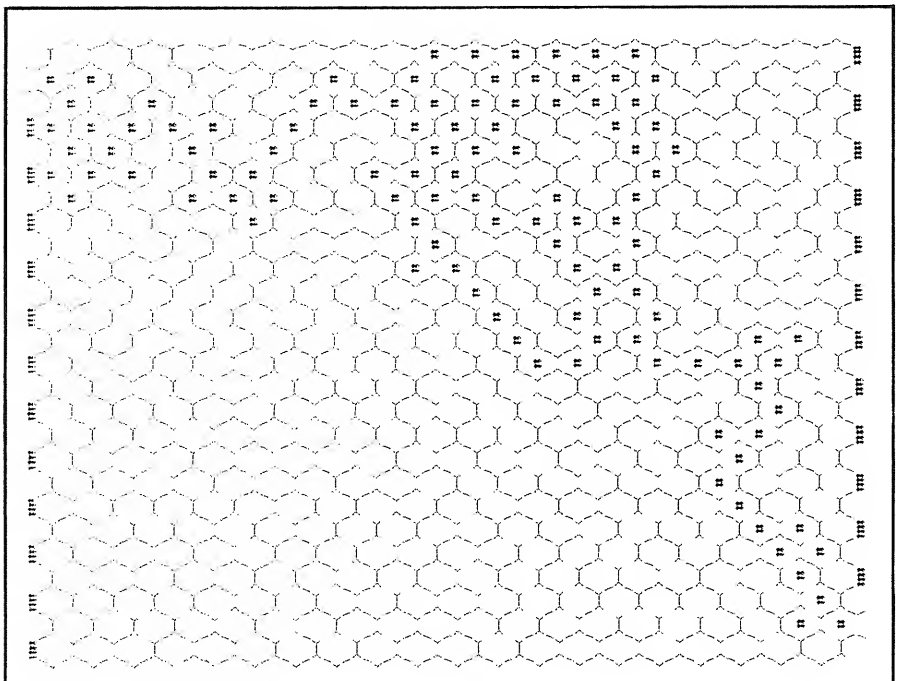
The Printout

The printer output routine is designed for printer portability. Lines 830-900 define the character strings which correspond to the walls and doors of the maze. I used FS$ and BS$ in lines 880-890 because my TRS-80 has no key for the backslash character. If your printer has graphics capabilities, it will be easy to experiment with different combinations, as they need be typed at only one place.

Lines 940-950 print the top output line. Lines 960-1060 examine the body of the array, printing four lines for two vertical coordinates. The last three lines (the bottom cell) are printed in lines 1070-1130.

The examples I've included were printed on an IDS 440 Paper Tiger set at 132 characters per line and 8 lines per inch. These settings allow a horizontal dimension of 16 with a height of 40 to be fit on a single page.

The best-route is represented by cells containing "**." These characters will never be printed until the routine at lines 770-810 has been called. Once this is done, all further printouts will have the solution.



Example 1b. Printout with solution.

The Race

Game players of all ages will enjoy the challenge of racing the bee through the maze. Once the maze is generated, the screen is saved in a string array. Note that four bytes of the 1024 in the TRS-80 screen memory are ignored as a string can only be 255 bytes long. When the screen is redrawn (line 2480) one space is inserted between each string to account for this.

After a short countdown, the screen is restored and the race begins. The bee starts in the top right corner and you are at the bottom left corner. Line 1330 reads the byte of memory at address 14400. This returns a value for the arrow and control keys. Lines 1340-1380 decode this into an adjustment to your present position. This routine is superior to an INKEY$ function as the keybord is constantly being strobed. As long as a key is depressed (and the action doesn't result in crossing a wall) the dot continues moving. Pressing horizontal and vertical keys simultaneously results in diagonal motion.

While the bee makes moves directly from cell to cell, you must work your way one step at a time. The bee's advantage is offset by the fact that it must perform a little dance in each cell. You get five moves for each one made by the bee. This ratio makes the race very close. You probably won't win unless you make every move count—including diagonal motion where possible. Since the mazes are created randomly, some will be more difficult than others. If you find that you always lose (or win!) try changing Line 1420 to increase or decrease the number of dance steps per cell. Also the size of the maze may be decreased by changing H and V in line 2100.

Regardless of whether you win or lose,

you have the option of running the same maze again.

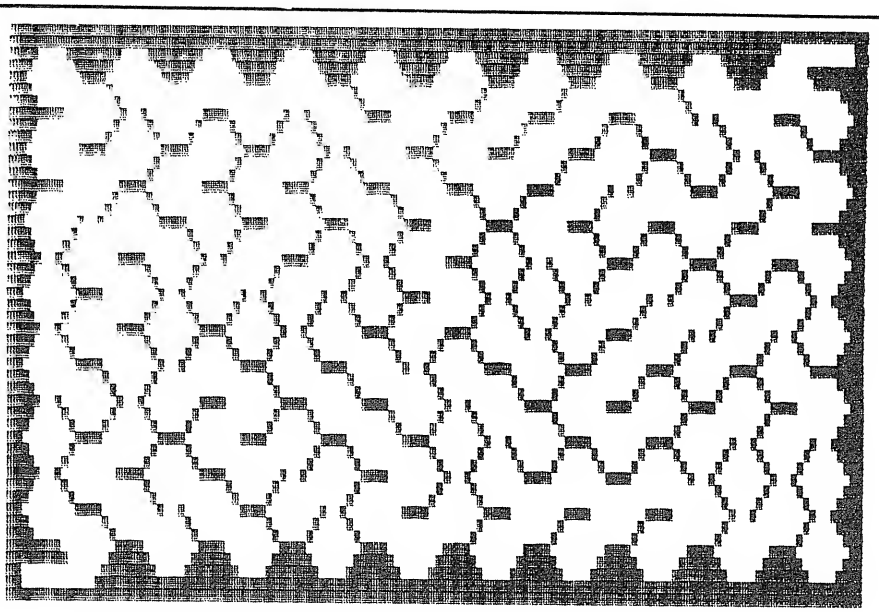
Beemaze has been optimized for speed. "Dummy" FOR...NEXT loops are used to avoid "backward GOTOs." An example is in the K-loop of lines 530-570. This variable is never meant to reach MAX and the value of the loop counter is never used. The advantage is that when NEXT is reached, the correct line number is simply POPped off the stack. A GOTO directive requires that the ENTIRE program be searched for a match with (Line #). The end result is that Basic has much less work to do to find its new position and the program executes faster.

All REMarks are expendable and should be removed. I have indented the loops and added liberal spaces to make the listing more readable. They also slow the program down and take extra bytes. You may omit these when you key the program.

If you don't have a TRS-80 or don't intend to play the game, the lines containing the variable S1 are unnecessary as are lines 1150-1450 and 1280-2590.

It took less than seven minutes to generate, solve and print the mazes in example 1 and 2. My ten-foot monster required a total of 2 hours and 23 minutes to complete. The average time for creating and solving the game maze on the screen (9 by 13) is about three minutes.

Mazes are more than fun and games. The algorithms presented here might be extended to control and guard robot or an automated floor-sweeper. If nothing else, just learning these techniques of array manipulation and bit-testing can be very worth your while. □



Example 2. Completed game maze. The bee will find the only direct path from entrance to exit.

```
10 'PROGRAM ID   :       BEEMAZE
   PROGRAMMER    :       DAN ROLLINS
   DATE          :       1/20/80
20 '
   THIS PROGRAM CREATES AND SOLVES A MAZE OF HEXAGONAL CELLS.
   IT INCLUDES OPTIONS FOR LINEPRINTER OR SCREEN OUTPUT.
   LINEPRINT THE MAZE WITH OR WITHOUT THE SOLUTION.
   THE SCREEN OUTPUT INCLUDES A RACE THROUGH THE MAZE.

30 CLEAR 1200 :DEFINT B-Z
40 GOTO 2000
90 '
    *** MAZE GENERATION ROUTINE ***
100 DIM M(H,V),B((H+1)*(V+1))   '* MAZE ARRAY & BEST-ROUTE LIST
110 C=(H+1)*(V+1) :X=0 :Y=RND(V) :MAX=32765
120 FOR J=1 TO C-1
130   FOR K=0 TO MAX
140     Q=0 :Z=-((Y AND 1)=1) :ZX=X+Z
150     IF Y>1 IF M(X,Y-2)=0 THEN Q=Q+1 :T(Q)=0              '*N
160     IF Y>0 IF ZX>0 IF M(ZX-1,Y-1)=0 THEN Q=Q+1 :T(Q)=1 '*NW
170     IF Y<V IF ZX>0 IF M(ZX-1,Y+1)=0 THEN Q=Q+1 :T(Q)=2 '*SW
180     IF Y<V-1 IF M(X,Y+2)=0 THEN Q=Q+1 :T(Q)=3           '*S
190     IF Y<V IF ZX<H+1 IF M(ZX,Y+1)=0 THEN Q=Q+1 :T(Q)=4 '*SE
200     IF Y>0 IF ZX<H+1 IF M(ZX,Y-1)=0 THEN Q=Q+1 :T(Q)=5 '*NE
210     IF Q>0  THEN K=MAX :GOTO 270
220     '** NO MOVE SO SCAN FOR OPEN CELL **
230     FOR L=0 TO MAX
240       Y=Y+1 :IF Y>V THEN Y=0 :X=X+1 :IF X>H THEN X=0
250       IF M(X,Y)>0 THEN L=MAX
260     NEXT L
270   NEXT K
280   IF S1 THEN X1=X*12+Z*6+3 :Y1=Y*3+2
290   '** ADJUST POINTERS, UPDATE ARRAY AND SCREEN **

300   ON T(RND(Q))+1 GOTO 310,340,370,400,430,460
310   M(X,Y)=M(X,Y)+1 :Y=Y-2 :M(X,Y)=M(X,Y)+8              '* N
320   IF S1 THEN RESET(X1+2,Y1-1) :RESET(X1+3,Y1-1)
330 NEXT J :RETURN
340   M(X,Y)=M(X,Y)+2 :X=ZX-1 :Y=Y-1 :M(X,Y)=M(X,Y)+16     '* NW
350   IF S1 THEN RESET(X1,Y1) :RESET(X1-1,Y1+1)
360 NEXT J :RETURN
370   M(X,Y)=M(X,Y)+4 :X=ZX-1 :Y=Y+1 :M(X,Y)=M(X,Y)+32     '* SW
380   IF S1 THEN RESET(X1,Y1+4) :RESET(X1-1,Y1+3)
390 NEXT J :RETURN
400   M(X,Y)=M(X,Y)+8 :Y=Y+2 :M(X,Y)=M(X,Y)+1              '* S
410   IF S1 THEN RESET(X1+2,Y1+5) :RESET(X1+3,Y1+5)
420 NEXT J :RETURN
430   M(X,Y)=M(X,Y)+16 :X=ZX :Y=Y+1 :M(X,Y)=M(X,Y)+2       '* SE
440   IF S1 THEN RESET(X1+5,Y1+4) :RESET(X1+6,Y1+3)
450 NEXT J :RETURN
460   M(X,Y)=M(X,Y)+32 :X=ZX :Y=Y-1 :M(X,Y)=M(X,Y)+4       '* NE
470   IF S1 THEN RESET(X1+5,Y1) :RESET(X1+6,Y1+1)
480 NEXT J :RETURN
490 '
    *** BUZZ THE MAZE FOR SOLUTION ***
500 A$=STRING$(2,173) :B$=STRING$(2,158)
510 Y=V :X=0 :Z=1 :PTR=0 :B(0)=20*Y :A=LOG(2) :Q=M(X,Y)
520 IF Q AND 2 THEN D=1 ELSE IF Q AND 1 THEN D=0 ELSE D=5
530 FOR K=0 TO MAX
540   Z=-((Y AND 1)=1)
550   IF S1 THEN P=Y*64+X*6+Z*3+66 :PRINT@ P,A$;
      :FOR J=0 TO 30 :NEXT :PRINT@ P,B$;
      :FOR J=0 TO 30 :NEXT :PRINT@ P," ";     '*BLINKER
560   IF Y=N IF Y=0 THEN K=MAX :GOTO 750      '*GO IF FINISHED
570   ON D+1 GOTO 580,590,600,610,620,630
580   Y=Y-2 :GOTO 640                          '* N
590   X=X+Z-1 :Y=Y-1 :GOTO 640                 '* NW
600   X=Y+Z-1 :Y=Y+1 :GOTO 640                 '* SW
610   Y=Y+2 :GOTO 640                          '* S
620   X=Y+Z :Y=Y+1 :GOTO 640                   '* SE
630   X=X+Z :Y=Y-1                             '* NE
640   PTR=PTR+1 :B(PTR)=Y*20+X
650   D=D+3 :IF D>5 THEN D=D-6
660   IF M(X,Y)=2^D THEN NEXT K              '*IF CUL-DE-SAC
670   Q=M(X,Y)-2^D :AA=LOG(Q)/A
      :IF AA=INT(AA) THEN D=AA :NEXT K       '*IF ONLY 2 DOORS
```

```
680     FOR J=0 TO 5                            '*MULTIPLE DOORS
690       D=D-1 :IF D<0 THEN D=5                '*TURN CLOCKWISE
700       IF M(X,Y) AND (2↑D)  THEN J=6         '*  UNTIL EXIT
710     NEXT J                                  '*    IS FOUND.
720     FOR J=PTR-1 TO 0 STEP -1                '*ALSO UPDATE
730       IF B(J)=B(PTR) THEN PTR=J :J=-1       '*  BEST-ROUTE
740     NEXT J                                  '*    LIST.
750 NEXT K :RETURN
760 '
    *** ROUTINE MARKS BEST ROUTE FOR PRINTOUT ***
770 FOR J=PTR TO 0 STEP -1
780   Y=INT(B(J)/20) :X=B(J)-Y*20
790   M(X,Y)=M(X,Y)+64                          '* MARKER ON CELL
800   IF X=0 IF Y=V THEN J=-1                   '* EXIT LOOP
810 NEXT J :RETURN
820 '
    *** SEND COMPLETED MAZE TO THE PRINTER ***
830 A$="      " :FS$="/" :BS$=CHR$(92)      '** BS$ IS BACKSLASH
840 P$(0,1)= "  >--" :P$(2,3)=P$(0,1) :PE$(1)= "  >"
850 P$(1,1)= "  >  " :P$(3,3)=P$(1,1) :PE$(2)= " "+FS$
860 P$(2,1)= "<    " :P$(0,3)=P$(2,1) :PE$(3)= "<"
870 P$(3,1)= "<  **" :P$(1,3)=P$(3,1) :PE$(4)= " "+BS$
880 P$(0,2)= " "+FS$+"    " :P$(2,4)=P$(0,2)
890 P$(2,2)= " "+BS$+"    " :P$(0,4)=P$(2,2)
900 P$(1,2)=A$ :P$(3,2)=A$ :P$(1,4)=A$ :P$(3,4)=A$
910 RESTORE :FOR J=1 TO 8 :READ T(J) :NEXT
920 DATA  1,  2, 64,  4, 64, 32,  1, 16
930 '
940 FOR K=0 TO H-1 :LPRINT"  ****    "; :NEXT
950 LPRINT P$(2,2)+P$(0,2)
960 FOR J=0 TO V STEP 2
970   FOR L=1 TO 4
980     IF J=0 AND L=1 THEN L=2
990     IF L=1 THEN R=-1 ELSE IF L=3 THEN R=1 ELSE R=0
1000    FOR K=0 TO H :A=0 :B=2
1010      IF M(K,J) AND T(L) THEN A=1
1020      IF M(K,J+R) AND T(L+4) THEN B=3
1030      LPRINT P$(A,L); P$(B,L);
1040    NEXT K :IF J=0 AND L=2 LPRINT" " ELSE LPRINT PE$(L)
1050  NEXT L
1060 NEXT J
1070 FOR K=0 TO H
1080   IF M(K,V) AND 64 THEN B=3 ELSE B=2
1090   LPRINT P$(0,1)+ P$(B,1);
1100 NEXT K :LPRINT PE$(1) :LPRINT A$;
1110 FOR K=0 TO H-1 :LPRINT P$(2,2); P$(0,2); :NEXT
1120 LPRINT P$(2,2)+PE$(2) :LPRINT A$; P$(0,2); P$(2,2);
1130 FOR K=1 TO H :LPRINT"  ****    "; :NEXT :LPRINT" "
1140 RETURN
1150 '
      *** DRAW HONEYCOMB ON SCREEN ***
1160 CLS
1170 W$=CHR$(191) :DB$=CHR$(29)+CHR$(26) :A$=CHR$(143)
     :A1$=CHR$(140) :A2$=CHR$(188)
1180 FOR K=0 TO 3 :PRINT W$W$A$A$W$W$;:NEXT :PRINT W$W$W$DB$;
1190 FOR J=0 TO V :PRINT W$;
1200   IF J AND 1 THEN PRINT W$A1$A1$;
1210   FOR K=0 TO 9
1220     PRINTCHR$(145)"  "CHR$(162)A1$A1$; :NEXT
1230   IF J AND 1 THEN PRINT CHR$(8)CHR$(8)W$; ELSE PRINT W$W$;
1240   PRINT DB$;
1250 NEXT :PRINTW$W$W$;
1260 FOR K=0 TO 9 :PRINT W$W$A2$A2$W$W$; :NEXT :PRINTDB$;
1270 FOR K=69 TO 127 STEP 6
1280     PRINT@ K,A$A$; :PRINT@ K+64*V-3,A2$A2$;
1290 NEXT :RETURN
1300 '
   *** GAME ROUTINE.  PLAYER RACES BEE THROUGH MAZE ***
1310 X=3 :Y=V*3+6 :C=PTR+1 :W=0 :AR=14400 :J=0 :PX=H :PY=0
1320 B$(0)="  " :B$(1)=A$ :B$(2)=CHR$(175)+CHR$(172)
     :B$(3)=CHR$(142)+CHR$(190) :B$(4)=B$
1330 K=PEEK(AR) :X1=0 :Y1=0
     :RESET(X,Y) :RESET(X+1,Y+1) :RESET(X,Y+1) :RESET(X+1,Y)
1340   IF K AND 8 THEN Y1=-1              '* UP ARROW
1350   IF K AND 16 THEN Y1=1              '* DN ARROW
1360   IF K AND 32 THEN X1=-2             '* LEFT ARROW
1370   IF K AND 64 THEN X1=2              '* RIGHT ARROW
1380   IF K=2 THEN W=-1 :RETURN           '*<CLEAR> TO RESIGN
1390   X2=X+X1 :Y2=Y+Y1 :IF POINT(X2,Y2) OR POINT (X2+1,Y2)
       OR POINT(X2+1,Y2+1) OR POINT(X2,Y2+1) THEN 1410
1400   X=X2 :Y=Y2
1410   SET(X,Y) :SET(X+1,Y+1) :SET(X,Y+1) :SET(X+1,Y)
1420   J=J-1 :IF J<0 THEN J=4 :C=C-1
       :IF PX=0 AND PY=V THEN W=1 :RETURN
        ELSE PY=INT(B(C)/20) :PX=B(C)-PY*20
       :P=PY*64+PX*6+66 -((PY AND 1)=1)*3     '* MOVE THE BEE
1430   PRINT@ P,B$(J);
1440   IF X>120 IF Y<4 THEN RETURN     '*PLAYER WINS
1450 GOTO 1330
1990 '
     ** PRINT INSTRUCTIONS AND SET UP MAZE DIMENSIONS **
2000 CLS
2010 PRINT@ 72,"BEE AMAZED  --  PROGRAMMED BY DAN ROLLINS"
2020 PRINT :PRINT"THIS PROGRAM CREATES A MAZE WHICH IS"
2030 PRINT"COMPRISED OF HEXAGONAL CELLS, SIMILAR TO A HONEYCOMB."
2040 PRINT"YOU MAY SEND THE MAZE DIRECTLY TO YOUR PRINTER"
2050 PRINT"(SOLUTION OPTIONAL)" :PRINT                    OR"
2060 PRINT"WATCH AS THE MAZE IS CREATED AND SOLVED ON YOUR"
2070 PRINT"SCREEN, THEN RACE THE BEE THROUGH THE MAZE."
2080 PRINT : INPUT"PRINT THE MAZE OR RACE THE BEE (P/R)";Q1$
2090 IF Q1$="R" THEN S1=1 ELSE IF Q1$<>"P" THEN 2080
2100 IF S1 THEN V=13 :H=9  :GOTO 2280
2110 '** GET INPUT FOR PRINTER OPTION **
2120 CLS :PRINT :INPUT"HEIGHT OF MAZE ";V :V=INT(V/2)*2-1
2130 PRINT :INPUT"WIDTH OF THE MAZE ";H
2140 PRINT"CREATING THE MAZE.....,PLEASE BE PATIENT."
2150 GOSUB 100        '* CREATE MAZE
2160 PRINT"SOLVING THE MAZE" :GOSUB 500
2170 PRINT"YOU MAY PRINT THE MAZE AS MANY TIMES AS DESIRED"
2180 PRINT"BUT AFTER ONE PRINTOUT WITH THE SOLUTION, ALL "
2190 PRINT"THE FOLLOWING PRINTOUTS WILL BE SOLVED." :PRINT
2200 INPUT"READY PRINTER..... HIT <ENTER>";Q2$ :PRINT
2210 INPUT"DO YOU WANT THE SOLUTION ON THIS PRINTOUT (Y/N)";Q2$
2220 IF Q2$="Y" THEN GOSUB 770 ELSE IF Q2$<>"N" THEN 2210
2230 GOSUB 830              '* LINEPRINT ROUTINE
2240 INPUT 'SAME OR DIFFERENT MAZE (S/D)  ";Q3$
2250 IF Q3$="D" THEN RUN ELSE IF Q3$<>"S" THEN 2240
2260 GOTO 2210
2270 '** INSTRUCTIONS FOR THE RACE **
2280 CLS :PRINT :PRINT'   RELAX AND WATCH AS A UNIQUE MAZE"
2290 PRINT"DEVELOPES ON YOUR SCREEN.  WHEN THE FLASHING 'BEE'"
2300 PRINT"APPEARS, THE MAZE IS COMPLETE AND THE SHORTEST"
2310 PRINT'ROUTE IS BEING DETERMINED. USE THIS TIME TO PLAN"
2320 PRINT"YOUR OWN PATH." :PRINT
2330 PRINT"YOU WILL BEGIN IN THE LOWER-LEFT CORNER.  WORK YOUR"
2340 PRINT"WAY UP AND RIGHT TO THE EXIT BY PRESSING ONE OR "
2350 PRINT"MORE OF THE ARROW KEYS."
2360 PRINT"USE <CLEAR> TO RESIGN A HOPELESS GAME."
2370 PRINT :PRINT
2380 PRINT :INPUT"HIT <ENTER> TO BEGIN";Q1$
2390 GOSUB 1160 :GOSUB 100
2400 PRINT@ 56,CHR$(135)+CHR$(131)+STRING$(4,131);
     :PRINT@ V*64+129,STRING$(5,176)CHR$(184);
2410 '** FOLLOWING LINE SAVES THE SCREEN IN S$() ARRAY **
2420 S$="":K=VARPTR(S$) :POKE K,255 :POKE K+1,0
     :FOR J=0 TO 3 :POKE K+2,60+J :S$(J)=S$ :NEXT
2430 GOSUB 500
2440 CLS :PRINT CHR$(23) :PRINT :PRINT
2450 FOR J=1 TO 3 :PRINT :PRINT J :FOR K=1 TO 1500 :NEXT :NEXT
2470 '** THIS LINE RESTORES SAVED SCREEN **
2480 CLS :PRINT S$(0)' 'S$(1)" "S$(2)" "S$(3);
2490 GOSUB 1310           '* RACE THROUGH THE MAZE
2500 CLS :PRINT CHR$(23) :PRINT@ 600,"BZZZZZZZZ"
2510 IF W=1 THEN 2540
2520 IF W=-1 THEN PRINT"YOU RESIGNED!" :GOTO 2550
2530 PRINT:PRINT"YOU WON!!!!!!!"
    :PRINT"YOU ARE NOW AN HONORARY"
    :PRINT"BUMBLE BEE!" :GOTO 2550
2540 PRINT :PRINT'LOOKS LIKE I STUNG YOU GOOD !!"
2550 PRINT :INPUT "PLAY AGAIN (Y/N)";Q4$
2560 IF Q4$="N" THEN CLS :END ELSE IF Q4$<>"Y" THEN 2550
2570 INPUT'SAME OR DIFFERENT MAZE (S/D)";Q5$
2580 IF Q5$="D" THEN RUN ELSE IF Q5$<>"S" THEN 2570
2590 GOTO 2440
```

Apple to TRS-80 to Pet and Back

Graphics Conversion

Richard Kaplan

He sat in the basement hunched over his computer for hours, ambitiously entering a listing of his favorite game. "Converting this program to my machine should be a snap," he thought. "After all, I'm already an old pro at Apple programming. How much different can a TRS-80 be?" Months of fruitless programming later, he surrendered to his computer. He had discovered the hard way just how bewildering program conversion can be to someone with a knowledge of only his own computer.

Graphics conversion is perhaps one of the most frustrating problems with which a microcomputer owner must deal. To an Apple owner, the command "PRINT @ 1000, A + B" might seem like a way of instructing his computer to wait until ten o'clock before printing A + B. By the same token, a TRS-80 owner is just as likely to be able to decipher the meaning of the command "HGR" as he is likely to know offhand the Hungarian word for "disk drive."

But nowhere in the Apple owner's manual can he discover the meaning of PRINT @. Thus, very often, extremely competent programmers find themselves totally out of luck when translating programs for their machines.

This article will deal with the Apple II, the TRS-80 Models I and III, and the PET. In many cases it is possible to translate graphics for one machine directly to another, just as one translates foreign languages. However, there are many situations in which it is quite unrealistic to attempt direct conversion.

At those times, the best approach is to begin by finding out exactly what each graphics function in the program from which you are translating actually does. If you can plot each point on paper, then often it will be possible to modify the screen or devise an algorithm better suited to your computer. So, let's first take a look at the graphics capabilities of each computer.

Richard Kaplan, 53 Capral Lane, New City, NY 10956.

APPLE

The Apple produces graphics in three ways: standard PRINT statements and two special graphics modes.

Any computer can produce graphics by printing characters on the screen. A simple bar graph, for instance, can easily be generated by drawing asterisks in the appropriate positions on the screen. The Apple provides two commands which greatly aid in developing programs of this nature and which will be very helpful in translating TRS-80 programs to the Apple.

The first step with "printed" graphics is to clear the screen. Typing HOME (or executing this statement from within a program in Applesoft) will accomplish this. If you have Integer Basic, the correct statement is CALL -936.

VTAB and HTAB

The VTAB statement controls the location of the cursor along the Y axis. There are 24 lines on which the Apple can print in Text mode. Typing VTAB XX, where XX is any number from 1 to 24, will move the cursor to that location without erasing any previous characters. As an example, suppose we had executed the following: FOR I=1 to 12:PRINT"HELLO":NEXT. Executing the statement VTAB5:PRINT "GOODBYE" would cause the "HELLO" on the fifth line to be replaced with "GOODBYE."

This same principle can be used with horizontal tabbing. Typing HTAB XX, where XX is any number from 1 to 40, will move the cursor to the appropriate horizontal position.

HTAB and VTAB can be very useful when converting other programs to the Apple, especially when used in conjunction with the other special functions.

PEEK (37) contains a number, which can range from 0 to 23, holding the value of the vertical position of the cursor. This number is one *less* than the value for the same line if used in a VTAB statement. If the cursor is on line 10 and you wish to move the cursor up one position, the command VTAB PEEK(37) will do just that. HTAB PEEK(36) or HTAB POS(0) will do the same horizontally, i.e. move the cursor back one position. Caution should be exercised, however, not to HTAB to a position less than 1 or greater than 40, or to VTAB to a position less than 1 or greater than 24.

Although using ordinary PRINT statements is a very primitive means of programming graphics, in some cases it may be the best and most direct method to use in converting a program. In situations involving more intricate graphics, however, you may wish to use one of the Apple graphics modes.

The Apple has two graphics modes. These modes allow the use of as many as sixteen colors, as well as some very powerful plotting statements. The only disadvantage to using the Apple graphics modes is that text and graphics cannot be mixed on the same area of the screen without tremendous programming effort. For most purposes the programmer is restricted to four lines of text at the bottom of the screen.

Lo-Res Graphics

Apple low-resolution graphics are very convenient for simple graphics programs. An array of graphics blocks 40 x 40 may be used, with four lines of text at the bottom. A 40 x 48 array is possible without text. Sixteen colors are available with lo-res graphics.

Typing GR (or using this from within a program) enters the lo-res mode (mixed text-graphics.) The screen is cleared to black and PRINT statements produce output only on the bottom four lines of text.

If you want the larger (40 x 48) graphics area, simply type POKE -16302,0. The four lines of text at the bottom disappear and you have an additional eight lines of graphics to work with on the bottom of the screen.

Before plotting any points, the Apple must be assigned a specific color. Sixteen colors are available. To assign a color to graphics, type COLOR=X, where X is any of the following: 0 black, 1 magenta, 2 dark blue, 3 purple, 4 dark green, 5 grey, 6 medium blue, 7 light blue, 8 brown, 9 orange, 10 grey, 11 pink, 12 green, 13 yellow, 14 aqua, 15 white.

Assigning a color has no effect on graphics already on the screen. Only graphics statements executed after this will be of that color. Therefore, executing several color statements allows multiple colors to be used on the same screen.

Now for a very basic question: How do you plot a point? Basically, the Apple screen operates similarly to a mathematical coordinate system. The X axis can be pictured as running along the top of the screen, numbered with coordinates from 0 to 39. The Y axis runs parallel to the left side of the screen, with 0 at the top and 39 or 47 at the bottom, depending on whether you have chosen to use the extra eight lines or not. Thus the point 0,0 is at the top left of the screen and the point 39,39 is at the bottom right of the screen (in mixed text-graphics mode).

The PLOT statement actually plots a specific point. Its format is PLOT X,Y. Thus PLOT 20,20 would place a graphics square at a location 20 points away from the left of the screen and twenty points down from the top. To erase this point, set the color to 0 (black) or whatever the background color is and re-plot the point.

It is also possible with the Apple to draw a line between two points on the screen. The command HLIN X,Y AT Z would plot a horizontal line between horizontal coordinates X and Y at vertical location Z. Thus the statement HLIN 1,20 AT 10 would connect the points 1,10 and 20,10. VLIN X,Y at Z does the same thing for a vertical line. Thus, the command VLIN 1,20 AT 10 would connect the points 10,1 and 10,20.

Figure 1.

```
10 GR                   Enter lo-res graphics mode
20 COLOR = 3            Set color to be purple
30 HLIN 0,39 AT 0       Draw line at top of screen
40 VLIN 0,39 AT 39      Draw line at right of screen
50 HLIN 0,39 at 39      Draw line at bottom of screen
60 VLIN 0,39 AT 0       Draw line at left of screen
```

For an example of lo-res graphics see the program in Figure 1 which shows a border around the lo-res screen.

As a last note to using lo-res graphics, the user should know how to exit this mode. Simply type TEXT and the screen will revert to its usual 24 lines of text and 40 characters per line.

Hi-Res Graphics

The Apple high-resolution graphics mode offers some of the best graphics capabilities available on any microcomputer. Although only eight colors can be used, resolution of 280 x 192 pixels may be obtained, allowing highly detailed objects and extremely impressive graphics to be programmed.

To enter hi-res graphics, simply type HGR. This gives you a 280 x 160 grid with four lines of text at the bottom. Typing HGR2 instead of HGR, or typing POKE -16302,0 after entering HGR, will place you in the full-screen graphics mode, with a resolution of 280 x 192.

The high-resolution colors are set very similarly to low-resolution colors. HCOLOR= is the equivalent of the lo-res statement COLOR=. The eight colors available in high resolution graphics are: 0 black, 1 green, 2 blue, 3 white 1, 4 black, 5 depends on TV, 6 depends on TV, 7 white 2.

The hi-res coordinate system is numbered from 0 to 279 along the X axis (top of the screen) and from 0 to 159 (HGR) or 0 to 191 (HGR2) along the Y axis.

The hi-res equivalent of PLOT is HPLOT. HPLOT X,Y plots a point at location X on the X axis and location Y on the Y axis.

In high-resolution graphics, it is possible to plot from any location to any other location, even if it necessitates the drawing of a diagonal line. The statement HPLOT X,Y TO X,Y or HPLOT X,Y TO X,Y TO X,Y TO X,Y connects the points between the "TO." This is a very powerful statement, and it is not available in lo-res mode.

For an example of Apple hi-res graphics, see the program in Figure 2 which draws a border around the screen, as in the last example.

As with lo-res graphics, TEXT will cause the computer to revert to normal text mode.

TRS-80

TRS-80 graphics are much simpler than Apple graphics, although not quite as versatile. No special graphics modes are required. Text may be printed at a specific location on the screen (as with the Apple VTAB and HTAB statements), and graphics may be used on the same screen. The resolution of the TRS-80 may be compared to the Apple lo-res mode.

The Model I and the Model III are almost identical machines; 95% of TRS-80 statements can be used on both machines. For this reason, I will use "TRS-80" to refer to both models. When a specific feature is available on only one model, I will specify that model.

PRINT Statements

The TRS-80, like the Apple, can produce graphics through PRINT statements. How-

Table 1.

CHR$(X) X-Value	Action
24	Move cursor one space left
25	Move cursor one space right
26	Move cursor one line down
27	Move cursor one line up
28	Move cursor to upper-left corner

Figure 2.

```
10 HGR                                          Enter hi-res graphics mode
20 COLOR = 1                                    Set color to green
30 HPLOT 0,0 TO 0,159 TO 279,159 TO 279,0 TO 0,0   Connect the four corners of the screen
```

ever, the TRS-80 has a special statement, PRINT @, which makes it possible to refer to any screen location specifically by number. This can be a very powerful statement if used efficiently.

The TRS-80 screen is composed of 16 lines of 64 characters each, for a total of 1024 possible character positions. These positions are numbered from 0 to 1023, with 0 in the upper left of the screen, 63 at the end of the first line, 64 at the beginning of the second line, etc., and 1023 as the last position in the last line of the screen. The correct syntax for this statement is PRINT @ XXXX,... where XXXX is a number from 0 to 1023 and ... is any expression valid in a standard PRINT statement.

Let's go back to the example we used with the Apple. First type FOR I = 1 to 12:PRINT"HELLO":NEXT. In order to replace the fifth "HELLO" on the Apple with "GOODBYE" we typed VTAB 5: PRINT"GOODBYE". On the TRS-80, however, the best way to accomplish the same thing is to identify the numerical value of the first location on the fifth line of the screen.

The formula (X-1)*64 is used to locate the point at which to print if you wish to begin printing at the first position on the Xth line of the screen. Therefore, in order to print "HELLO" twelve times and replace the fifth with "GOODBYE," we would type: FORI=1to12:PRINT"HELLO": NEXT:PRINT@(5-1)*64,"GOODBYE". Note, though, that the cursor location has been moved to the fifth line of the screen, so that the word "READY" will now print where a "HELLO" formerly was. If you did not wish this to happen, you could add PRINT@(13-1)*64,"; which would move the cursor location back down to the thirteenth line.

Very often in converting graphics, you will want to move the cursor up or down one column without using a PRINT @ statement. Maybe you do not know the current cursor position, or perhaps you are converting a PET program which uses a special PET feature to relocate the cursor. To do this on the TRS-80 you would use the CHR$ function. Typing PRINT CHR$(X); where X is one of several special cursor movement codes, will perform the desired action. The codes are listed in Table 1.

Graphics Characters

The TRS-80 can also create graphics by printing special graphics characters. These characters (see Figure 3) consist of all 64 possible on/off permutations of a 2 x 3 matrix ($2^2 = 2^6 = 64$). These graphics characters may be printed by using the CHR$ function. Typing PRINT CHR$(X), where X is the numerical code for the special graphics character desired, prints that character. This function can also be used in conjunction with the PRINT @ statement. In addition, the statement PRINT STRING$ (X,Y) will print a string composed of graphics character Y concatenated with itself X times. Thus, the statement PRINT @0,STRING$(64,191) will print a horizontal line across the top of the screen.

The Model III TRS-80 has a set of 96 additional special characters. Sixty-four of these can be printed exactly as the 64 described above.They are codes 192-255 (see Figure 4). However, there is one short statement which must be executed prior to printing these characters.

When the Model III is powered up, these 64 codes represent "space compression" characters. PRINT CHR$(192) prints no spaces, PRINT CHR$(193) prints one space, etc., until PRINT CHR$(255), which would print 63 spaces. In order to replace these space compression characters with the special graphics characters, type PRINT CHR$(21). This statement functions as a toggle switch between space compression characters and special graphics characters.

In addition to the 64 special graphics characters available to the Model III owner, there exists a special set of Japanese characters. These characters are CHR$ numbers 192-255, as are the special graphics characters. They are selected by executing the statement PRINT CHR$(22) after selecting the special character set (PRINT CHR$(21)).

If you are amazed at the number of characters available on the Model III, you are in for still another surprise. There is yet another set of special graphics characters available to the Model III user. These characters are codes 0-31 (see Figure 4). However, they are only accessible by means of a POKE statement.

In order to print a graphics character from 0 to 31, the value of that character must be poked into the appropriate memory location, or what Radio Shack refers to as VIDRAM. These video addresses start at 15360 and end at 16383, and are equivalent to a PRINT @ address plus 15360. Thus, in order to print special character 10 at the beginning of the screen, you would type "POKE 15360,10."

We are not done yet with the TRS-80 graphics capabilities. Both models can also plot specific points on the screen. These

Figure 4. TRS-80 Model III special characters (codes 0-31, 192-255).

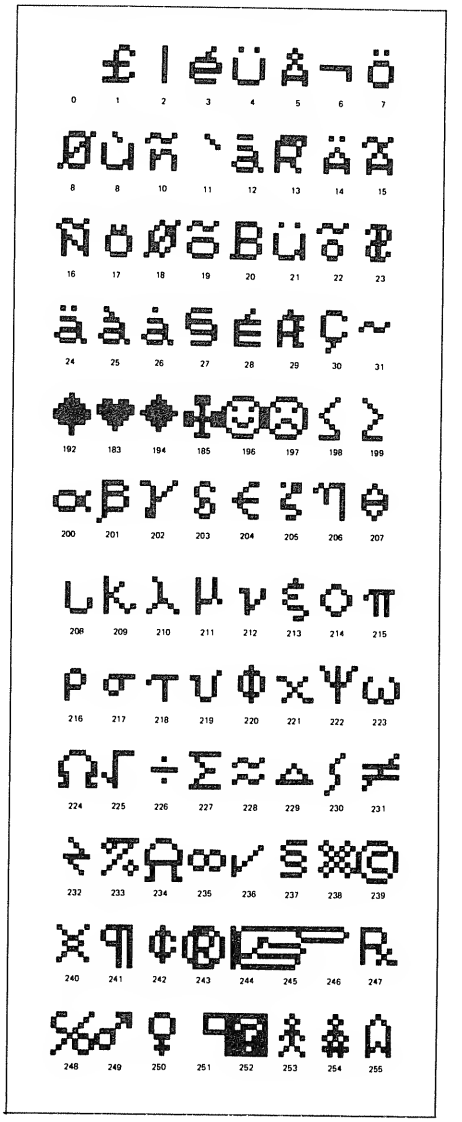


Figure 3. TRS-80 graphics characters (codes 128-191).

DEC	HEX	Z-80 OP CODE	GRAPHIC	TRS-80 BASIC
128	80	ADD A,B		END
129	81	ADD A,C		FOR
130	82	ADD A,D	80 81 82 83	RESET
131	83	ADD A,E		SET
132	84	ADD A,H		CLS
133	85	ADD A,L		CMD
134	86	ADD A,(HL)	84 85 86 87	RANDOM
135	87	ADD A,A		NEXT
136	88	ADC A,B		DATA
137	89	ADC A,C		INPUT
138	8A	ADC A,D	88 89 8A 8B	DIM
139	8B	ADC A,E		READ
140	8C	ADC A,H		LET
141	8D	ADC A,L	8C 8D 8E 8F	GOTO
142	8E	ADC A,(HL)		RUN
143	8F	ADC A,A		IF
144	90	SUB B		RESTORE
145	91	SUB C	90 91 92 93	GOSUB
146	92	SUB D		RETURN
147	93	SUB E		REM
148	94	SUB H		STOP
149	95	SUB L	94 95 96 97	ELSE
150	96	SUB (HL)		TRON
151	97	SUB A		TROFF
152	98	SBC A,B		DEFSTR
153	99	SBC A,C	98 99 9A 9B	DEFINT
154	9A	SBC A,D		DEFSNG
155	9B	SBC A,E		DEFDBL
156	9C	SBC A,H		LINE
157	9D	SBC A,L	9C 9D 9E 9F	EDIT
158	9E	SBC A,(HL)		ERROR
159	9F	SBC A,A		RESUME
160	A0	AND B		OUT
161	A1	AND C	A0 A1 A2 A3	ON
162	A2	AND D		OPEN
163	A3	AND E		FIELD
164	A4	AND H		GET
165	A5	AND L	A4 A5 A6 A7	PUT
166	A6	AND (HL)		CLOSE
167	A7	AND A		LOAD
168	A8	XOR B		MERGE
169	A9	XOR C	A8 A9 AA AB	NAME
170	AA	XOR D		KILL
171	AB	XOR E		LSET
172	AC	XOR H		RSET
173	AD	XOR L	AC AD AE AF	SAVE
174	AE	XOR (HL)		SYSTEM
175	AF	XOR A		LPRINT
176	B0	OR B		DEF
177	B1	OR C		POKE
178	B2	OR D	B0 B1 B2 B3	PRINT
179	B3	OR E		CONT
180	B4	OR H		LIST
181	B5	OR L		LLIST
182	B6	OR (HL)	B4 B5 B6 B7	DELETE
183	B7	OR A		AUTO
184	B8	CP B		CLEAR
185	B9	CP C		CLOAD
186	BA	CP D		CSAVE
187	BB	CP E	B8 B9 BA BB	NEW
188	BC	CP H		TAB(
189	BD	CP L		TO
190	BE	CP (HL)		FN
191	BF	CP A	BC BD BE BF	USING

plotted points can appear on the screen in conjunction with any other graphics features on the TRS-80, as well as text.

The TRS-80 screen is divided into a 128 x 48 array, any block of which may simply be turned on or off. Color is not supported by either TRS-80.

The statement SET (X,Y) turns on the graphics block at horizontal location X (X axis) and vertical location Y (Y axis). The X value can be between 0 and 127, while the Y value can range from 0 to 47. An important difference between turning on a graphics block and printing a graphics character is that a graphics block will not scroll off the screen. The only way to eliminate it is through the RESET statement or clearing the screen, which is done with CLS.

The RESET statement, as previously stated, turns off the specified graphics block. The syntax of the statement is RESET (X,Y), and it has exactly the same parameters as does the SET statement.

See Figure 5 for an example demonstrating some basic characteristics of TRS-80 graphics.

PET

PET graphics are very different from TRS-80 graphics. There are no special graphics modes on the PET, nor can a specific point on the screen be referred to by means of a coordinate system. Essentially PET graphics consist of standard PRINT statements combined with special cursor movement characters. (The graphics characters which may be printed are accessed by pressing the Shift key and the appropriate keyboard key. The cursor movement keys are specifically marked, and sometimes must be pressed in conjunction with the Shift key.)

The PET has six cursor movement characters. These characters are treated just like any other character on the keyboard, as they may be assigned to a string variable and printed. When they are printed, they appear as special symbols, quite unique from any character on any other computer.

The Home Cursor key returns the cursor to the upper lefthand corner of the screen. It is printed as an "S" in reverse video.

The shifted Home Cursor key returns the cursor to the upper lefthand corner of the screen and also clears the screen. It appears on the screen as a heart in reverse video.

The Cursor Down/Up key moves the cursor down one line. It appears as a "Q" in reverse video.

The shifted Cursor Down/Up key moves the cursor up one line. It appears as an empty circle with a black border.

The Cursor Right/Left key moves the

cursor one position to the right. It appears as a right bracket in reverse video.

The shifted Cursor Right/Left key moves the cursor one position to the left. It appears as a black rectangle with a vertical white line through it.

These six cursor control characters can be treated just like any other character in the PET character set. For example, the sequence PRINT "(Shifted Home Cursor) (Cursor Down) (Cursor Down) (Cursor Down) HELLO" would clear the screen and place the word "HELLO" on the fourth line.

The PET also has an alternate set of characters, which can be selected by typing POKE 59468,14. The keyboard will then function with the alternate character set (see Figure 6). To return to the standard character set, execute the statement POKE 59468,12.

The program in Figure 7, though quite simple, illustrates the basic method of incorporating graphics into a PET program. The program clears the screen, moves the cursor to the fourth line, and draws a square.

Figure 5.

```
10  CLS                                  Clear screen
20  FOR X = 0 to 127:SET (X,47):         Draw a border around the screen
    SET (X,0):NEXT
    FOR X=0 to 47:SET (0,X):SET(127,
    X):NEXT:
30  PRINT @512,"Press ENTER to see       Print message at center of screen
    special characters";
40  INPUT"";X$                           Wait for Enter key
50  CLS                                  Clear screen
60  PRINT CHR$(21)                       Select graphics characters
70  FOR I=192 to 255:                    Print characters
    PRINT CHR$(I):" ";:NEXT
80  INPUT"PRESS ENTER TO SEE             Wait for Enter
    Japanese characters";X$
90  PRINT CHR$(22)                       Select Japanese characters
100 INPUT "PRESS ENTER TO                Wait for Enter
    END@;X$5.B END";X$
110 PRINT CHR$(22);CHR(21);:             Select standard character sets
    CLS:END                              Clear screen
```

CONVERSION TO APPLE

From TRS-80

When converting a program from the TRS-80 to the Apple, you may use the text mode, low-resolution graphics, or high-resolution graphics.

TEXT mode should be used when the original program involves PRINT @ statements, or simply PRINT statements, and the text or graphics on the screen can be condensed to 40 columns wide. Aside from the smaller screen, the only disadvantage to using the Apple will be that graphics characters cannot be generated in text mode.

The statement causing the most confusion in conversion is probably PRINT @. However, this is really the easiest statement to convert. The TRS-80 statement PRINT @ X, "THIS IS A TEST" can be changed into three Apple statements:

```
VTAB INT(X/64) + 1
HTAB X + 1 - INT(X/64) * 64
PRINT "THIS IS A TEST"
```

When using this conversion procedure, however, the user must be very cautious not to HTAB past column 40. The TRS-80 screen is 64 columns wide, in contrast to the 40-column screen of the Apple. If only 40 columns are needed, then this procedure is probably the easiest to use. It might be advisable, however, to plot out on graph paper the results of the PRINT @ statements to obtain more aesthetically pleasing results.

The easiest conversion between TRS-80 and Apple (in TEXT mode) is clearing the screen. Essentially all that must be done is to replace every occurrence of CLS in a TRS-80 program with HOME.

The TRS-80 graphics should be simulated in either lo-res or hi-res graphics. These methods will provide the most graphically pleasing results. However, if text and graphics must be placed on the same screen, then TEXT mode must be used. In this case, you should follow the instructions under lo-res graphics, but substituting HOME for GR (in order to clear the screen but not enter the graphics mode).

PLOT statements, when used from TEXT mode, will not place graphics blocks at the appropriate coordinates, but will instead place standard text characters on the screen. The actual character which will be printed can be predetermined, but that is beyond the scope of this article.

The TRS-80 statement PRINT CHR$(31) will clear the screen from the current cursor position on. This can be emulated on the Apple by executing the statement CALL -958.

The TRS-80 special graphics characters (including the alternate character sets of the Model III) cannot be easily duplicated on the Apple. If you have a program which mixes text and special graphics characters, the only options available are to substitute characters from the Apple standard character set or use hi-res graphics and create a character generator, which is a most formidable task for an inexperienced programmer.

Apple lo-res graphics can be used when only graphics are used on the TRS-80 (SET statements), as opposed to text and graphics. But remember, lo-res offers at best a 40 x

Figure 7.

```
10 PRINT"(CLEAR SCREEN)                  Clear screen and move cursor
   (CURSOR DOWN) (CURSOR                 down to fourth line
   DOWN) (CURSOR DOWN)";
20 PRINT"------"                         Draw top of square
30 FOR I = 1 TO 7:                       Draw sides of square
   PRINT"     ":NEXT
40 PRINT"------"                         Draw bottom of square
```

48 array (with no text), while the TRS-80 has a 128 x 48 array of graphics. However, if it is possible to program a particular application within these constraints then lo-res graphics are preferable. Lo-res is easier to use than hi-res and provides twice as many colors from which to choose.

Using lo-res again requires condensing the TRS-80 screen. In this case, the graphics must be condensed to either 40 x 48 or 40 x 40. Once this has been done, the conversion procedure is quite simple.

In the TRS-80 program, look to see where the graphics portion begins. Usually a CLS statement will appear at this point. Replace the CLS with GR to clear the screen and enter lo-res mode.

Subsequent PRINT statements in the program will have to be restricted to four lines of text. These lines of text must be contiguous at the bottom of the screen. No special conversion of PRINT statements is required unless PRINT @ is used. In that case, keep in mind that only lines 21-24 may be used for text in lo-res graphics.

If you wish to replace the bottom four lines of text with an additional eight rows

Figure 6. PET standard and alternate character sets.

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
	0		16	SPACE	32	ø	48
	1	CRSR ↓	17	!	33	1	49
	2	RVS ON	18	"	34	2	50
	3	CLR HOME	19	#	35	3	51
	4	INST DEL	20	$	36	4	52
WHT	5		21	%	37	5	53
	6		22	&	38	6	54
	7		23	'	39	7	55
	8		24	(	40	8	56
	9		25	)	41	9	57
	10		26	*	42	:	58
	11		27	+	43	;	59
	12	RED	28	,	44	<	60
RETURN	13	CRSR →	29	—	45	=	61
SWITCH TO LOWER CASE	14	GRN	30	.	46	>	62
	15	BLU	31	/	47	?	63
@	64	U	85		106		127
A	65	V	86		107		128
B	66	W	87		108		129
C	67	X	88		109		130
D	68	Y	89		110		131
E	69	Z	90		111		132
F	70	[	91		112	f1	133
G	71	£	92		113	f3	134

PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$	PRINTS	CHR$
H	72	]	93		114	f5	135
I	73	↑	94		115	f7	136
J	74	←	95		111	f2	137
K	75		96		117	f4	138
L	76		97		118	f6	139
M	77		98		119	f8	140
N	78		99		120	SHIFT RETURN	141
O	79		100		121	SWITCH TO UPPER CASE	142
P	80		101		122		143
Q	81		102		123	BLK	144
R	82		103		124	CRSR ↑	145
S	83		104		125	RVS OFF	146
T	84		105		126	CLR HOME	147
INST DEL	148	CYN	159		170		181
	149	SPACE	160		171		182
	150		161		172		183
	151		162		173		184
	152		163		174		185
	153		164		175		186
	154		165		176		187
	155		166		177		188
PUR	156		167		178		189
CRSR ←	157		168		179		190
YEL	158		169		180		191

of graphics, execute the statement POKE -16302,0. You will then have a 40 x 48 array available.

A color should be selected before any points are plotted. (This color may be changed at any point in the program without changing previously plotted graphics.) This is done through the COLOR= statement (see above).

All SET statements should be replaced with PLOT statements. Essentially, SET (X,Y) becomes PLOT X,Y. Not all acceptable values for X and Y in a SET statement are valid values in a PLOT statement. X in a PLOT statement cannot exceed 39, and Y cannot exceed either 39 or 47, depending upon whether full screen graphics or mixed text-graphics is chosen.

The final step is to convert the TRS-80 RESET statements. This is done exactly as a SET statement is converted, with one exception. The color should be set to whatever the background is (usually black). Then executing a PLOT statement will transform that graphics block back into its original state (off).

High-Resolution Graphics

Converting TRS-80 graphics to hi-res graphics is much more involved than converting to TEXT mode or lo-res graphics, but the results are well worth the effort. The entire 128 x 48 grid can be incorporated into the Apple screen, along with all 64 graphics characters (ASCII codes 128-191). The alternate character sets of the Model III can also be simulated, though this requires substantial programming effort in some cases.

Before discussing the actual conversion process, let's take a closer look at the graphics capabilities of the TRS-80. We have said that the screen is a 128 x 48 array. But is this really so? In actuality, each graphics block is, itself, an array two blocks wide and three blocks high. This means that the TRS-80 graphics screen can be represented as a screen of (128 * 2) * (48 * 3), or 256 * 144, blocks. The Apple high resolution mixed text-graphics mode can accommodate 280 * 160 blocks, so the entire TRS-80 screen can in fact, be represented on the Apple.

If you have been following along, you will probably have noticed that there is one small problem with this conversion procedure. The TRS-80 screen, you will recall, is composed of 6144 blocks. The portion of the Apple screen we will use, however, contains 256 * 144, or 36,864, blocks. This means we will have to plot 36864/6144, or 6, points on the Apple for every point on the TRS-80. The way to do this follows.

First, select the hi-res graphics mode appropriate for your application (HGR or HGR2). Usually HGR will be sufficient, because even with the extra lines of text at the bottom of the screen there is enough room to accommodate the full TRS-80 screen.

The next step is to select a color with the HCOLOR= statement. This can be done by simply choosing a color from the chart in this article.

Whenever you encounter a SET (X,Y) statement, it must be converted into the equivalent HPLOT statements. The X and Y coordinates of the SET statements can be related to the Apple screen. The coordinates X * 2, Y * 3 correspond to the upper left point of the Apple 2 * 3 grid for that point. See Table 2 for a list of the six points on the Apple which compose that one point.

If the entire block is to be filled in, you should execute the statements HPLOT X * 2,Y * 3 TO X * 2,Y * 3 + 2:HPLOT X * 2 + 1,Y * 3 TO X * 2 + 1, Y * 3 + 2. It is a good idea to incorporate these statements into a subroutine. Then, any time a statement such as SET (2,3) appears in your source listing, you can simply substitute the statements X = 2:Y = 3:GOSUB 10000 (assuming you have used the above line of code as line 10000 and added a RETURN statement at the end).

Plotting one of the 64 TRS-80 graphics characters is very simple. First consult the chart to determine which of the six graphics blocks should be turned on or off. Then apply the formulas in the above chart and HPLOT the appropriate coordinates.

For example, let's say we wanted to print character 179. By examining the chart, we can see that this is composed of the top left, top right, bottom left, and bottom right portions of the 2 x 3 graphics grid. If we wanted to print this at TRS-80 coordinates (50,100), we would execute the following statements:

```
HPLOT 50 * 2+1,100 * 3 + 2
HPLOT 50 * 2,100 * 3 + 2
HPLOT 50 * 2+1,100 * 3
HPLOT 50 * 2,100 * 3
```

From PET

Converting PET graphics to the Apple can be exceedingly frustrating if the PET special graphics characters are used.

Table 2.

Apple X Value	Apple Y Value	Position
X * 2	Y * 3	Upper Left
X * 2 + 1	Y * 3	Upper Right
X * 2	Y * 3 + 1	Middle Left
X * 2 + 1	Y * 3 + 1	Middle Right
X * 2	Y * 3 + 2	Lower Left
X * 2 + 1	Y * 3 + 2	Lower Right

Table 3.

PET Cursor Control Character	Apple Cursor Location Statement
Home Cursor	VTAB 1:HTAB 1
Shifted Home Cursor (Clear Screen)	HOME
Cursor Down/Up	VTAB PEEK (37) + 2
Shifted Cursor Down/Up	VTAB PEEK (37)
Cursor Right/Left	HTAB PEEK (36) + 2
Shifted Cursor Right/Left	HTAB PEEK (36)

Table 4.

PET Character	Apple Statement	Function
Reverse On	Inverse	Print all subsequent characters in reverse video
Shifted Reverse On	Normal	Cancel all reverse video statements previously executed

Producing many of these on the Apple is comparable to producing the Model III special character sets. In many cases it is advisable to rewrite the entire programming algorithm so that it is more adaptable to use on the Apple.

If the graphics characters used on the PET are such that there is a comparable character in the Apple character set, then conversion is very easy. The PET screen is composed of 25 lines of 40 characters each, and the Apple screen contains 24 lines of 40 characters. The hardest part of the conversion is simply reducing the screen to 24 lines, which can usually be accomplished without much problem.

The main problem in converting between the PET and the Apple is substituting appropriate VTAB and HTAB statements for the cursor movement characters on the PET. This can usually be done directly using Table 3.

Now, let's say we have a PET program which clears the screen and draws a line on the fifth line of the screen. The program would have a statement which read PRINT" (Shifted Home Cursor) (Cursor Down/Up) (Cursor Down/Up) (Cursor Down/Up) (Cursor Down/Up) ..." The translated Apple program would read HOME:VTAB PEEK (37)+2:VTABPEEK(37)+2:VTAB PEEK (37)+2:VTAB PEEK(37)+2:PRINT"..." Note that the cursor location characters on the PET are actually part of the PET character set and thus are PRINTed as elements of a string literal (or even a string variable). The Apple, on the other hand, has cursor movement *statements* which *cannot* be used from within PRINT statements.

Both the PET and Apple support reverse video. On the PET, there are two special characters, which, again, can be used from within a string and PRINTed. On the Apple, there are separate statements to control this function. The appropriate commands are shown in Table 4.

The methods which the PET and the Apple incorporate to access reverse video are quite similar. Executing the appropriate statement causes all subsequent output to be printed in reverse video. There is one small difference, however. The Apple INVERSE statement can *only* be cancelled by a NORMAL statement. On the PET, either a carriage return *or* a Shifted Reverse on will do it.

Let's say the PET program you are translating has the statement PRINT "(Reverse On) THIS IS IN REVERSE FIELD (Shifted Reverse On) AND THIS IS NOT". The equivalent Apple statements would be INVERSE:PRINT"THIS IS IN REVERSE FIELD ";:NORMAL:PRINT "AND THIS IS NOT". Note the use of a semicolon after the first PRINT statement to cancel the carriage return.

CONVERSION TO TRS-80

From Apple

Conversion to TRS-80 from the Apple, when possible, is very easy. Most printed output from the Apple can be duplicated on the TRS-80, but since the TRS-80 screen only has 16 lines, whereas the Apple screen has 24 lines, in some cases the screen must be compressed or modified in some other way.

Only two of the three Apple "modes" can be emulated on the TRS-80. PRINTed output (VTAB, HTAB, etc.) can be easily converted, as can Apple lo-res graphics. The TRS-80 does not, however, have the ability to reproduce Apple high resolution graphics. If a hi-res graphics program must be converted to the TRS-80, the standard TRS-80 graphics must be used, and a substantial amount of resolution will be lost.

The only potential problem in converting an Apple program to the TRS-80 is in locating the equivalent PRINT @ location for a given set of VTAB and HTAB locations. For any given values X and Y, the Apple statement VTAB X:HTAB Y is equivalent to the TRS-80 expression PRINT @ (X * 64) + Y -65. One problem with using this method, however, is that since the TRS-80 screen only has 16 lines of text, this formula cannot be used in a situation where the X value is greater than 16. The only solution to this problem is to redesign the screen so that only 16 lines of text are used.

Another element in converting an Apple program to the TRS-80 is clearing the screen. Quite simply, replace each occurrence of HOME in the Apple program with a CLS in the TRS-80 version.

Apple lo-res graphics can be duplicated very easily on the TRS-80. Since the Apple lo-res screen contains at most a 40 x 48 matrix and the TRS-80 screen has a 128 x 48 matrix, this particular conversion is ideal.

When you encounter a GR statement, replace it with CLS. Then, simply replace each occurrence of PLOT X,Y with SET (X,Y). The actual values of X and Y will not change in this instance.

Color cannot be reproduced on the TRS-80, so COLOR= statements should be ignored, except where the COLOR is set to 0 or whatever the background color is at that moment. In that case, subsequent PLOT statements should be replaced with RESET statements in order to erase the graphics blocks at the appropriate coordinates.

From PET

Converting a PET program to run on the TRS-80 is similar to converting it to run on the Apple. However, the screen will have to be reduced not to 24 lines from the PET's 25, but to 16 lines. In addition, many of the PET special characters have no parallels on the TRS-80. If no acceptable character can be found on the TRS-80, the only solution is to plot out the individual points from the PET program and devise an algorithm to access TRS-80 SET statements or to PRINT TRS-80 special graphics characters.

When converting from the PET to the TRS-80, cursor control characters on the PET will probably cause the most confusion. These cursor control characters, however, have *direct* equivalents on the TRS-80, as shown in Table 5.

In order to access the TRS-80 codes, you should use the CHR$ function and PRINT the appropriate character. (Be sure to place a semicolon after the PRINT statement.) Thus, the equivalent of the PET statement PRINT"(Home Cursor) (Cursor Down/Up) (Cursor Right/Left) TEST" would be PRINT CHR$(28);CHR$ (26);CHR$(25);"TEST".

One PET character does not have an ASCII counterpart on the TRS-80. The Shifted Home Cursor on the PET (which clears the screen) should be replaced with the CLS on the TRS-80.

CONVERSION TO PET

From Apple

Converting a program from Apple to PET is, in many cases, almost impossible. The graphics capabilities of the PET simply operate very differently from those of most other computers.

The main problem in converting graphics from the Apple to the PET is that with the

Table 5.

PET Cursor Control Character	TRS-80 ASCII Code
Home Cursor	28
Cursor Down/Up	26
Shifted Cursor Down/Up	27
Cursor Right/Left	25
Shifted Cursor Right/Left	24

PET there is no way to access a particular screen location directly with a set of coordinates, such as Apple VTAB and HTAB statements or the TRS-80 PRINT @ statement. The best advice to PET owners is to rewrite the graphics routines of their programs in order to reproduce graphics efficiently.

There is a way to create a subroutine to simulate VTAB and HTAB statements, but it is generally not advisable unless relatively simple graphics are being used. It can be used with PRINTed graphics only, lo-res and hi-res graphics cannot be directly translated.

Essentially, we must first home the cursor then print as many Cursor Downs as the value of the argument of the VTAB minus 1 and as many Cursor Rights as the value of the argument of the HTAB statement minus 1. Thus, the statement VTAB 4: HTAB4;PRINT"THIS IS A TEST" can be translated as PRINT "(Home Cursor) (Cursor Down/Up) (Cursor Down/Up) (Cursor Down/Up) (Cursor Right/Left) (Cursor Right/Left) (Cursor Right/Left) THIS IS A TEST". By incorporating a FOR-NEXT loop, this can be made into a subroutine. While it is a very primitive means of duplicating a VTAB statement, it is a possibility for relatively simple PET graphics programs.

The one other necessary conversion between the Apple and the PET is the clear screen code. On the PET, the equivalent statement would be PRINT"(Shifted Home Cursor)".

From TRS-80

Converting a program from the TRS-80 to the PET is basically the same as converting an Apple program to the PET. It is exceedingly difficult to do, and the resulting program will not usually operate very efficiently if the same algorithm is used with both programs.

The clear screen code on the TRS-80 is CLS. This means that every occurrence of CLS in the TRS-80 program should be replaced with PRINT"(Shifted Home Cursor)".

If it is absolutely necessary to convert a TRS-80 program to the PET directly, it can be done. However, as with the Apple, only TEXT can be directly translated. This means that a graphics program which uses SET statements can usually *not* be translated.

PRINT @ statements are the primary means of producing graphics on the TRS-80 without SET statements. The PRINT @ address must first be converted into an equivalent set of horizontal and vertical coordinates. For any given PRINT @ coordinate X, the corresponding vertical (Y axis) coordinate is INT (X/64) + 1 and the horizontal coordinate is X + 1 - INT (X/64) * 64. Once these coordinates have been computed, the procedure above for Apple to PET conversion can be followed to move the cursor.

The graphics conversion techniques described here do not exhaust all possible conversion methods, nor have I covered every graphics statement on every computer mentioned. What I have attempted to do is familiarize the reader with the basic graphics principles of each computer and provide some insight as to how to approach the conversion process.

Graphics conversion is by no means an objective endeavor; once he has an understanding of how each computer operates the programmer's own creativity will more than likely influence his conversion technique more than anything else. □

Profile of a Legend

Leo Christopherson

Bob Liddil

The bee flits precariously above the tarantula, his little wings buzzing, waiting for a chance to deliver a points-winning sting. The tarantula sees this flying yummy and leaps into the air, trying to make him dinner. Who will win, the spider or the bee?

The number one android looks at the number to his right, then at the droids to his left. He approves the number, nodding, then points his laser pistol and dispatches the right number of droids to continue the game.

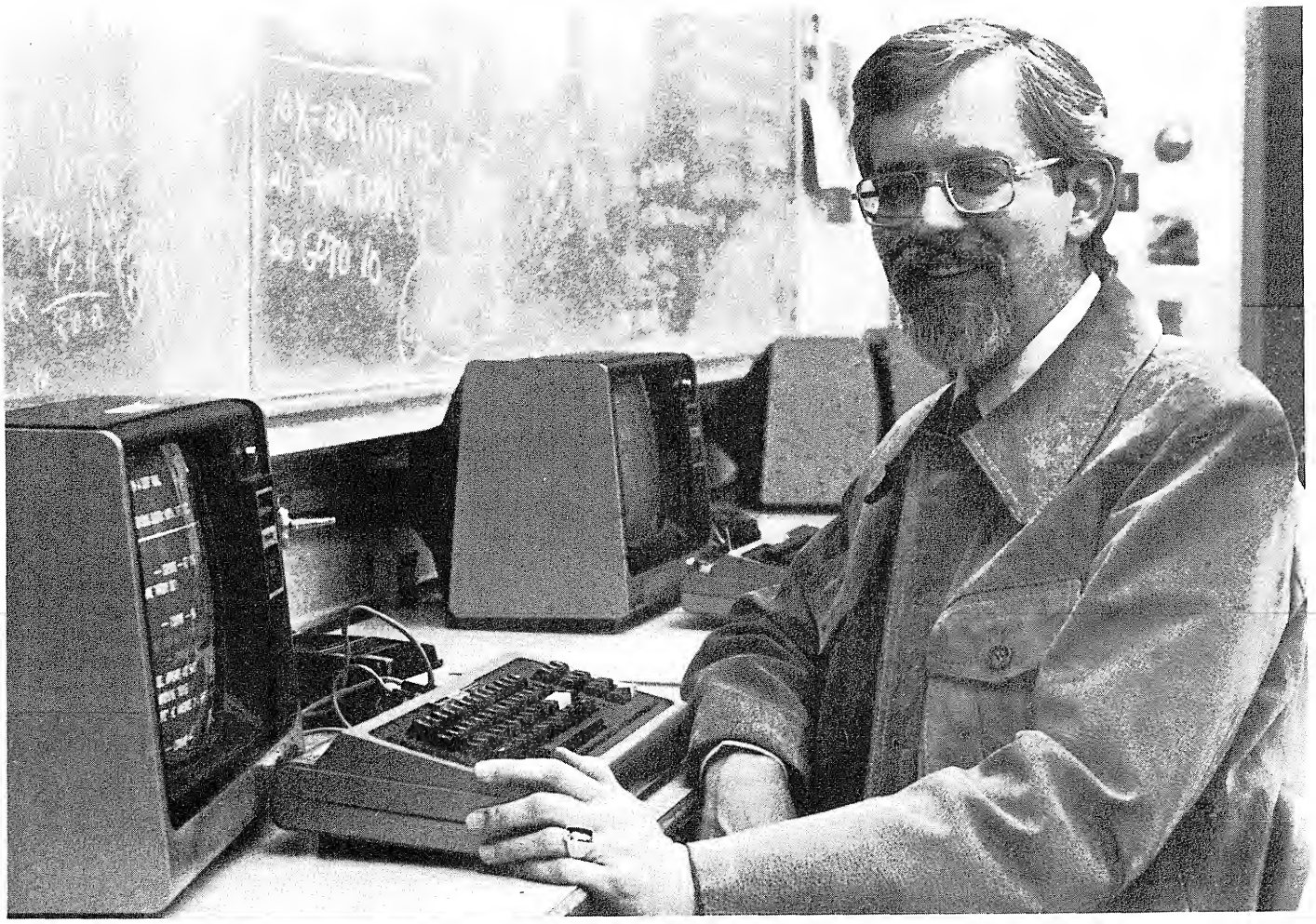
The demon does the softshoe to the strains of "Swanee River" and pertly bows at the end of his routine. He does his dance flawlessly, to the delight of all who watch.

Andy Android's brother Harvey, Lasersaber in hand, steps out to begin the tournament to the Celeste Sound version of "March of the Gladiators." As he battles for supremacy, the child who moved him through six practice fights watches breathlessly, rooting for this brave and tireless droid.

The droids, the bee, the spider, the demon, and those wacky, weird characters from Life Two seem alive. They have an amazing amount of vitality and personality. They move smoothly, and with a grace that belies their residence in a block graphic computer such as the TRS-80. They are the products of genius, talent and sensitivity. They are the children of Leo B. Christopherson.

Leo is a math and science teacher for Keithly Junior High School in Tacoma, WA. He also leads a Title I modified

Bob Liddil, The Programmer's Guild, Box 66, Petersborough, NH 03458.



math class which uses Radio Shack's K-8 Math Practice program to good advantage.

After school, at least four days of the week he tries to stay for the Computer Club. Called the Tabari Society, they have both closed meetings and a "game night," and so are able to meet the needs of most students wishing to participate. On game nights an admission fee of 25 cents is charged. The money is used to purchase computer related materials for the club.

Observers have noted that the young people in Leo's classes are "crazy about him." His quiet, intelligent approach to teaching serves as an inspiration to those with whom he comes in contact. He is organized and creative in his teaching, and these same traits are evident in the environment he creates for himself at home.

Leo is an accomplished musician. In his youth he played professionally in a band. Now, however, he contents himself with composing and executing ethnic specialty music on the organ and his Apple computer, which contains the most advanced hardware available for musical rendition. His organ and the amplification equipment that it employs, occupy most of his apartment.

Android Nim created a sensation in the commercial TRS-80 software market. Where most TRS-80 graphics programs were slow and herky-jerky, Nim was animated, and alive, with little eyes that stared ubiquitously, emotionally out from cloned Andys awaiting the executioner's laserbeam. Leo had brought the fine art of humoresque cartooning to the TRS-80.

Snake Eggs was next. It uses animated rattlesnakes, modelled after those in a Disney production of "Robin Hood," to illustrate the traditional game of "21." This program was also one of the first to incorporate sound effects, a technology that was in its infancy, and was, to a great extent, pioneered by Leo himself.

One of the most beloved of Leo's TRS-80 graphics presentations and without question the one most revealing of his subtle sense of humor, is Bee Wary. The Tarantula, the heavy in this slapstick micro-melodrama, has many faces and a wry vocabulary that leaves its audiences in pain from laughter. The Bee, controlled by the user, is much more mobile, and can deliver a deadly sting. But its opponent has several unexpected moves that can turn the bee into a hearty meal.

Life Two introduced a trio of new characters. A quadreped, a triped and a monoped, plus Andy representing bipeds added a new twist to the Game of Life. Billed as the Battle of Life, this presentation may be the most articulate portrayal of Life ever conceived.

Dancin' Demon used an extension of a graphics innovation technique called "line packing" as opposed to the "string packing" that characterized Leo's earlier efforts.

"These ideas have never been completely thought out in my mind before I start them," says Leo. "Rather, one completed part often suggests another new idea to try in the same program. I have yet to think of a concept to program without also coming up with a new technique, which often modifies the original idea of the program."

As a display, Demon does a fine job. But it is not enough to just have great graphics. The program must be interactive. And it is. The user can use an onboard music generator to program his own musical score. Thus, the actions of the Demon become more personalized. An added bonus is in the programmability of the dance steps. The user has a choice of endless combinations of dance routines. Through these features, the non-programmer joins Leo

in the often hilarious musical productions of Dancin' Demon.

Radio Shack, which is not well known for making fast decisions on submissions, acted very quickly upon receiving Dancin' Demons. In short order, Leo joined that elite fraternity of programmers whose works are mass marketed around the world.

"My newest program, Duel-n-Droids" says Leo, "uses the new graphics technique that I started for Demon but couldn't use due to memory space problems. The technique involves placing the droids on the screen so fast that one overlaps the other, creating the illusion of shading. The androids appear to be wearing uniforms and the display is rather alive."

"Droids also employs a technique I've wanted to try for some time. The player can 'teach' his android to fight before entering the tournament. I also found a place to try out some new sound and musical effects."

The musical effects Leo speaks of are a series of songs using the TRS-80 cassette port to produce a "Celeste" tone with vibrato that rivals music accessory boards.

Two new programs by Leo are in prerelease, at this writing. Snake Eggs II for the Apple is a mathematics drill application of the earlier game. Featuring the hires graphics available to Apple users, and the sound built into the computer, the result is a highly rewarding and potentially very popular program. Also, by popular request, our old friend Andy makes an Apple appearance in Apple Andy, to be known commercially as Apple Android Nim. It, too, stretches the Apple computer to its limits.

Sadly, at least from a commercial point of view, the new programs are destined for the same intense bootlegging that has plagued the TRS-80 releases. How does Leo feel about the bootleggers?

"I am concerned about the moral implications of software piracy," Leo comments sadly. "Most bootleggers aren't hard core pirates at all but family people with little children who use the computer daily. These children see the parent get something for nothing, and they quickly form similiar attitudes. The software traders and pirates set a very poor example for their kids."

There has developed, over the last couple of years, a public fascination with Leo's work, prompting talented young programmers to steer away from clever and highly animated techniques for fear of infringing on what has become known as "Christopherson Graphics."

Leo's reaction to those who would imitate his graphics style is one of encouragement. His attitude is "go for it." He believes that the animation style of programming belongs to anyone who is able to produce it. His own techniques are easily available to those who would use or improve them.

When asked what advice he has for new programmers who would like to be commercially successful, Leo mentioned six main points:

"Get to know your machine. The manufacturer usually publishes only a fraction of the routines available in your computer's particular language or chip machine code.

"When you do your program, write for *your* machine. Don't generalize in hopes of easy translation. Maximize your techniques to draw every last bit of power your machine is able to muster. Only then will you have your best program.

"Let ideas flow creatively. Build an atmosphere conducive to free thinking. Relax and allow your mind to work. Never throw out an idea; store it for a future application.

"Know the limitations of your machine. Never juryrig a routine or place your techniques in a position where failure of logic is possible.

"Expect months of hard work. Nothing comes easily, least of all production of commercial software. Short cuts and less than exhaustive debugging will cost you far more than the time you save. Be thorough.

"Never let inferior work leave your sight. You travel with each reproduction of your software, into thousands of computers. Never let less than perfect copies of your program get away from your control.

"Do not manufacture or attempt to sell your programs on your own unless you are prepared for heavy expense and disappointment or possibly even the loss of your work to unscrupulous companies. Find, instead, an agent or software company of high reputation that can either place your product or market it for you. These individuals or corporations have the knowhow to produce or place your program where it will make the most profit for you. But beware of grandiose promises. Check up on them before releasing anything."

Who is Leo B. Christopherson? He is Andy and The Spider, and a little bit of the Demon, for they are his creations and he travels with them. But most of all he is a gentle, quiet schooleacher from Tacoma, WA, who loves his kids as much as they love him. He is happy and spreads that happiness around. Who could ask for more. □

Appearing below and on the next few pages, is the complete listing of "Android Nim" by Leo Christopherson.

```
 1 REM  * COPYRIGHT 1978 80-NW PUBLISHING TACOMA, WA
   *
 2 REM  * ALL RIGHTS RESERVED *
 3 CLS :
   PRINT @20,"N O T I C E"
 4 PRINT :
   PRINT "IT IS ILLEGAL, UNFAIR & UNCOUTH TO COPY TH
   IS PGM FOR SOME"
 5 PRINT "ONE ELSE. IT CHEATS THE AUTHOR OF ROYALTIE
   S AND REMOVES ANY"
 6 PRINT "INCENTIVE TO PRODUCE MORE GOOD WORKS - THA
   NKS"
 8 INPUT "PUSH ENTER";
 9 CLS :
   RUN 10020
10 CLEAR 525:
   DEFINT X,Y,Z:
   RANDOM
25 GOTO 900
50 MD$ = "RIDICULOUS":
   RETURN
51 MD$ = "ABSURD":
   RETURN
52 MD$ = "GROTESQUE":
   RETURN
53 MD$ = "NONSENSICAL":
   RETURN
54 MD$ = "FARCICAL":
   RETURN
55 MD$ = "PREPOSTEROUS":
   RETURN
56 MD$ = "SILLY":
   RETURN
57 MD$ = "SENSELESS":
   RETURN
58 MD$ = "IRRATIONAL":
   RETURN
```

```
 59 MD$ = "FANTASTIC":
    RETURN
 60 MD$ = "ODD":
    RETURN
 61 MD$ = "RUDE":
    RETURN
 62 MD$ = "BRUTISH":
    RETURN
 63 MD$ = "BARBARIC":
    RETURN
 64 MD$ = "PLEBEIAN":
    RETURN
 65 MD$ = "UNCIVIL":
    RETURN
 66 MD$ = "DISCOURTEOUS":
    RETURN
 67 MD$ = "VULGAR":
    RETURN
 68 MD$ = "COARSE":
    RETURN
 69 MD$ = "GROSS":
    RETURN
 70 MD$ = "UNGRACEFUL":
    RETURN
 71 MD$ = "MONSTROUS":
    RETURN
 72 MD$ = "HORRID":
    RETURN
 73 MD$ = "SHOCKING":
    RETURN
 74 MD$ = "CHEAP":
    RETURN
900 GOSUB 9000
1000 X9 = 1:
     CLS :
     PRINT @25, CHR$(34);"ANDROID NIM"; CHR$(34);
1005 X = 194:
     GOSUB 8000:
     PRINT @X,BQ$;
1010 X = 517:
     GOSUB 8000:
     PRINT @X,BQ$;
1011 X = 835:
     GOSUB 8000:
     PRINT @X,BQ$;
1012 Y9 = 1:
     RW$(1) = "0":
     RW$(2) = "0":
     RW$(3) = "0"
1013 GOSUB 1040:
     GOTO 1050
1016 X = 254:
     FOR N = 1 TO 7:
      X = X - 7:
      GOSUB 8000:
      NEXT N:
     X = 570:
     FOR N = 1 TO 5:
      X = X - 8:
      GOSUB 8000:
      NEXT N:
     X = 885:
     FOR N = 1 TO 3:
      X = X - 9:
      GOSUB 8000:
      NEXT N
1017 RW$(1) = "7":
     RW$(2) = "5":
     RW$(3) = "3"
1018 GOTO 8200
1040 FOR N = 77 TO 333 STEP 64:
      PRINT @N, CHR$(30);:
      NEXT N:
     FOR N = 402 TO 658 STEP 64:
      PRINT @N, CHR$(30);:
      NEXT N:
     FOR N = 730 TO 986 STEP 64:
      PRINT @N, CHR$(30);:
      NEXT N:
     PRINT @22," ";:
     RETURN
1050 PRINT @540,"FIRST MOVE BY YOU (1) OR ME (2) ?";
1054 GOTO 1060
1055 X = 194:
     GOSUB 8005:
     X = 517:
     GOSUB 8005:
     X = 835:
     GOSUB 8005:
     RETURN
1060 GOSUB 8070:
     K$ = INKEY$:
     IF K$ = ""
      THEN
       1060:
      ELSE
       IF K$ = "1"
        THEN
         1110:
        ELSE
         IF K$ = "2"
          THEN
           1075:
          ELSE
           IF K$ = "Y"
            THEN
             1064:
            ELSE
             IF K$ = "M"
              THEN
               1099:
              ELSE
               1060
1064 GOTO 1065
1065 E$ = INKEY$:
     IF E$ = ""
      THEN
       1065:
      ELSE
       1069
1069 GOSUB 1055
1070 Y$ = INKEY$:
     IF Y$ = ""
      THEN
       1070:
      ELSE
       K$ = K$ + E$ + Y$:
       IF K$ = "YOU"
        THEN
         1075:
        ELSE
         1500
1075 X9 = 2:
     PRINT @540,"OK, I'LL START !                    ";:
     FOR O = 1 TO 12:
      GOSUB 8070:
      NEXT O:
     PRINT @540,"                                    ";:
     GOTO 1016
1099 GOSUB 1055
1100 E$ = INKEY$:
     IF E$ = ""
      THEN
       1100:
      ELSE
       K$ = K$ + E$:
       IF K$ = "ME"
        THEN
         1110:
        ELSE
         1500
1110 X9 = 1:
     PRINT @540," VERY WELL , YOU MAY START !        ";:
     X = 517:
     GOSUB 8010:
     X = 835:
     GOSUB 8010:
     X = 194:
     GOSUB 8010:
     FOR M = 1 TO 12:
      GOSUB 8070:
      NEXT M:
     PRINT @540,"                                    ";:
     GOTO 1016
1500 PRINT @540,"WHICH ONE OF US IS , "; CHR$(34);K$;
     CHR$(34);" ?       ";:
     X = 835:
     GOSUB 8010:
```

```
      X = 194:
      GOSUB 8010:
      X = 517:
      GOSUB 8010:
      GOSUB 8112:
      GOTO 1050
 7000 PRINT @22," ";
 7005 ON Y9 GOTO 7020,7010,7010
 7006 GOTO 7200
 7010 M = RND(2):
      ON M GOTO 7020,7200
 7020 FOR O = 1 TO 12:
       GOSUB 8070:
       NEXT O:
      GOTO 7275
 7200 Z = 1:
      GOSUB 7500:
      Z1 = Z0:
      Z = 2:
      GOSUB 7500:
      Z2 = Z0:
      Z = 3:
      GOSUB 7500:
      Z3 = Z0
 7210 Z4 = Z1 + Z2:
      Z5 = Z1 + Z3:
       Z6 = Z2 + Z3:
       Z4$ = STR$(Z4):
       Z5$ = STR$(Z5):
       Z6$ = STR$(Z6)
 7220 GOSUB 8070:
       IF Z4 = 0
        THEN
         7230:
        ELSE
         Z0 = LEN(Z4$):
         Y0 = 1:
         Z4 = 0:
         FOR N = Z0 TO 1 STEP - 1
 7222    IF MID$(Z4$,N,1) = "1"
          THEN
           Z4 = Z4 + Y0
 7223    Y0 = Y0 * 2:
        NEXT N
 7230 GOSUB 8070:
       IF Z5 = 0
        THEN
         7240:
        ELSE
         Z0 = LEN(Z5$):
         Y0 = 1:
         Z5 = 0:
         FOR N = Z0 TO 1 STEP - 1
 7232    IF MID$(Z5$,N,1) = "1"
          THEN
           Z5 = Z5 + Y0
 7233    Y0 = Y0 * 2:
        NEXT N
 7240 GOSUB 8070:
       IF Z6 = 0
        THEN
         7250:
        ELSE
         Z0 = LEN(Z6$):
         Y0 = 1:
         Z6 = 0:
         FOR N = Z0 TO 1 STEP - 1
 7242    IF MID$(Z6$,N,1) = "1"
          THEN
           Z6 = Z6 + Y0
 7243    Y0 = Y0 * 2:
        NEXT N
 7245 GOSUB 8070
 7250 IF VAL(RW$(3)) < = Z4
       THEN
        7255
 7251 RW = 3:
      GOSUB 8430:
      RN = VAL(RW$(3)) - Z4:
      RN$ = STR$(RN):
      RN$ = RIGHT$(RN$,1):
      X9 = 1:
      GOTO 8240
 7255 IF VAL(RW$(2)) < = Z5
       THEN
        7260
 7256 RW = 2:
      GOSUB 8420:
      RN = VAL(RW$(2)) - Z5:
      RN$ = STR$(RN):
      RN$ = RIGHT$(RN$,1):
      X9 = 1:
      GOTO 8240
 7260 IF VAL(RW$(1)) < = Z6
       THEN
        7270
 7261 RW = 1:
      GOSUB 8410:
      RN = VAL(RW$(1)) - Z6:
      RN$ = STR$(RN):
      RN$ = RIGHT$(RN$,1):
      X9 = 1:
      GOTO 8240
 7270 GOSUB 8070:
      X8 = 0:
      IF (Z4 = 0) AND (Z5 = 0)
       THEN
        7300
 7275 M = RND(3)
 7276 ON M GOTO 7277,7278,7279
 7277 IF VAL(RW$(1)) = 0
       THEN
        7278:
       ELSE
        RW = 1:
        GOSUB 7290:
        GOSUB 8410:
        X9 = 1:
        X8 = 0:
        GOTO 8240
 7278 IF VAL(RW$(2)) = 0
       THEN
        7279:
       ELSE
        RW = 2:
        GOSUB 7290:
        GOSUB 8420:
        X9 = 1:
        X8 = 0:
        GOTO 8240
 7279 IF VAL(RW$(3)) = 0
       THEN
        7277:
       ELSE
        RW = 3:
        GOSUB 7290:
        GOSUB 8430:
        X9 = 1:
        X8 = 0:
        GOTO 8240
 7290 M = RND(7):
      ON M GOTO 7291,7292,7293,7294,7295,7296,7297
 7291 RN = 1:
      RN$ = "1":
      RETURN
 7292 GOTO 7291
 7293 GOTO 7291
 7294 GOTO 7291
 7295 RN = 2:
      RN$ = "2":
      RETURN
 7296 RN = 3:
      RN$ = "3":
      RETURN
 7297 IF RW=3
       THEN
        7296
       ELSE
        RN = 4 :
        RN$ = "4" :
      RETURN
 7300 GOTO 7350
 7350 GOSUB 1040
 7360 IF X9 = 1
       THEN
        7400
 7370 GOSUB 7600:
```

```
      GOSUB 8070:
      X = 194:
      GOSUB 8035:
      GOSUB 8114:
      X = 835:
      GOSUB 8035:
      GOSUB 8116:
      X = 517:
      GOSUB 8035:
      GOSUB 8114:
      PRINT @453,H5$;:
      GOSUB 8116:
      PRINT @771,H5$;:
      GOSUB 8116:
      PRINT @130,H5$;:
      GOSUB 8114:
      PRINT @130,HR$;
7375  GOSUB 8114:
      PRINT @771,HR$;:
      GOSUB 8116:
      PRINT @453,HR$;:
      GOSUB 8114:
      GOTO 7420
7400  GOSUB 1055:
      PRINT @536,MB$;:
      PRINT @472,MA$;:
      PRINT @600,MC$;:
      GOSUB 8112:
      GOSUB 8112:
      GOTO 7420
7420  PRINT @794,"NEW GAME?    YES (1) OR NO (2)";
7425  GOSUB 8070:
      K$ = INKEY$:
      IF K$ = ""
       THEN
        7425:
       ELSE
         IF K$ = "1"
          THEN
           1012:
          ELSE
           7430
7430  CLS :
      GOTO 3
7450  GOSUB 1040:
      RW$(1) = "0":
      RW$(2) = "0":
      RW$(3) = "0":
      PRINT @540,"SINCE YOU HAVE GIVEN UP ... ";:
      FOR O = 1 TO 8:
       GOSUB 8070:
       NEXT O:
      GOSUB 1040:
      GOTO 7400
7500  ON VAL(RW$(Z)) + 1 GOTO 7505,7510,7515,7520,7525,
      7530,7535,7540
7505  Z0 = 0:
      RETURN
7510  Z0 = 1:
      RETURN
7515  Z0 = 10:
      RETURN
7520  Z0 = 11:
      RETURN
7525  Z0 = 100:
      RETURN
7530  Z0 = 101:
      RETURN
7535  Z0 = 110:
      RETURN
7540  Z0 = 111:
      RETURN
7600  GOSUB 7700:
      W1$ = MD$
7601  GOSUB 7700:
      IF MD$ = W1$
       THEN
        7601:
       ELSE
        W2$ = MD$
7602  GOSUB 7700:
      IF (MD$ = W1$) OR (MD$ = W2$)
       THEN
        7602:
       ELSE
        W3$ = MD$
7603  GOSUB 7700:
      IF (MD$ = W1$) OR (MD$ = W2$) OR (MD$ = W3$)
       THEN
        7603:
       ELSE
        W4$ = MD$
7604  GOSUB 7700:
      IF (MD$ = W1$) OR (MD$ = W2$) OR (MD$ = W3$)
      OR (MD$ = W4$)
       THEN
        7604:
       ELSE
        W5$ = MD$
7605  GOSUB 7700:
      IF (MD$ = W1$) OR (MD$ = W2$) OR (MD$ = W3$)
      OR (MD$ = W4$) OR (MD$ = W5$)
       THEN
        7605:
       ELSE
        W6$ = MD$
7606  GOSUB 7700:
      IF (MD$ = W1$) OR (MD$ = W2$) OR (MD$ = W3$)
      OR (MD$ = W4$) OR (MD$ = W5$) OR (MD$ = W6$)
       THEN
        7606:
       ELSE
        W7$ = MD$
7610  PRINT @412,"THROUGH SOME ";W1$ + ",";:
      GOSUB 8118:
      PRINT @470,W2$ + ",";:
      GOSUB 8118:
      PRINT @470,W2$ + "," + W3$ + ",";:
      GOSUB 8118:
      PRINT @470,W2$ + "," + W3$ + "," + W4$ + ",";:
      GOSUB 8118
7620  PRINT @534,W5$ + ",";:
      GOSUB 8070:
      PRINT @534,W5$ + "," + W6$ + ",";:
      GOSUB 8070:
      PRINT @534,W5$ + "," + W6$ + "," + W7$ + ",";:
      GOSUB 8070
7630  PRINT @610,"STROKE OF FATE...";:
      GOSUB 8118:
      PRINT @678,"YOU WIN !";
7640  RETURN
7700  M = RND(25)
7705  ON M GOSUB 50,51,52,53,54,55,56,57,58,59,60,61,62
      ,63,64,65,66,67,68,69,70,71,72,73,74
7710  RETURN
7951  IF VAL(RW$(1)) < 7
       THEN
        7952:
       ELSE
        X = 205:
        RETURN
7952  IF VAL(RW$(1)) < 6
       THEN
        7953:
       ELSE
        X = 212:
        GOTO 7975
7953  IF VAL(RW$(1)) < 5
       THEN
        7954:
       ELSE
        X = 219:
        RETURN
7954  IF VAL(RW$(1)) < 4
       THEN
        7955:
       ELSE
        X = 226:
        RETURN
7955  IF VAL(RW$(1)) < 3
       THEN
        7956:
       ELSE
        X = 233:
        GOTO 7979
7956  IF VAL(RW$(1)) < 2
       THEN
        7957:
       ELSE
```

```
       X = 240:
       RETURN
7957 IF VAL(RW$(1)) < 1
      THEN
       7958:
      ELSE
       X = 247:
       RETURN
7958 X = 194:
     RETURN
7961 IF VAL(RW$(2)) < 5
      THEN
       7962:
      ELSE
       X = 530:
       RETURN
7962 IF VAL(RW$(2)) < 4
      THEN
       7963:
      ELSE
       X = 538:
       GOTO 7983
7963 IF VAL(RW$(2)) < 3
      THEN
       7964:
      ELSE
       X = 546:
       GOTO 7988

7964 IF VAL(RW$(2)) < 2
      THEN
       7965:
      ELSE
       X = 554:
       RETURN
7965 IF VAL(RW$(2)) < 1
      THEN
       7966:
      ELSE
       X = 562:
       RETURN
7966 X = 517:
     RETURN
7971 IF VAL(RW$(3)) < 3
      THEN
       7972:
      ELSE
       X = 858:
       RETURN
7972 IF VAL(RW$(3)) < 2
      THEN
       7973:
      ELSE
       X = 867:
       GOTO 7992
7973 IF VAL(RW$(3)) < 1
      THEN
       7974:
      ELSE
       X = 876:
       RETURN
7974 X = 835:
     RETURN
7975 N = RND(3):
     ON N GOTO 7976,7977,7978
7976 RETURN
7977 PRINT @X - 64,HP$;:
     PRINT @X - 127,HO$;:
     PRINT @162,HJ$;:
     PRINT @99,HI$;:
     PRINT @169,HD$;:
     PRINT @106,HC$;:
     RETURN
7978 PRINT @X - 64,HD$;:
     PRINT @X - 127,HC$;:
     PRINT @176,HB$;:
     PRINT @113,HA$;:
     RETURN
7979 N = RND(3):
     ON N GOTO 7980,7981,7982
7980 RETURN
7981 PRINT @X - 64,HR$;:
     PRINT @X - 127,HQ$;:
     PRINT @176,HJ$;:
     PRINT @113,HI$;:
     RETURN
7982 PRINT @X - 64,HD$;:
     PRINT @X - 127,HC$;:
     PRINT @183,HJ$;:
     PRINT @120,HI$;:
     RETURN
7983 N = RND(4):
     ON N GOTO 7984,7985,7986
7984 RETURN

7985 PRINT @X - 64,HD$;:
     PRINT @X - 127,HC$;:
     PRINT @482,HJ$;:
     PRINT @419,HI$;:
     PRINT @498,HB$;:
     PRINT @435,HA$;:
     RETURN

7986 PRINT @X - 64,HB$;:
     PRINT @X - 127,HA$;:
     PRINT @482,HP$;:
     PRINT @419,HO$;:
     RETURN
7987 PRINT @X - 64,HR$;:
     PRINT @X - 127,HQ$;:
     PRINT @490,HD$;:
     PRINT @427,HC$;:
     RETURN
7988 N = RND(3):
     ON N GOTO 7989,7990,7991
7989 RETURN
7990 PRINT @X - 64,HD$;:
     PRINT @X - 127,HC$;:
     PRINT @490,HJ$;:
     PRINT @427,HI$;:
     RETURN
7991 PRINT @X - 64,HB$;:
     PRINT @X - 127,HA$;:
     PRINT @498,HP$;:
     PRINT @435,HO$;:
     RETURN
7992 N = RND(6):
     ON N GOTO 7993,7994,7995,7996,7997,7993
7993 RETURN
7994 PRINT @X - 64,HP$;:
     PRINT @X - 127,HO$;:
     PRINT @812,HR$;:
     PRINT @749,HQ$;:
     RETURN
7995 IF X3 < > 0
      THEN
       RETURN :
      ELSE
       PRINT @X - 64,HD$;:
       PRINT @X - 127,HC$;:
       PRINT @130,HJ$;:
       PRINT @67,HI$;:
       RETURN
7996 IF X3 < > 0
      THEN
       RETURN :
      ELSE
       PRINT @X - 64,HB$;:
       PRINT @X - 127,HA$;:
       PRINT @453,HP$;:
       PRINT @390,HO$;:
       RETURN
7997 IF X3 < > 0
      THEN
       RETURN :
      ELSE
       PRINT @X - 64,HD$;:
       PRINT @X - 127,HC$;:
       PRINT @771,HJ$;:
       PRINT @708,HI$;:
       RETURN
8000 PRINT @X,BA$;:
     PRINT @X + 64,BB$;:
     PRINT @X - 64,HB$;:
     PRINT @X - 127,HA$;:
     PRINT @X + 128,FC$;:
     RETURN
```

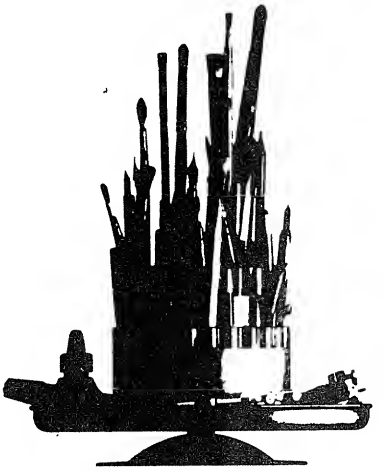
```
8005 GOSUB 8100:
     PRINT @X - 64,HD$;:
     PRINT @X - 127,HC$;:
     GOSUB 8100:
     PRINT @X - 64,HF$;:
     PRINT @X - 127,HE$;:
     GOSUB 8100:
     PRINT @X - 64,HH$;:
     PRINT @X - 127,HG$;:
     RETURN
8010 GOSUB 8100:
     PRINT @X - 64,HF$;:
     PRINT @X - 127,HE$;:
     GOSUB 8100:
     PRINT @X - 64,HD$;:
     PRINT @X - 127,HC$;:
     GOSUB 8100:
     PRINT @X - 64,HB$;:
     PRINT @X - 127,HA$;:
     RETURN
4
8015 GOSUB 8100:
     PRINT @X - 64,HJ$;:
     PRINT @X - 127,HI$;:
     GOSUB 8100:
     PRINT @X - 64,HL$;:
     PRINT @X - 127,HK$;:
     GOSUB 8100:
     PRINT @X - 64,HN$;:
     PRINT @X - 127,HM$;:
     RETURN
8020 GOSUB 8100:
     PRINT @X - 64,HL$;:
     PRINT @X - 127,HK$;:
     GOSUB 8100:
     PRINT @X - 64,HJ$;:
     PRINT @X - 127,HI$;:
     GOSUB 8100:
     PRINT @X - 64,HB$;:
     PRINT @X - 127,HA$;:
     RETURN
8025 M = RND(2) + 1:
     FOR L = 1 TO M:
      GOSUB 8100:
      PRINT @X - 64,HJ$;:
      PRINT @X - 127,HI$;:
      GOSUB 8100:
      PRINT @X - 64,HB$;:
      PRINT @X - 127,HA$;:
      GOSUB 8100:
      PRINT @X - 64,HD$;:


      PRINT @X - 127,HC$;:
      GOSUB 8100:
      PRINT @X - 64,HB$;:
      PRINT @X - 127,HA$;:
      NEXT L:
     RETURN
8030 GOSUB 8100:
     PRINT @X - 64,HP$;:
     PRINT @X - 127,HO$;:
     RETURN
8035 GOSUB 8100:
     PRINT @X - 64,HR$;:
     PRINT @X - 127,HQ$;:
     RETURN
8040 M = RND(3):
     FOR L = 1 TO M:
      GOSUB 8030:
      GOSUB 8100:
      PRINT @X - 64,HB$;:
      PRINT @X - 127,HA$;:
      GOSUB 8035:
      GOSUB 8100:
      PRINT @X - 64,HB$;:
      PRINT @X - 127,HA$;:
      NEXT L:
     RETURN
8045 GOSUB 8100:
     PRINT @X + 4,BE$;:
     PRINT @X + 68,BF$;:
     GOSUB 8100:
     PRINT @X + 4,BG$;:
     PRINT @X + 68,BH$;:
     GOSUB 8100:
     PRINT @X + 4,BI$;:
     PRINT @X + 68,BJ$;:
     GOSUB 8100:
     PRINT @X + 4,BK$;:
     GOSUB 8100:
     PRINT @X + 4,BL$;:
     RETURN
8050 GOSUB 8100:
     PRINT @X + 4,BK$;:
     X3 = X:
     GOSUB 8070:
     GOSUB 8070:
     X = X3:
     GOSUB 8110:
     PRINT @X + 4,BI$;:
     PRINT @X + 68,BJ$;:
     GOSUB 8100:
     PRINT @X + 4,BG$;:
     PRINT @X + 68,BH$;:
     GOSUB 8100:
     PRINT @X + 4,BE$;:
     PRINT @X + 68,BF$;:
     GOSUB 8110:
     PRINT @X + 4,BC$;:
     PRINT @X + 68,BD$;:
     RETURN
8055 GOSUB 8100:
     PRINT @X - 64,HD$;:
     PRINT @X - 127,HC$;:
     RETURN
8060 GOSUB 8100:
     PRINT @X - 64,HJ$;:
     PRINT @X - 127,HI$;:
     RETURN
8065 GOSUB 8100:
     PRINT @X - 64,HB$;:
     PRINT @X - 127,HA$;:
     RETURN
8070 M = RND(16):
     K = RND(50) + 50:
     Y = RND(18):
     GOSUB 8075:
     IF X3 = X
      THEN
       8070:
      ELSE
       IF X3 = 0
        THEN
         8071:
        ELSE
         IF (M < 8) OR (M > 12)
          THEN
           8071:
          ELSE
           8070
8071 IF M > 7
      THEN
       8089:
      ELSE
       ON M GOSUB 8005,8015,8030,8035,8055,8060,8065
8074 GOTO 8079
8075 ON Y GOTO 8076,8077,8078,7951,7952,7953,7954,7955
     ,7956,7957,7961,7962,7963,7964,7965,7971,7972,797
     3
8076 X = 194:
     RETURN
8077 X = 517:
     RETURN
8078 X = 835:
     RETURN
8079 X1 = RND(5)
8080 ON X1 GOTO 8081,8070,8081,8070,8081
8081 ON M GOTO 8087,8088,8082,8083,8084,8085,8086
8082 PRINT @X - 127,H4$;:
     GOSUB 8100:
     PRINT @X - 127,HO$;:
     RETURN
8083 PRINT @X - 64,H5$;:
     GOSUB 8100:
     PRINT @X - 64,HR$;:
     RETURN
```

```
8084 PRINT @X - 64,H2$;:
     GOSUB 8100:
     PRINT @X - 64,HD$;:
     RETURN
8085 PRINT @X - 64,H3$;:
     GOSUB 8100:
     PRINT @X - 64,HJ$;:
     RETURN
8086 PRINT @X - 64,HI$;:
     GOSUB 8100:
     PRINT @X - 64,HB$;:
     RETURN
8087 PRINT @X - 64,H6$;:
     GOSUB 8100:
     PRINT @X - 64,HH$;:
     RETURN
8088 PRINT @X - 64,H7$;:
     GOSUB 8100:
     PRINT @X - 64,HN$;:
     RETURN
8089 ON M - 7 GOTO 8090,8091,8092,8093,8094,8095,8096,
     8097,8098
8090 PRINT @67,HO$;:
     PRINT @130,HP$;:
     PRINT @390,HC$;:
     PRINT @453,HD$;:
     GOSUB 8100:
     RETURN
8091 PRINT @67,HQ$;:
     PRINT @130,HR$;:
     PRINT @708,HI$;:
     PRINT @771,HJ$;:
     GOSUB 8100:
     RETURN
8092 PRINT @390,HA$;:
     PRINT @453,HB$;:
     PRINT @708,HO$;:
     PRINT @771,HP$;:
     GOSUB 8100:
     RETURN
8093 PRINT @67,HQ$;:
     PRINT @130,HR$;:
     PRINT @390,HI$;:
     PRINT @453,HJ$;:
     PRINT @708,HO$;:
     PRINT @771,HP$;:
     GOSUB 8100:
     RETURN
8094 PRINT @390,HO$;:
     PRINT @453,HP$;:
     PRINT @67,HQ$;:
     PRINT @130,HR$;:
     GOSUB 8100:
     RETURN
8095 IF (X = 194) OR (X = 517) OR (X = 835)
      THEN
       8120:
      ELSE
       PRINT @X,BM$;:
       PRINT @X + 64,BN$;:
       GOSUB 8100:
       RETURN
8096 IF (X = 194) OR (X = 517) OR (X = 835)
      THEN
       8130:
      ELSE
       PRINT @X,BA$;:
       PRINT @X + 64,BB$;:
       GOSUB 8100:
       RETURN
8097 PRINT @X + 4,BO$;:
     PRINT @X + 68,BP$;:
     GOSUB 8100:
     RETURN
8098 PRINT @X + 4,BG$;:
     PRINT @X + 68,BH$;:
     GOSUB 8100:
     RETURN
8100 FOR N = 1 TO 8:
      NEXT N:
     RETURN
8110 RETURN :
     FOR N = 1 TO 50:
      NEXT N:
     RETURN
8112 FOR N = 1 TO 400:
      NEXT N:
     RETURN
8114 FOR N = 1 TO 200:
      NEXT N:
     RETURN
8116 FOR N = 1 TO 100:
      NEXT N:
     RETURN
8118 FOR O = 1 TO 3:
      GOSUB 8070:
      NEXT O:
     RETURN
8120 PRINT @X,BR$;:
     PRINT @X + 64,BN$;:
     GOSUB 8100:
     RETURN
8130 PRINT @X,BQ$;:
     PRINT @X + 64,BB$;:
     GOSUB 8100:
     RETURN
8200 X8 = 0:
     IF X9 = 2
      THEN
       7000:
      ELSE
       X3 = 0:
       PRINT @22,"*";:
       GOSUB 8070:
       K$ = INKEY$:
       IF K$ = ""
        THEN
         8200:
        ELSE
         IF (K$ > "0") AND (K$ < "4")
          THEN
           8210:
          ELSE
           IF K$ = "R"
            THEN
             7450:
            ELSE
             8200
8210 RW = VAL(K$):
     ON RW GOSUB 8410,8420,8430
8220 GOSUB 8070:
     K$ = INKEY$:
     IF K$ = ""
      THEN
       8220:
      ELSE
       IF (K$ > "0") AND (K$ < "8")
        THEN
         8225:
        ELSE
         IF K$ = " "
          THEN
           GOSUB 8325:
          ELSE
           8220
8221 GOTO 8200
8225 X9 = 2
8230 RN$ = K$:
     RN = VAL(RN$)
8240 IF (RN > VAL(RW$(RW))) AND (X9 = 1)
      THEN
       X8 = 1
8243 IF (RN > VAL(RW$(RW))) AND (X9 = 2)
      THEN
       X9 = 1
8245 ON RW GOSUB 8260,8270,8280
8247 IF ( VAL(RW$(1)) = 0) AND ( VAL(RW$(2)) = 0)
     AND ( VAL(RW$(3)) = 0)
      THEN
       7300
8250 IF X8 = 0
      THEN
       8200:
      ELSE
       7270
8260 PRINT @128,RN$;:
     X = 194:
     GOTO 8300
```

```
8270 PRINT @448,RN$;:
     X = 517:
     GOTO 8300
8280 PRINT @768,RN$;:
     X = 835:
     GOTO 8300
8300 GOSUB 8015:
     X3 = X:
     GOSUB 8070:
     X = X3:
     GOSUB 8060:
     GOSUB 8005:
     X3 = X:
     GOSUB 8070:
     X = X3:
     GOSUB 8010
8310 IF RN > VAL(RW$(RW))
      THEN
       8320:
      ELSE
       8400
8320 GOSUB 8025
8325 ON RW GOTO 8330,8340,8350
8330 PRINT @128," ";:
     PRINT @192," ";:
     RETURN
8340 PRINT @448," ";:
     PRINT @512," ";:
     RETURN
8350 PRINT @768," ";:
     PRINT @832," ";:
     RETURN
8400 GOTO 8405:
     GOSUB 8040:
     X3 = X:
     GOSUB 8070:
     X = X3:
     GOSUB 8005:
     X3 = X:
     GOSUB 8070:
     GOSUB 8070:
     X = X3:
     GOSUB 8045:
     GOSUB 8450:
     NEXT N:
     GOSUB 8500:
     GOSUB 8050:
     GOSUB 8010:
     GOTO 8325
8405 GOSUB 8040:
     GOSUB 8005:
     X3 = X:
     GOSUB 8070:
     X = X3:
     GOSUB 8045:
     GOSUB 8450:
     GOSUB 8500:
     GOSUB 8050:
     GOSUB 8010:
     GOTO 8325
8410 PRINT @192,"1";:
     RETURN
8420 PRINT @512,"2";:
     RETURN
8430 PRINT @832,"3";:
     RETURN
8450 ON RW GOTO 8460,8470,8480
8460 X = 254:
     IF VAL(RW$(1)) = 0
      THEN
       8461:
      ELSE
       FOR O = 1 TO VAL(RW$(1)):
        X = X - 7:
        GOSUB 8015:
        NEXT O
8461 X = 517:
     GOSUB 8030:
     X = 570:
     IF VAL(RW$(2)) = 0
      THEN
       8462:
      ELSE
       FOR O = 1 TO VAL(RW$(2)):
        X = X - 8:
        GOSUB 8030:
        NEXT O
8462 X = 835:
     GOSUB 8030:
     X = 885:
     IF VAL(RW$(3)) = 0
      THEN
       8463:
      ELSE
       FOR O = 1 TO VAL(RW$(3)):
        X = X - 9:
        GOSUB 8030:
        NEXT O
8463 X = 194:
     RETURN
8470 X = 570:
     IF VAL(RW$(2)) = 0
      THEN
       8471:
      ELSE
       FOR O = 1 TO VAL(RW$(2)):
        X = X - 8:
        GOSUB 8015:
        NEXT O
8471 X = 194:
     GOSUB 8035:
     X = 254:
     IF VAL(RW$(1)) = 0
      THEN
       8472:
      ELSE
       FOR O = 1 TO VAL(RW$(1)):
        X = X - 7:
        GOSUB 8035:
        NEXT O
8472 X = 835:
     GOSUB 8030:
     X = 885:
     IF VAL(RW$(3)) = 0
      THEN
       8473:
      ELSE
       FOR O = 1 TO VAL(RW$(3)):
        X = X - 9:
        GOSUB 8030:
        NEXT O
8473 X = 517:
     RETURN
8480 X = 885:
     IF VAL(RW$(3)) = 0
      THEN
       8481:
      ELSE
       FOR O = 1 TO VAL(RW$(3)):
        X = X - 9:
        GOSUB 8015:
        NEXT O
8481 X = 517:
     GOSUB 8035:
     X = 570:
     IF VAL(RW$(2)) = 0
      THEN
       8482:
      ELSE
       FOR O = 1 TO VAL(RW$(2)):
        X = X - 8:
        GOSUB 8035:
        NEXT O
8482 X = 194:
     GOSUB 8035:
     X = 254:
     IF VAL(RW$(1)) = 0
      THEN
       8483:
      ELSE
       FOR O = 1 TO VAL(RW$(1)):
        X = X - 7:
        GOSUB 8035:
        NEXT O
```

```
8483 X = 835:
     RETURN
8500 Z1 = VAL(RW$(RW)):
     Z2 = Z1 - RN:
     RW$(RW) = STR$(Z1 - RN)
8520 ON RW GOTO 8530,8540,8550
8530 Z$ = "===-":
     Z3 = 205:
     Z4 = 7:
     Z5 = 254 - 7 * Z1:
     GOTO 8560
8540 Z$ = "===-":
     Z3 = 528:
     Z4 = 8:
     Z5 = 570 - 8 * Z1:
     GOTO 8560
8550 Z$ = "===-":
     Z3 = 846:
     Z4 = 9:
     Z5 = 885 - 9 * Z1:
     GOTO 8560
8560 FOR M = 1 TO RN:
      PRINT @Z3 - 1,"=";:
      FOR N = Z3 TO Z5 STEP 3:
       PRINT @Z3 - 1,"-";:
       PRINT @N,Z$;:
       PRINT @Z3 - 1,"=";:
       NEXT N:
      PRINT @Z5,FA$;:
      PRINT @Z5 + 64,FA$;:
      PRINT @Z5 - 64,FA$;:
      PRINT @Z5 + 128,FA$;:
      PRINT @Z5 - 128,FA$;
8570  PRINT @Z5,FB$;:
      PRINT @Z5 + 64,FB$;:
      PRINT @Z5 - 64,FB$;:
      PRINT @Z5 + 128,FB$;:
      PRINT @Z5 - 128,FB$;:
      FOR N = Z3 TO Z5 STEP 3:
       PRINT @N,"    ";:
       NEXT N
8580  Z5 = Z5 + Z4:
      NEXT M
8590 Y9 = Y9 + 1:
     X3 = X:
     GOSUB 8070:
     X = X3:
     RETURN
9000 HA$ = CHR$(160) + CHR$(181) + CHR$(186) +
     CHR$(144)
9001 HB$ = CHR$(128) + CHR$(143) + CHR$(182) +
     CHR$(185) + CHR$(143) + CHR$(128)
9002 H1$ = CHR$(128) + CHR$(143) + CHR$(183) +
     CHR$(187) + CHR$(143)
9005 HC$ = CHR$(160) + CHR$(186) + CHR$(176) +
     CHR$(149)
9006 HD$ = CHR$(128) + CHR$(143) + CHR$(189) +
     CHR$(179) + CHR$(142) + CHR$(128)
9007 H2$ = CHR$(128) + CHR$(143) + CHR$(191) +
     CHR$(179) + CHR$(143)
9010 HE$ = CHR$(160) + CHR$(176) + CHR$(181) +
     CHR$(149)
9011 HF$ = CHR$(128) + CHR$(143) + CHR$(191) +
     CHR$(189) + CHR$(135) + CHR$(132) + CHR$(128)
9015 HG$ = CHR$(160) + CHR$(176) + CHR$(186) +
     CHR$(144)
9016 HH$ = CHR$(128) + CHR$(143) + CHR$(191) +
     CHR$(191) + CHR$(142) + CHR$(132) + CHR$(128)
9017 H6$ = CHR$(128) + CHR$(143) + CHR$(191) +
     CHR$(191) + CHR$(143) + CHR$(132)
9020 H1$ = CHR$(170) + CHR$(176) + CHR$(181) +
     CHR$(144)
9021 HJ$ = CHR$(128) + CHR$(141) + CHR$(179) +
     CHR$(190) + CHR$(143) + CHR$(128)
9022 H3$ = CHR$(128) + CHR$(143) + CHR$(179) +
     CHR$(191) + CHR$(143)
9025 HK$ = CHR$(170) + CHR$(186) + CHR$(176) +
     CHR$(144)
9026 HL$ = CHR$(136) + CHR$(139) + CHR$(190) +
     CHR$(191) + CHR$(143) + CHR$(128)
9030 HM$ = CHR$(160) + CHR$(181) + CHR$(176) +
     CHR$(144)
9031 HN$ = CHR$(136) + CHR$(141) + CHR$(191) +
     CHR$(191) + CHR$(143) + CHR$(128)
9032 H7$ = CHR$(136) + CHR$(143) + CHR$(191) +
     CHR$(191) + CHR$(143)
9035 BA$ = CHR$(184) + CHR$(135) + CHR$(191) +
     CHR$(191) + CHR$(139) + CHR$(180)
9036 BB$ = CHR$(130) + CHR$(141) + CHR$(151) +
     CHR$(171) + CHR$(142) + CHR$(129)
9040 FC$ = CHR$(136) + CHR$(140) + CHR$(133) +
     CHR$(138) + CHR$(140) + CHR$(132)
9050 HO$ = CHR$(160) + CHR$(164) + CHR$(152) +
     CHR$(144)
9051 HP$ = CHR$(128) + CHR$(143) + CHR$(189) +
     CHR$(190) + CHR$(143) + CHR$(128)
9052 H4$ = CHR$(160) + CHR$(180) + CHR$(184) +
     CHR$(144)
9055 HQ$ = CHR$(160) + CHR$(180) + CHR$(184) +
     CHR$(144)
9056 HR$ = CHR$(128) + CHR$(175) + CHR$(155) +
     CHR$(167) + CHR$(159) + CHR$(128)
9057 H5$ = CHR$(128) + CHR$(175) + CHR$(159) +
     CHR$(175) + CHR$(159)
9060 BC$ = CHR$(139) + CHR$(180)
9061 BD$ = CHR$(142) + CHR$(129)
9065 BE$ = CHR$(191) + CHR$(128)
9066 BF$ = CHR$(143) + CHR$(128)
9070 BG$ = CHR$(191) + CHR$(128) + CHR$(128)
9071 BH$ = CHR$(130) + CHR$(141)
9075 B1$ = CHR$(191) + CHR$(176) + CHR$(176) +
     CHR$(128) + "  "
9076 BJ$ = CHR$(128) + CHR$(128)
9080 BK$ = CHR$(139) + CHR$(180) + CHR$(156) + ",-= "
9085 BL$ = STRING$(4, CHR$(131)) + "'-="
9090 BM$ = CHR$(128) + CHR$(191) + CHR$(191) +
     CHR$(191) + CHR$(171) + CHR$(148)
9091 BN$ = CHR$(142) + CHR$(129) + CHR$(151) +
     CHR$(171) + CHR$(142) + CHR$(129)
9095 BO$ = CHR$(171) + CHR$(148):
     BP$ = CHR$(130) + CHR$(141)
9100 FA$ = STRING$(6, CHR$(191)):
     FB$ = STRING$(6, CHR$(128))
9105 BR$ = CHR$(128) + CHR$(191) + CHR$(191) +
     CHR$(187) + CHR$(171) + CHR$(148)
9110 BQ$ = CHR$(184) + CHR$(135) + CHR$(191) +
     CHR$(187) + CHR$(139) + CHR$(180)
9200 A$ = CHR$(191):
     MA$ = A$ + A$ + "      " + CHR$(175) + A$
     + CHR$(148) + "  " + CHR$(190) + A$ + CHR$(189)
     + "  " + CHR$(168) + A$ + CHR$(159) + " "
     + A$ + A$ + " " + A$ + A$ + CHR$(175) + CHR$(189)
      + CHR$(144) + "  " + A$ + A$
9210 MB$ = A$ + A$ + "      " + CHR$(130) + A$
     + A$ + CHR$(144) + CHR$(186) + A$ + CHR$(131)
     + A$ + CHR$(181) + CHR$(160) + A$ + A$ +
     CHR$(129) + " " + A$ + A$ + " " + A$ + A$
     + " " + CHR$(139) + A$ + CHR$(180) + " "
     + A$ + A$
9220 MC$ = A$ + A$ + "       " + CHR$(138) + A$
     + CHR$(189) + A$ + CHR$(135) + " " + CHR$(139)
     + A$ + CHR$(190) + A$ + CHR$(133) + "  "
     + A$ + A$ + " " + A$ + A$ + "  " + CHR$(130)
     + CHR$(175) + CHR$(189) + A$ + A$
9300 RETURN
10000 CLS :
      GOSUB 7600
10010 K$ = INKEY$:
      IF K$ = ""
       THEN
        10010:
       ELSE
        10000
10020 PRINT "WHEN YOU SEE AN (*) TO THE LEFT OF 'ANDROI
      D NIM' ITS YOUR TURN."
10030 PRINT :
      PRINT "YOU NEED NOT PUSH ENTER, JUST THE ROW NUMB
      ER, 1 2 OR 3,"
10040 PRINT "AND THE NUMBER OF ANDRIODS YOU WISH TO HAV
      E REMOVED."
10060 PRINT "IF YOU WISH TO GIVE UP, PUSH THE 'R' KEY."
10080 PRINT :
      PRINT "RULES - - YOU MAY REMOVE AS MANY ANDROIDS
      FROM ANY"
```

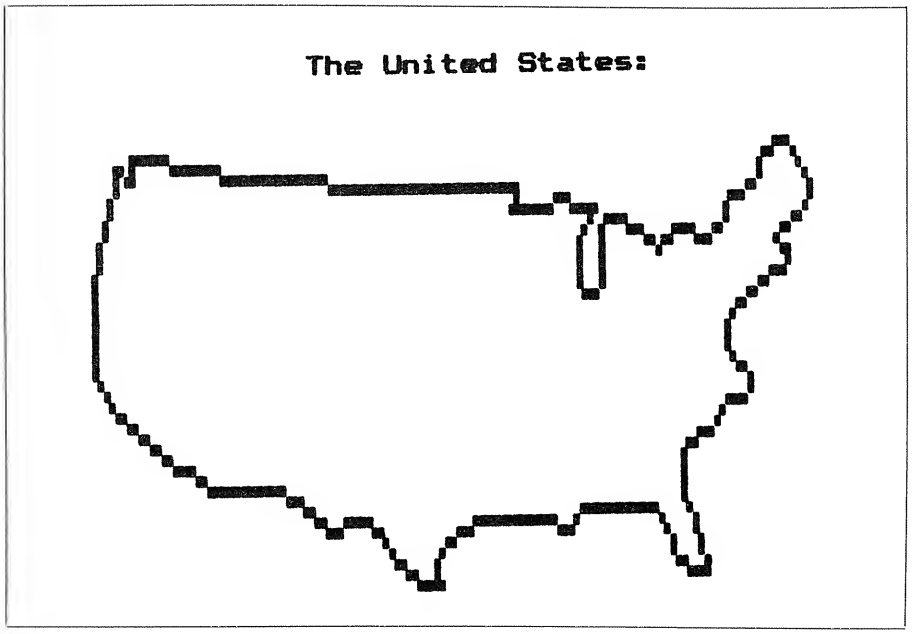
```
10090 PRINT "ROW AS YOU WISH WHEN IT IS YOUR TURN.  TO
      WIN YOU MUST"
10100 PRINT "REMOVE THE LAST ANDROID."
10110 PRINT :
      INPUT "PUSH ENTER WHEN READY & WAIT A BIT";X:
      GOTO 10
```



Designing TRS-80 Graphics

Graphics Drawingboard

William E. Bailey



William E. Bailey, 4601 N. Park Ave., Apt. 1811, Chevy Chase, MD 20815.

Designing TRS-80 computer graphics and converting them into ASCII graphics codes is a very boring and time-consuming job. In addition, it is next to impossible to complete the conversion process without making at least one mistake. Graphics Drawingboard greatly facilitates the design and conversion of graphic figures. In addition to saving hours of time, it also turns what would normally be considered a burden into a pleasure.

At the start of the program a cursor appears in the middle of the screen and the corners are lit to indicate their locations. The cursor can then be moved around the screen using the arrow keys. The six pixels in the character block are represented by the numbers 2, 3, 5, 6, 8, and 9 on the numeric keypad. Hitting one of these numbers lights up the corresponding pixel in the character block at the cursor location. Now the fun starts.

Using these keys, you can draw anything from maps to androids. To draw a given figure, trace the outline on a clear sheet of plastic, tape it to the TRS-80 screen, and use Graphics Drawingboard to transfer the figure to the screen. The outline of the continental United States (Figure 1) took about 15 minutes to complete—a task that would have taken hours otherwise.

Figure 1.

The United States:

LOCATION	VALUE	LOCATION	VALUE	LOCATION	VALUE	LOCATION	VALUE
60	160	61	176	66	156	67	186
68	131	69	131	70	131	71	140
72	140	73	140	74	140	75	164
76	176	77	176	78	176	79	176
80	176	81	176	82	176	83	176
84	176	123	150	124	129	126	137
127	144	129	168	130	129	149	131
150	131	151	131	152	131	153	131
154	131	155	131	156	131	157	131
158	131	159	131	160	131	161	131
162	131	163	131	164	131	165	187
166	176	167	176	168	176	169	140
170	164	171	176	172	176	184	152
185	140	186	131	191	154	192	160
193	133	235	152	236	129	237	150
238	131	239	137	240	140	241	176
242	160	243	152	244	140	245	164
246	176	247	140	248	129	252	160
253	140	254	131	256	154	299	149
301	149	306	129	316	176	317	155
320	149	363	137	364	140	365	129
377	176	378	140	379	131	384	149
440	150	448	149	504	137	505	176
512	130	513	164	568	176	569	176
570	133	578	131	579	140	580	176
629	160	630	176	631	134	645	131
646	140	647	176	648	176	692	150
693	129	713	131	714	140	715	140
716	140	717	140	718	140	719	140
720	140	721	176	722	144	756	149
786	130	787	137	788	164	789	176
790	140	791	140	792	164	793	144
800	176	801	152	802	140	803	140
804	140	805	140	806	140	807	140
808	140	809	176	810	152	811	131
812	131	813	131	814	131	815	131
816	131	817	131	818	164	820	130
821	148	857	130	858	164	859	144
862	152	863	131	883	165	884	144
885	138	886	144	923	130	924	137
925	140	926	141	948	130	949	131
950	129						

Figure 2.

Android:

LOCATION	VALUE	LOCATION	VALUE	LOCATION	VALUE	LOCATION	VALUE
480	176	481	191	482	191	483	191
484	176	544	190	545	189	546	143
547	190	548	189	608	184	609	174
610	191	611	157	612	180	671	130
672	173	673	186	674	191	675	181
676	158	677	129	737	170	739	149
800	136	801	142	803	141	804	132

The picture of an android (Figure 2) was also rapidly created in this manner.

To correct a mistake in any part of the design, just move the cursor to the block in which the mistake is located and hit the "C" key to clear that character block of its contents; then, use the numbers to change the contents of the block as needed.

When the design is finished, hit the "P" key to print out the location of the character blocks composing the picture (same as "PRINT AT" locations) and their corresponding values (ASCII graphics codes). The tedious graphics-ASCII conversion is completed without human error. □

Listing for Graphics Drawing Board.

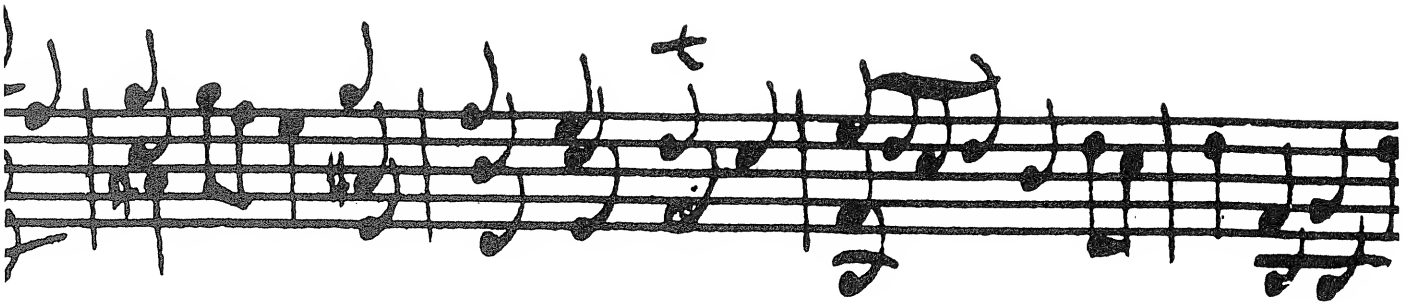
```
10 I=0:LN=128:L=480:DIM LO(1023):C=1
20 CLS
22 PRINT"     HIT 'P' TO PRINT SCREEN DATA
           HIT 'C' TO CLEAR LOCATION"
25 SET(0,0):SET(127,0):SET(127,47):SET(0,47)
30 POKE 15360+L,95
35 '-----------SCAN KEYBOARD-----------
40 IF PEEK(14340)=1 THEN 900
50 V=PEEK(14400):V=V/8
60 ON V GOTO 500,600
70 V=V/4
80 ON V GOTO 700,800
85 IF PEEK(14337)=8 THEN POKE15360+L,95:LO(L)=0
87 LN=PEEK(15360+L):IF LN<128 THEN LN=128
90 ON VAL(INKEY$)GOTO 100,400,450,100,300,350,100,200,250
100 GOTO 40
195 '-------------MAKE GRAPHIC PIXELS LIGHT UP--------------
200 IF LN+1>191 THEN40 ELSE LN=LN+1:POKE 15360+L,LN:GOTO 40
250 IF LN+2>191 THEN40 ELSE LN=LN+2:POKE 15360+L,LN:GOTO 40
300 IF LN+4>191 THEN40 ELSE LN=LN+4:POKE 15360+L,LN:GOTO 40
350 IF LN+8>191 THEN40 ELSE LN=LN+8:POKE 15360+L,LN:GOTO 40
400 IF LN+16>191 THEN40 ELSE LN=LN+16:POKE 15360+L,LN:GOTO 40
450 IF LN+32>191 THEN40 ELSE LN=LN+32:POKE 15360+L,LN:GOTO 40
495 '-----------MOVE CURSOR----------
500 IFL-64<10RL-64=63THEN85
505 IFPEEK(15360+L)>127THENLO(L)=LN
510 IFPEEK(15360+L)=95THENPOKE15360+L,32
520 L=L-64:IFPEEK(15360+L)<128THENPOKE15360+L,95:FORI=1TO20:NEXTI
530 GOTO85
600 IFL+64>1022ORL+64=960THEN85
605 IFPEEK(15360+L)>127THENLO(L)=LN
610 IFPEEK(15360+L)=95THENPOKE15360+L,32
620 L=L+64:IFPEEK(15360+L)<128THENPOKE15360+L,95:FORI=1TO20:NEXTI
630 GOTO85
700 IFL-1<10RL-1=63ORL-1=960THEN85
705 IFPEEK(15360+L)>127THENLO(L)=LN
710 IFPEEK(15360+L)=95THENPOKE15360+L,32
720 L=L-1:IFPEEK(15360+L)<128THENPOKE15360+L,95
730 GOTO85
800 IFL+1>1022ORL+1=63ORL+1=960THEN85
805 IFPEEK(15360+L)>127THENLO(L)=LN
810 IFPEEK(15360+L)=95THENPOKE15360+L,32
820 L=L+1:IFPEEK(15360+L)<128THENPOKE15360+L,95
830 GOTO85
895 '-----------DISPLAY GRAPHIC CODES FOR PICTURE-----------
900 IFPEEK(15360+L)<128THEN905ELSELO(L)=LN
905 CLS:FORI=1TO4:PRINT"LOCATION-VALUE",:NEXTI
910 FORI=0TO1023:IFLO(I)=0THENGOTO930
920 PRINTTAB(P+2)I;TAB(P+9)LO(I);
925 P=P+16
927 C=C+1:IFC=5THENPRINT:C=1:P=0
930 NEXTI
940 PRINT:PRINT:INPUT"DO YOU WANT TO OUTPUT
    THE DATA TO A PRINTER(Y/N)";A$
950 IFA$="N"THEN1030
960 '-----------OUTPUT TO PRINTER----------
965 C=1:P=0
970 FORI=1TO4:LPRINT"LOCATION-VALUE",:NEXTI
975 LPRINT
980 FORI=0TO1023:IFLO(I)=0THEN1020
990 LPRINTTAB(P+2)I;TAB(P+9)LO(I);
1000 P=P+16
1010 C=C+1:IFC=5THENLPRINT:C=1:P=0
1020 NEXTI
1025 LPRINT
1030 INPUT"DO YOU WANT TO ADD TO THE PICTURE";B$
1040 IF B$="N" THEN PRINT"GOOD-BYE":END
1050 '-----------DRAW PICTURE ON SCREEN-------------
1060 CLS
```

```
1065 PRINT" HIT 'P' TO PRINT SCREEN DATA  HIT 'C' TO CLEAR LOCATION"
1067 SET(0,0):SET(127,0):SET(127,47):SET(0,47)
1070 FORI=0TO1023:IFLO(I)=0THEN1090
1080 POKE15360+I,LO(I)
1090 NEXTI
1095 IF LO(L)<128THENPOKE15360+L,95
1097 C=1:P=0
1100 GOTO 40
```

Disk version.

```
10 I=0:LN=128:L=480:DIM LO(1023):C=1
20 CLS
30 INPUT "DO YOU WANT TO CALL A PREVIOUSLY SAVED FILE";B$
40 IF B$<>"N" THEN 940
50 CLS
60 PRINT"    HIT 'P' TO PRINT SCREEN DATA   HIT 'C' TO CLEAR LOCATION"
70 SET(0,0):SET(127,0):SET(127,47):SET(0,47)
80 POKE 15360+L,95
90 '-----------SCAN KEYBOARD-----------
100 IF PEEK(14340)=1 THEN 480
110 V=PEEK(14400):V=V/8
120 ON V GOTO 270  ,320
130 V=V/4
140 ON V GOTO 370  ,420
150 IF PEEK(14337)=8 THEN POKE15360+L,95:LO(L)=0
160 LN=PEEK(15360+L):IF LN<128 THEN LN=128
170 ON VAL(INKEY$)GOTO 180  ,240  ,250  ,180  ,220  ,230  ,180  ,200  ,210
180 GOTO 100
190 '------------MAKE GRAPHIC PIXELS LIGHT UP--------------
200 IF LN+1>191 THEN100  ELSE LN=LN+1:POKE 15360+L,LN:GOTO 100
210 IF LN+2>191 THEN100  ELSE LN=LN+2:POKE 15360+L,LN:GOTO 100
220 IF LN+4>191 THEN100  ELSE LN=LN+4:POKE 15360+L,LN:GOTO 100
230 IF LN+8>191 THEN100  ELSE LN=LN+8:POKE 15360+L,LN:GOTO 100
240 IF LN+16>191 THEN100  ELSE LN=LN+16:POKE 15360+L,LN:GOTO 100
250 IF LN+32>191 THEN100  ELSE LN=LN+32:POKE 15360+L,LN:GOTO 100
260 '-----------MOVE CURSOR----------
270 IFL-64<1ORL-64=63THEN150
280 IFPEEK(15360+L)>127THENLO(L)=LN
290 IFPEEK(15360+L)=95THENPOKE15360+L,32
300 L=L-64:IFPEEK(15360+L)<128THENPOKE15360+L,95:FORI=1TO20:NEXTI
310 GOTO150
320 IFL+64>1022ORL+64=960THEN150
330 IFPEEK(15360+L)>127THENLO(L)=LN
340 IFPEEK(15360+L)=95THENPOKE15360+L,32
350 L=L+64:IFPEEK(15360+L)<128THENPOKE15360+L,95:FORI=1TO20:NEXTI
360 GOTO150
370 IFL-1<1ORL-1=63ORL-1=960THEN150
380 IFPEEK(15360+L)>127THENLO(L)=LN
390 IFPEEK(15360+L)=95THENPOKE15360+L,32
400 L=L-1:IFPEEK(15360+L)<128THENPOKE15360+L,95
410 GOTO150
420 IFL+1>1022ORL+1=63ORL+1=960THEN150
430 IFPEEK(15360+L)>127THENLO(L)=LN
440 IFPEEK(15360+L)=95THENPOKE15360+L,32
450 L=L+1:IFPEEK(15360+L)<128THENPOKE15360+L,95
460 GOTO150
470 '-----------DISPLAY GRAPHIC CODES FOR PICTURE-----------
480 IFPEEK(15360+L)<128THEN490  ELSELO(L)=LN
490 CLS:FORI=1TO4:PRINT"LOCATION-VALUE",:NEXTI
500 FORI=0TO1023:IFLO(I)=0THENGOTO540
510 PRINTTAB(P+2)I;TAB(P+9)LO(I);
520 P=P+16
530 C=C+1:IFC=5THENPRINT:C=1:P=0
540 NEXTI
550 PRINT:PRINT:INPUT"DO YOU WANT TO OUTPUT THE DATA TO A PRINTER(Y/N)";A$
560 IFA$="N"THEN670
570 '-----------OUTPUT TO PRINTER----------
580 C=1:P=0
590 FORI=1TO4:LPRINT"LOCATION-VALUE",:NEXTI
600 LPRINT
610 FORI=0TO1023:IFLO(I)=0THEN650
620 LPRINTTAB(P+2)I;TAB(P+9)LO(I);
```

```
630 P=P+16
640 C=C+1:IFC=5THENLPRINT:C=1:P=0
650 NEXTI
660 LPRINT
670 INPUT"DO YOU WANT TO ADD TO THE PICTURE";B$
680 IF B$="N" THEN 790
690 '----------DRAW PICTURE ON SCREEN--------------
700 CLS
710 PRINT"     HIT 'P' TO PRINT SCREEN DATA  HIT 'C' TO CLEAR LOCATION"
720 SET(0,0):SET(127,0):SET(127,47):SET(0,47)
730 FORI=0TO1023:IFLO(I)=0THEN750
740 POKE15360+I,LO(I)
750 NEXTI
760 IF LO(L)<128THENPOKE15360+L,95
770 C=1:P=0
780 GOTO 100
790 '----------SAVE PICTURE TO DISK---------------
800 PRINT"DO YOU WANT TO SAVE THE PICTURE TO DISK";:INPUT B$
810 IF B$="N" THEN GOTO 890
820 INPUT"INPUT FILESPEC";F$
830 F$=F$+":1"
840 OPEN "O",1,F$
850 FOR I=0 TO 1023
860 PRINT#1,LO(I);
870 NEXT I
880 CLOSE 1
890 PRINT"DO YOU WANT TO SEE THE PICTURE";:INPUT B$
900 IF B$="Y" THEN 690
910 INPUT"DO YOU WANT TO DUMP PICTURE TO THE PRINTER";B$
920 IF B$="Y" THEN GOTO 30010 ELSE END
930 GOTO 690
940 '---------------RECALL FILE----------------
950 INPUT "FILESPEC";F$
960 F$=F$+":1"
970 OPEN "I",1,F$
980 FOR I=0 TO 1023
990 INPUT #1,LO(I)
1000 NEXT I
1010 CLOSE 1
1020 GOTO 690
30000 REM-----SCREEN DUMP TO PRINTER WITH GRAFTRAX--------
30010 FOR B1=0 TO 1023 STEP 64
30020 LPRINT TAB(25);
30030 FOR B2=0 TO 32 STEP 32
30040 LPRINT CHR$(27)"L"CHR$(192)CHR$(0);
30050 FOR B3=0 TO 31
30060 B4=0:B5=0
30070 B6=PEEK(15360+B2+B3+B1)
30080 IF (B6 AND 1)=1 THEN B4=B4+240
30090 IF (B6 AND 4)=4 THEN B4=B4+15
30100 IF (B6 AND 2)=2 THEN B5=B5+240
30110 IF (B6 AND 8)=8 THEN B5=B5+15
30120 FOR B7=1 TO 3:OUT 251,B4:NEXT B7
30130 FOR B7=1 TO 3:OUT 251,B5:NEXT B7
30140 NEXT B3
30150 GOSUB 30360
30160 NEXT B2
30170 LPRINT CHR$(27)"A"CHR$(8)
30180 GOSUB 30360
30190 LPRINT TAB(25);
30200 FOR B2=0 TO 32 STEP 32
30210 LPRINT CHR$(27)"L"CHR$(192)CHR$(0);
30220 FOR B3=0 TO 31
30230 B4=0:B5=0
30240 B6=PEEK(15360+B2+B3+B1)
30250 IF (B6 AND 16)=16 THEN B4=240
30260 IF (B6 AND 128)=128 AND (B6 AND 32)=32 THEN B5=240
30270 FOR B7=1 TO 3:OUT 251,B4:NEXT B7
30280 FOR B7=1 TO 3:OUT 251,B5:NEXT B7
30290 NEXT B3
30300 GOSUB 30360
30310 NEXT B2
30320 LPRINT CHR$(27)"A"CHR$(4)
30330 GOSUB 30360
30340 NEXT B1
30350 LPRINT CHR$(27)"@":RETURN
30360 FOR B7=1 TO 500:NEXT B7    'ADDITIONAL WAIT TO BE SURE
30370 RETURN
```



Celestial Music

Leo Christopherson

In my most recent programs, Duel-N-Droids and Voyage of the Valkyrie, I have used a musical sound effect which has attracted a great deal of attention. I call it a "celeste" sound. This article will show you how to use the musical sound effect in your own programs on the TRS-80, Models I or III, on the Apple II Plus, and on the Atari.

Before learning the details, you might be interested in hearing how this technique came about.

In the Beginning...

In the beginning, there was the TRS-80 Model I (at least that's where I started). This machine seemed to be unable to output sound effects. But then, along came the tape output, machine level routines, and string packing, which combined to provide a way to create sound effects. These ideas led to "one-note-at-a-time" music. I'll call this first musical routine, "Number One." But, I found myself wishing to be able to output two (or more) music notes simultaneously. In other words, I wanted at least two-part harmony.

The first musical routine begat a second. Number Two was really just the first one used twice. The routine sent out one note followed immediately by another, and then repeated this several times to give the notes duration. Depending on how the timing loops were set, the sound varied from a warbling, up and down kind of thing, to the effect of a base (fundamental) note with an overtone.

Leo Christopherson, 179 E. 129th, Tacoma, WA 98445.

```
 5 CLS: PRINT "CELESTE MUSIC DEMO"
10 S$="............(70 PERIODS IN THIS LINE)..............."
11 S1=PEEK(VARPTR(S$)+1): S2=PEEK(VARPTR(S$)+2): S0=S1+S2*256
12 POKE 16422,S1: POKE 16423,S2
20 A$="............(52 PERIODS IN THIS LINE)..............."
21 A1=PEEK(VARPTR(A$)+1): A2=PEEK(VARPTR(A$)+2): A0=A1+A2*256
25 GOSUB 60
30 POKE S0+1,A1: POKE S0+2,A2: LPRINT: STOP
50 DATA 33,1,1,243,62,1,8,62,35,61,190,32,2,251
51 DATA 201,126,35,86,94,29,14,10,6,225,21,32,14,87
52 DATA 62,120,190,40,6,8,238,3,211,255,8,122,86,29
53 DATA 32,15,95,62,120,190,40,6,8,238,3,211,255,8
54 DATA 123,94,29,16,219,13,32,214,61,32,209,35,24,193
55 DATA 32,149,32,74,16,79,8,99,8,88,16,79,12,74,4,120
56 DATA 32,149,32,88,48,99,16,120,32,177,32,111,16,118
57 DATA 8,149,8,133,16,118,12,111,4,120,16,133,8,158
58 DATA 8,149,16,133,16,118,64,149
60 RESTORE: FOR N=0TO69: READ D: POKE S0+N,D: NEXT N
65 FOR N=0TO51: READ D: POKE A0+N,D: NEXT N: RETURN
```

Celeste Program for the TRS-80, Models I and III.

```
10 HOME: PRINT "CELESTE MUSIC DEMO"
15 DATA 1,.....(81 PERIODS IN THIS LINE).....
20 DATA 2,.....(53 PERIODS IN THIS LINE).....
25 RESTORE: READ D$: S1=PEEK(125): S2=PEEK(126): S0=S1+S2*256+1
30 READ D$: READ D$: A1=PEEK(125): A2=PEEK(126): A0=A1+A2*256+1
35 READ D$: GOSUB 70
40 POKE 250,A1: POKE 251,A2: CALL S0: STOP
50 DATA 169,1,133,249,164,249,177,250,201,255,208
51 DATA 1,96,133,252,200,177,250,133,253,166,253
52 DATA 164,253,136,169,32,133,254,169,255,133,255,202
53 DATA 208,11,165,253,201,100,240,3,173,48,192,166,253
54 DATA 136,208,12,165,253,201,100,240,3,173,48,192,164
55 DATA 253,136,198,255,208,223,198,254,208,215,198,252
56 DATA 208,207,230,249,230,249,24,144,179
57 DATA 16,215,16,107,8,113,4,143,4,127,8,113,6,107,2
58 DATA 100,16,215,16,127,24,143,8,100,16,254,16,161,8,171
59 DATA 4,215,4,192,8,171,6,161,2,100,8,192,4,229
60 DATA 4,215,8,192,8,171,32,215,255
70 FOR N=0TO80: READ D: POKE S0+N,D: NEXT N
75 FOR N=0TO52: READ D: POKE A0+N,D: NEXT N: RETURN
```

Celeste Program for the Apple, using Applesoft Basic.

Number Two begat Number Three. It used three Number Ones, end to end. The useful result was to give the effect of a base note with two overtones. These three music routines, Numbers One, Two, and Three have been used in most of my programs.

But, I still didn't have the two-part harmony I wanted.

And so it came to pass that Number Three begat Number Four. And Number Four begat Number five...and Number Twelve begat Number Thirteen. After this many generations, many mutations had crept in, however. Most of these monsters were laid to rest immediately after birth.

Ah, but behold Lucky Number Thirteen. It did not produce a base note with twelve overtones. I had finally created a routine which would output two notes simultaneously.

Number Thirteen ties the two pitch delay loops together with a couple of duration delay loops. It works. But, as is true with many offspring, my Number Thirteen has one major flaw in its character. In the case of Thirteen, under most circumstances, it sounds absolutely awful!

Square waves do not add together well. I found that combining two pitches, such as a "C" and an "E," would produce a myriad of overtones which clashed. But, the most serious problem resulted from beats between the fundamentals of the two notes. The beat frequency was often a note lower in pitch than either of the two original notes. It was also usually not in harmony with them. It was also very loud.

There was only one combination of pitches that actually did sound very good. That occurred when the frequency of the one note was almost the same as that of the other. This is the "celeste" sound.

I think the name, celeste, comes from the pipe organ people. Organs often have a stop consisting of two ranks of pipes that are slightly out of tune with each other. It's called a celeste stop (from the word "celestial"), because it gives a very sweet and heavenly sound as the two frequencies slowly beat with each other. The mandolin produces this sort of sweet sound by using two strings at each pitch.

But, now let's get down to earth about this whole celestial business. As you're exorcizing those devilish little bugs from your next program, you may want to add a little heavenly music to soothe the savage beast.

The Word For TRS-80 and Apple Users

To get the celeste music routine working, all you have to do is type in and RUN the Basic program for your machine. Of course, every number of each DATA line has to be correct! We're dealing with machine level here, and once your program has jumped into the routine, the usual Basic error traps probably will not work. You may lose the whole program. I strongly suggest that the program be SAVEd before it is RUN.

The TRS-80 program uses string packing, while the Apple program uses DATA packing. These techniques allow the machine level routines to be saved as part of the Basic program. There doesn't need to be a separate machine level load. This

TRS-80 Celeste Music Scales.

Octave One			Ocatve Two			Octave Three		
Note	Decimal	Hex	Note	Decimal	Hex	Note	Decimal	Hex
Eb	251	FB	Eb	125	7D	Eb	62	3E
E	238	EE	E	118	76	E	59	3B
F	225	E1	F	111	6F	F	55	37
Gb	211	D3	Gb	105	69	Gb	52	34
G	199	C7	G	99	63	G	49	31
Ab	188	BC	Ab	93	5D	.....		
A	177	B1	A	88	58			
Bb	168	A8	Bb	83	53	Rest = 120		78
B	158	9E	B	79	4F			
C	149	95	C	74	4A			
Db	141	8D	Db	70	46			
D	133	85	D	66	42			
.....			.....					

can be important for tape users, for whom loading a separate binary program can be a pain.

The TRS-80 program uses VARPTR to find the absolute memory addresses of the music routine and the musical data.

The Apple program uses DATA, READ, and RESTORE to find the absolute memory addresses for the machine level material. It makes use of zero page addresses 125 and 126. These addresses contain the low and high bytes of the absolute address of the next DATA which will be read.

TRS-80 people will need to connect an amplifier to the tape AUX out plug.

Once the program is running correctly, TRS-80 programmers may DELETE line 25 and lines 50-65 and then SAVE the program again. After the first time through, lines 10 and 20 have been packed and there is no need to pack them again.

Apple programmers may DEL line 35, and then DEL lines 50 and 75. Do this after the program has been tested and found to be OK, then SAVE it again. Once DATA lines 15 and 20 are packed, there is no need to pack them again.

The Word For Atari Users

You Atari people are indeed the fortunate folk in this case. Since you can output up to four simultaneous musical notes directly

Celeste Program for the Atari.

```
10  PRINT "<ESC><CTRL><CLEAR>
    CELESTE MUSIC DEMO"
50  DATA 16,121,16,60,8,64,4,81,4,72
51  DATA 8,64,6,60,2,100,16,121,16,72
52  DATA 24,81,8,100,16,144,16,91,8,96
53  DATA 4,121,4,108,8,96,6,91,2,100
54  DATA 8,108,4,128,4,121,8,108,8,96
55  DATA 32,121,255
60  RESTORE
65  DISTORTION = 10: READ DURATION
70  IF DURATION = 255 THEN 110
75  READ PITCH
80  IF PITCH = 100 THEN DISTORTION = 1
85  SOUND 0, PITCH, DISTORTION, 10
90  SOUND 1, PITCH-1, DISTORTION, 5
95  FOR DELAY = 0 TO DURATION * 40
100 NEXT DELAY
105 GOTO 65
110 SOUND 0, 0, 0, 0: SOUND 1, 0, 0, 0
115 STOP
```

TRS-80 Celeste Music Data, (Decimal Values).

Note#	Duration	Pitch		Note#	Duration	Pitch	
1	32	149	C	14	32	111	F
2	32	74	C	15	16	118	E
3	16	79	B	16	8	149	C
4	8	99	G	17	8	133	D
5	8	88	A	18	16	118	E
6	16	79	B	19	12	111	F
7	12	74	C	20	4	120	Rest
8	4	120	Rest	21	16	133	D
9	32	149	C	22	8	158	B
10	32	88	A	23	8	149	C
11	48	99	G	24	16	133	D
12	16	120	Rest	25	16	118	E
13	32	177	A	26	64	149	C

Note#	Duration	Pitch		Note#	Duration	Pitch	
1	16	215	C	14	16	161	F
2	16	107	C	15	8	171	E
3	8	113	B	16	4	215	C
4	4	143	G	17	4	192	D
5	4	127	A	18	8	171	E
6	8	113	B	19	6	161	F
7	6	107	C	20	2	100	Rest
8	2	100	Rest	21	8	192	D
9	16	215	C	22	4	229	B
10	16	127	A	23	4	215	C
11	24	143	G	24	8	192	D
12	8	100	Rest	25	8	171	E
13	16	254	A	26	32	215	C
				27	255	(End Byte)	

Apple Celeste Music Data, (Decimal Values).

Note#	Duration	Pitch		Note#	Duration	Pitch	
1	16	121	C	14	16	91	F
2	16	60	C	15	8	96	E
3	8	64	B	16	4	121	C
4	4	81	G	17	4	108	D
5	4	72	A	18	8	96	E
6	8	64	B	19	6	91	F
7	6	60	C	20	2	100	Rest
8	2	100	Rest	21	8	108	D
9	16	121	C	22	4	128	B
10	16	72	A	23	4	121	C
11	24	81	G	24	8	108	D
12	8	100	Rest	25	8	96	E
13	16	144	A	26	32	121	C
				27	255	(End Byte)	

Atari Celeste Music Data, (Decimal Values).

from Basic, the celeste effect, using two notes, is quite easy to achieve. Just enter and RUN the Basic program for your machine to hear the celeste sounds.

Dancing To A Different Tune

It is likely that you will now want to try out a tune of your own choosing. The following instructions will help you create your own celestial music.

The music data for all three machines consists of a series of two-byte groups. The first byte of the pair is the duration of the note, and the second byte is the pitch of the note. Apple and TRS-80 programmers may use only about 120 of these note pairs since the notes are packed into Basic lines which have a maximum length of 255 bytes. The Atari programmer is limited only by the size of the memory in his machine.

To change the music data, you must first refer to the table of Celeste Music Scales for your machine. Using this table, you must write out pairs of bytes for each note of your music: the first byte is the duration and the second is the pitch from the table. Atari owners should use the table of pitches provided in the Basic Reference Manual which came with the machine. Programmers may also wish to refer to the table showing the Celeste Music Data from the original program.

All three versions use an "end byte" to tell the routine that the music is done. On the TRS-80, the quotation mark at the end of A$ is used. For the Apple and the Atari, a single byte of 255 must be placed at the end of your data pairs.

TRS-80

The data you have prepared must now be packed into line 20. Count how many separate items of data you have and then enter a new line 20 with A$ equal to a series of periods equal in number to the data count. Enclose the periods in quotation marks.

If you haven't DELETEd lines 50-65 yet, do so now. Have line 25 as shown in the original program. Put your new data into DATA lines from 50 to 59. Then add a line 60 as follows:

60 FOR N=0TOX: READ D: POKE A0+N,D: NEXT N: RETURN

The "X" is to be replaced by a number which is one less than your data count. When you RUN the program, you should hear your new musical data played. If all is well, you may DELETE line 25 and lines 50-60.

Apple

We must now pack line 20 with your new music data. Count the number of items of data you have, including the "end byte." Retype line 20 as follows:

Octave One			Ocatve Two			Octave Three		
Note	Decimal	Hex	Note	Decimal	Hex	Note	Decimal	Hex
A	254	FE	A	127	7F	A	63	3F
Bb	241	F1	Bb	120	8	Bb	60	3C
B	229	E5	B	113	71	B	56	38
C	215	D7	C	107	6B	C	53	35
Db	203	CB	Db	101	65	Db	50	32
D	192	C0	D	95	5F	D	47	2F
Eb	181	B5	Eb	90	5A	Eb	44	2C
E	171	AB	E	85	55	E	42	2A
F	161	A1	F	80	50	.....		
Gb	151	97	Gb	75	4B			
G	143	8F	G	71	47	Rest = 100		64
Ab	135	87	Ab	67	43			
.....			.....			Data End = 255		FF

Apple Celeste Music Scales.

20 DATA 2,....(number of periods is same as data count)....

If you haven't deleted lines 50 through 75, do so now. Have line 35 as shown in the original program. Put your new data into DATA lines from 50 to 69. Then type a line 70 to read:

70 FOR N=0TOX: READ D: POKE A0+N,D: NEXT N: RETURN

The "X" is to be replaced by the number you get by subtracting one from your data count. When you RUN the program, you should hear your new musical selection. You may now wish to delete line 35 and the lines from 50 through 70.

Atari

The data lines 50 to 55 are the music data. Just replace these lines with your new data and run the program to hear the new music you've made. A table of the original Celeste Music Data is included as an example.

...The Last Words

I have included the machine code listings for the Z-80 and 6502 celeste music subroutines. You more ambitious programmers may want to go through them and modify and improve upon them to suit your own purposes. You may find an especially valuable pot of gold somewhere over your own rainbow! □

Step#	Decimal	Hex	Statement	Step#	Decimal	Hex	Statement
0	33	21	LDHL,NN	35	3	03	N
1	1	01	N	36	211	D3	OUT(N),A
2	1	01	N	37	255	FF	N
3	243	F3	DI	38	8	08	EXAF,AF'
4	62	3E	LDA,N	39	122	7A	LDA,D
5	1	01	N	40	86	56	LDD,(HL)
6	8	08	EXAF,AF'	41	29	1D	DEC,D
7	62	3E	LDA,N	42	32	20	JRNZ
8	35	23	N	43	15	0F	e
9	61	3D	DEC,A	44	95	5F	LDE,A
10	190	BE	CP,(HL)	45	62	3E	LDA,N
11	32	20	JRNZ	46	120	78	N
12	2	02	e	47	190	BE	CP,(HL)
13	251	FB	EI	48	40	28	JRZ
14	201	C9	RET	49	6	06	e
15	126	7E	LDA,(HL)	50	8	08	EXAF,AF'
16	35	23	INC,HL	51	238	EE	XOR,N
17	86	56	LDD,(HL)	52	3	03	N
18	94	5E	LDE,(HL)	53	211	D3	OUT(N),A
19	29	1D	DEC,E	54	255	FF	N
20	14	0E	LDC,N	55	8	08	EXAF,AF'
21	10	0A	N	56	123	7B	LDA,E
22	6	06	LDB,N	57	94	5E	LDE,(HL)
23	255	FF	N	58	29	1D	DEC,E
24	21	15	DEC,D	59	16	10	DJNZ
25	32	20	JRNZ	60	219	DB	e
26	14	0E	e	61	13	0D	DEC,C
27	87	57	LDD,A	62	32	20	JRNZ
28	62	3E	LDA,N	63	214	D6	e
29	120	78	N	64	61	3D	DEC,A
30	190	BE	CP,(HL)	65	32	20	JRNZ
31	40	28	JRZ	66	209	D1	e
32	6	06	e	67	35	23	INC,HL
33	8	08	EXAF,AF'	68	24	18	JR
34	238	EE	XOR,N	69	193	C1	e

TRS-80/Z-80 Celeste Music Subroutine

Step#	Decimal	Hex	Statement	Step#	Decimal	Hex	Statement
Ø	169	A9	LDA	41	3	Ø3	e
1	1	Ø1	Ø1	42	173	AD	LDA
2	133	85	STA	43	48	3Ø	3Ø
3	249	F9	F9	44	192	CØ	CØ
4	164	A4	LDY	45	166	A6	LDX
5	249	F9	F9	46	253	FD	FD
6	177	B1	LDA,Y	47	136	88	DEY
7	25Ø	FA	FA	48	2Ø8	DØ	BNE
8	2Ø1	C9	CMP	49	12	ØC	e
9	255	FF	FF	5Ø	165	A5	LDA
1Ø	2Ø8	DØ	BNE	51	253	FD	FD
11	1	Ø1	e	52	2Ø1	CØ	CMP
12	96	6Ø	RTS	53	1ØØ	64	64
13	133	85	STA	54	24Ø	FØ	BEQ
14	252	FC	FC	55	3	Ø3	e
15	2ØØ	C8	INY	56	173	AD	LDA
16	177	B1	LDA,Y	57	48	3Ø	3Ø
17	25Ø	FA	FA	58	192	CØ	CØ
18	133	85	STA	59	164	A4	LDY
19	253	FD	FD	6Ø	253	FD	FD
2Ø	166	A6	LDX	61	136	88	DEY
21	253	FD	FD	62	198	C6	DEC
22	164	A4	LDY	63	255	FF	FF
23	253	FD	FD	64	2Ø8	DØ	BNE
24	136	88	DEY	65	223	DF	e
25	169	A9	LDA	66	198	C6	DEC
26	32	2Ø	2Ø	67	254	FE	FE
27	133	85	STA	68	2Ø8	DØ	BNE
28	254	FE	FE	69	215	D7	e
29	169	A9	LDA	7Ø	198	C6	DEC
3Ø	255	FF	FF	71	252	FC	FC
31	133	85	STA	72	2Ø8	DØ	BNE
32	255	FF	FF	73	2Ø7	CF	e
33	2Ø2	CA	DEX	74	23Ø	E6	INC
34	2Ø8	DØ	BNE	75	249	F9	F9
35	11	ØB	e	76	23Ø	E6	INC
36	165	A5	LDA	77	249	F9	F9
37	253	FD	FD	78	24	18	CLC
38	2Ø1	C9	CMP	79	144	9Ø	BCC
39	1ØØ	64	64	8Ø	179	B3	e
4Ø	24Ø	FØ	BEQ				

Apple/6502 Celeste Music Subroutine.

Create a Piano-Like Keyboard

Get Set to Tune Up Your TRS-80

Howard Silver

Have you ever wondered how some TRS-80 game programs create sound effects, or how people teach their computers to play music?

Howard Silver, 10 Clairmont Drive, Woodcliff Lake, NJ 07675.

Generating sounds with a computer has been the subject of recent articles in various computer journals. Most are written for the serious hobbyist with a knowledge of both hardware and software. My objective is to offer a tutorial introduction to this subject and give you some simple routines which you can run on your TRS-80, without any hardware modifications.

Your kids can have fun as well as learn from these routines, and it gives you something simple with which to impress your friends. When you get beyond this article you will undoubtedly want to consult

the literature and learn how to use such hardware as digital-to-analog converters and synthesizer boards, to allow you to generate higher quality audio.

I will attempt in this article to show how simple digital tones can be generated by software and how this technique can be extended to keyboard controlled sound generation. In this way you can convert your TRS-80 keyboard into a piano-like keyboard or into a sound effects generator.

The only hardware you need is an amplifier/speaker. Radio Shack's Archer mini-amplifier with adjustable volume control goes for about $12, and will do just fine for this purpose. All you have to do is connect the AUX jack from your tape recorder to the input of the amplifier and you're wired for sound. Don't forget, however, to reconnect the AUX jack when you want to CSAVE programs.

The easiest way to generate sound is to program your computer to output a square wave (i.e., alternating 0s and 1s) at a frequency in the audio range approximately 100-10,000 Hz). One thing I discovered quickly is that if you try to do this in Basic, the interpreter inserts far too much delay, and all you hear is low frequency "static."

In machine language, we have the opposite problem: since instructions are executed in a few microseconds, we get frequencies way above the audio range; however, we can slow things down to the desired speed by inserting some time delay into the program.

If your machine language routines are fairly short, as they are in these examples, they can be easily POKEd into memory and executed via the USR command with Level II Basic. For longer machine language programs, system programs such as EDTASM and T-Bug are handy.

To illustrate, here is a simple routine to output a square wave to the amplifier/speaker. It is written in Z-80 assembly language. (Such routines can be converted to machine code using EDTASM or by hand assembling with a Z-80 coding chart.) An advantage to assembly language or machine language programming is that you know exactly how long it takes to execute an instruction—that's important here. The cycle times are shown with the instructions in Listing 1.

			Instruction Cycle Times
NEXT	OUT	(0FFH), A	11
	LD	B, 150	7
LOOP	DJNZ	LOOP	8/13*
	INC	A	4
	JR	NEXT	12

*8 cycles if the condition is not true, 13 if true.

Program Listing 1. Assembly Language Routine to Output Square Wave at Audio Frequency.

The first instruction outputs the Z-80's accumulator (A) contents to Port 255 (hex FF); the TRS-80 decodes this port address and sends the two least significant bits of A to a cassette audio ouput circuit; a signal generated by this circuit goes to the AUX jack, which is now connected to the amplifier/speaker. The TRS-80 Reference handbook (page 46) discusses the audio circuitry if you are interested.

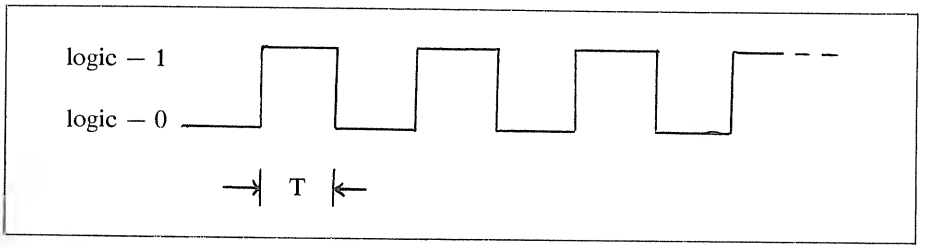


Figure 1. Square wave sent to audio output circuit.

```
5 'LOAD TONE GENERATION ROUTINE
10 POKE 16526,0 :POKE 16527,125
20 FOR X=32000 TO 32010:READ A:POKE X,A:NEXT
30 DATA 205,127,10,211,255,69,16,254,60,24,248
35 'ENTER FREQUENCY FACTOR AND CALL MACHINE LANGUAGE ROUTINE
40 INPUT N:X=USR(N)
```

Program Listing 2. Basic routine to load and execute tone generation program.

Accumulator A has a random number initially, but that doesn't matter since we are alternating 0s and 1s anyway. The OUT instruction, incidentally, is equivalent to the Level II Basic instruction OUT 255,X where X can be any 8-bit number.

Register B is used as a delay counter, and the LD instruction loads it with decimal 150 initially. This number was chosen arbitrarily for illustration. The DJNZ instruction causes B to be decremented by one, and then jumps to the specified location LOOP (which is the same place) if the result is not zero. Therefore, the program will continue to execute this instruction (i.e., remain in the LOOP) until the count reaches zero. For our illustration DJNZ is executed 150 times.

After exiting the loop, INC A results in incrementing A by one; the least significant bit is always complemented by this operation and the next bit is complemented every second time. This gives us the pattern of alternating 0s and 1s.

The last instruction, JR, returns us to the first instruction and causes the new bit value to be outputted. We, therefore, have generated a square wave as shown in Figure 1.

The delay counter (register B) establishes the time between successive outputs, and thus the frequency of the square wave.

From what I can gather from the description of the audio output circuit in the Reference Handbook, an approximation of a sine wave is generated at a frequency corresponding to the square wave frequency of the *next to least* significant bit. This frequency is then

$$f = 1/2\left(\frac{1}{2T}\right)$$

where T is the time between successive OUT instructions in the assembly language program. Since the DJNZ instruction is repeated 150 times (the initial value placed in the B register counter), the total number of cycles works out to

$$\underbrace{11}_{\text{OUT}} + \underbrace{7}_{\text{LD}} + \underbrace{13(150-1) + 8}_{\text{DNJZ}} + \underbrace{4}_{\text{INC}} + \underbrace{12}_{\text{JR}} = 1979$$

The TRS-80 clock runs at a rate of 1.774 MHz, so that T is then

$$\frac{1979}{1.774} = 1116 \text{ microseconds}$$

and the frequency of our tone should be

$$\frac{10^6}{4(1116)} = 224 \text{ Hz.}$$

For those musically inclined, this corresponds to the note A below middle C on

```
                ORG   7D00H
7D00 CD7F0A     CALL  0A7FH     ;PASS BASIC ROUTINE PARAMETER TO HL
7D03 D3FF   NEXT OUT  (0FFH),A ;OUTPUT ACCUMULATOR TO CASSETE RECORDER PORT
7D05 45         LD    B,L
7D06 10FE   LOOP DJNZ LOOP     ;DECREMENT COUNT UNTIL ZERO REACHED
7D07 3C         INC   A         ;BIT VALUE SENT TO CASSETE'S JACK
7D08 18F8       JR    NEXT      ;IS CHANGED EVERY SECOND TIME
```

Program Listing 3. Assembly language and machine code for revised tone generation routine.

the scale.

The Basic routine shown in Listing 2 will allow you to POKE the machine code for the tone generation program discussed. A modification was made to allow us to select the tone frequency by inputting an 8-bit number, which is then passed along to the machine language.

Lines 10 and 20 insert the coded machine language instructions (data on line 30). Line 40 allows us to input a variable N (N should be an integer from 1 to 255), and X = USR(N) calls the machine language routine. The corresponding machine code is shown in Listing 3.

Listing 3 has added one line to Listing 1 and changed another. The instruction CALL 0A7FH branches to a subroutine stored in the Level II Basic interpreter. This subroutine loads the value of N specified by the USR command into the 16-bit register paid HL. Since N is only an 8-bit number here, it appears in the 8-bit L register. The instruction LD B,L then moves N from L to B in preparation for the decrement and jump operation (DJNZ). LD B,L takes four clock cycles.

The total cycles are calculated as previously:

$$11 + 4 + 13(N-1) + 8 + 4 + 12 = 13N + 26 = 13(N+2)$$

and the tone frequency is

$$\frac{1.744 \times 10^6}{4 \times 13(N+2)} = \frac{34100}{N+2} \quad \text{Hz}$$

where N ranges from 1 to 255. Table 1 illustrates the fact that a variable 8-bit delay factor N allows us to produce tones which span the audio range.

N	f	
225	133 Hz	(lower C on music scale)
200	169	
100	335	
50	656	
20	1550	
10	2840	
5	4870	
2	8530	
1	11370	

Table 1. Audio Frequencies generated from values of Delay Factor N.

Incidentally, since the machine language routine loops endlessly, you can exit it only by pressing the Reset button (the Break key only gets you out of a Basic routine).

The revised tone generating routine allows us to input a number which, in effect, selects the tone frequency or pitch. However, we must stop and restart the program every time we want to change the pitch.

How about controlling the pitch from the keyboard while the program is running? That's easy to do, and it gives us something very useful—a "piano-like" TRS-80 keyboard. A Basic routine which produces keyboard controlled tones is shown in Listing 4; the assembly instructions and machine code are given in Listing 5.

Listings 4 and 5 illustrate the usefulness of linking the Basic and machine language routines. That part of the program which waits for a key to be pressed can be written in Basic since time is not critical. Once a key is pressed, the machine language code issues the tone and also checks the key to insure that it has remained down. When the key is released, control returns to our

```
10 DEFINT A-Z
15 'LOAD  KEYBOARD CONTROLLED TONE ROUTINE
20 POKE 16526,0 :POKE 16527,125
30 FOR X=32000 TO 32023 :READ A :POKE X,A :NEXT
40 DATA 205,127,10,14,255,221,33,255,56,237,89,
69,16,254,28,203,155,175,221,190,0,200,24,241
45 'WAIT FOR KEY TO BE PRESSED
50 K$=INKEY$
60 IF K$="" THEN 50
65 'SEND CHARACTER CODE TO MACHINE LANGUAGE ROUTINE
70 N=ASC(K$)
80 X=USR(N)
90 GO TO 50
```

Program Listing 4. Basic routine for keyboard controlled tones.

```
                ORG   7D00H
7D00 CD7F0A     CALL  0A7FH
7D03 0EFF       LD    C,0FFH
7D05 DD21FF38   LD    IX,38FFH ;ADDRESS FOR SENSING ANY KEY DOWN
7D09 ED59   NEXT OUT  (C),E    ;OUTPUT INDIRECTLY TO CASSETTE RECORDER PORT
7D0B 45         LD    B,L
7D0C 10FE   LOOP DJNZ LOOP
7D0E 1C         INC   E
7D0F CB9B       RES   3,E      ;MAINTAIN NORMAL SIZE CHARACTER DISPLAY
7D11 AF         XOR   A
7D12 DDBE00     CP    (IX+0)   ;RETURN TO BASIC ROUTINE
7D15 CA         RET   Z        ;ONLY WHEN KEY RELEASED
7D16 18F3       JR    NEXT
```

Program Listing 5. Assembly language and machine code for keyboard routine.

Basic routine and silence reigns until we press the next key.

From the Basic listing, we see that lines 20-40 load in the machine code and the program loops between 50 and 60 until any key is pressed. Line 70 returns the ASCII code of the corresponding key as N and line 80 calls the machine language routine, again passing to it the parameter N.

After loading register L with N by calling the subroutine at hex address 0A7F, the C-register is loaded with the device address 255 and a 16-bit index register, IX, with the keyboard address hex 38FF.

The OUT(C),E instruction outputs the number in E to the device whose address is stored in C. Register E is being used to output the alternating 1s and 0s. The accumulator A, used for this purpose in our previous programs has another function here.

The instruction RES3,E (reset bit 3 of E) is an optional one which I threw in after noticing that the video screen alternated between normal size and double size character width while the program was running. The hardware diagrams show that bit 3 of the word being outputted controls the character size mode select; by resetting bit 3 after incrementing the 8-bit number we keep the screen in the normal size character mode. This is advantageous if you wish to add software to display information while the tones are sounding.

The program segment

```
XOR     A
CP      (IX+0)
RET     Z
```

returns control to the Basic routine if the number found in the keyboard address hex 38FF is zero. The keyboard hardware (again see the TRS-80 Reference Handbook) will return zero to this address only when no key is down; hence, this is an easy way to sense the release of a key. If the key is still down, we remain in the machine language program, jump back to NEXT, and continue to output 1s and 0s.

The timing analysis is similar to that for the previous programs; for large enough N most of the time is spent executing DJNZ LOOP, at 13 cycles per instruction. When this program is run, we get a different tone for each key pressed—in fact we can also use the Shift key in conjunction with the other keys to give us twice the number of different frequencies. The Shift key does not affect the printing of alphabetic characters since they all print as upper case regardless; however, different ASCII codes are returned depending on whether Shift is down along with the other key. All keys, except Break, can be used here (including Enter and the space bar).

```
10 DEFINT A-Z: DIM B(128)
15 'LOAD PIANO KEYBOARD ROUTINE
20 POKE 16526,0: POKE 16527,125
30 FOR X=32000 TO 32023: READ A: POKE X,A :NEXT X
40 DATA 205,127,10,14,255,221,33,255,56,237,89,
69,16,254,28,203,155,175,221,190,0,200,24,241
45 'CONVERT CHARACTER CODE TO KEY POSITION
50 READ B(8),B(9),B(10),B(13),B(31)
60 FOR I=44 TO 59: READ B(I): NEXT I
70 FOR I=64 TO 91: READ B(I) : NEXT I
80 DATA 25,26,27,38,39,47,12,48,49,10,1,2,3,4,5,
6,7,8,9,11,37,24,28,44,42,30,16,31,32,33,21,34,35,36,
46,45,22,23,14,17,29,18,20,43,15,41,19,40,13
85 'WAIT FOR KEY TO BE PRESSED
90 K$=INKEY$
100 IF K$="" THEN 90
105 'CONVERT KEY POSITION TO FREQUENCY FACTOR AND SEND
 TO MACHINE LANGUAGE ROUTINE
110 Y=ASC(K$): Z=B(Y)
120 N=250-5*Z
130 X=USR(N)
140 GOTO 90
```

Program Listing 6. Basic routine for "piano-like" keyboard.

At this point there is one problem. The ASCII codes are rather unrelated to the physical position of the keys on the keyboard. To make the keyboard really "piano-like" requires a scheme whereby the ASCII codes are converted to a number relating to the physical position of the key. Then we can get a uniform increase (or decrease) in tone frequency as we hit keys in successive positions—as on a piano of course. The routine in Listing 6 accomplishes this. The assembly language routine is the same one used in Listing 5.

The ASCII-code-to-key-position conversion is done by defining an array B(I), where I is the ASCII code and the value of B(I) is the physical position of the corresponding key (lowest position number in upper left of keyboard and highest in lower right. For this example only the lower case characters are used (i.e. Shift key should not be held down): The ASCII codes are 8, 9, 10, 13, 31, 44 to 59, and 64 to 91—a total of 49 keys excluding Break.

The data on line 80 identifies the position from 1 to 49 in order of ascending numerical ASCII code. Lines 50 to 70 of the program assign these position values to the array and line 110 converts the ASCII code Y of the key pressed to its position Z.

Line 120 then converts the position number to the 8-bit delay factor N, which ranges from 245 for key 1 down to 5 for key 49. Since the larger the delay, the lower the frequency, we have the tone frequencies increasing in steps from the upper left to the lower right on the keyboard. As in our previous routine, we then pass N to the machine language routine and return to Basic only when the key is released.

Note that line 120 can be changed to suit one's needs. One useful variation for budding music composers is to choose values of N corresponding to notes on the

```
                        ORG   7D00H
7D00 CD7F0A             CALL  0A7FH
7D03 0EFF               LD    C,0FFH
7D05 DD21FF38           LD    IX,38FFH
7D09 3E01       START   LD    A,1       ;START WITH HIGHEST FREQUENCY
7D0B 55         DURATION LD   D,L       ;LOAD TONE DURATION FACTOR
7D0C 47         BITCMPL LD    B,A
7D0D ED59               OUT   (C),E
7D0F 1C                 INC   E
7D10 CB9B               RES   3,E
7D12 10FE       BITVAL  DJNZ  BITVAL
7D14 15                 DEC   D
7D15 20F5               JRNZ  BITCMPL
7D17 3C                 INC   A         ;DECREASE TONE FREQUENCY
7D18 BC                 CP    H         ;REPEAT UNTIL LOWEST
7D19 20F0               JRNZ  DURATION  ;FREQUENCY IS REACHED
7D1B AF                 XOR   A
7D1C DDBE00             CP    (IX+0)
7D1F C8                 RET   Z
7D20 18E7               JR    START     ;REPEAT SEQUENCE IF KEY HELD DOWN
```

Program Listing 7. Assembly language and machine code for sound effects generation.

scale—this is probably best done using another number conversion data table. One idea which I will leave for you to pursue, is to use alphabetic keys such as A, B, C, . . . G to generate corresponding notes on the music scale. The shift key can then be used to take you up or down one octave. Note that the array B(I) was dimensioned to 128 in line 10 of the program, allowing for use of upper case ASCII character codes.

As a parting shot, here is a suggested modification of Listing 6 that will give you a rather startling array of keyboard generated sound effects:

```
30 FOR X=32000 TO 32033:READ
   A: POKE X,A:NEXT X
40 DATA 205,127,10,14,255,221,33,255,
   56,62,1,85,71,237,89,28,203,155,16,
   254,21,32,245,60,188,32,240,175,221,
   190,0,200,24,231
120 M=INT (ABS ( (Z-2) /8)) :N=256*2
   ↑(M+1)+2↑(Z-8*M)
```

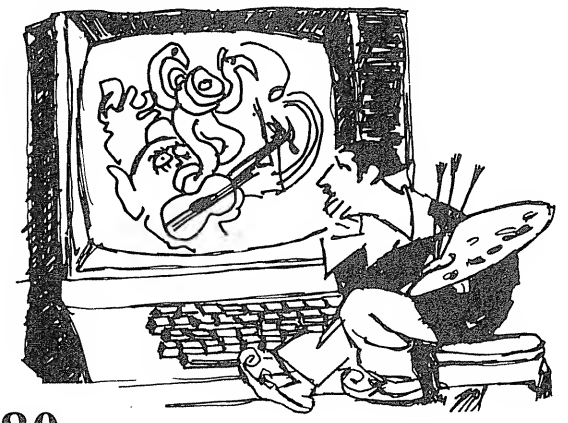
The revised assembly language routine and machine code is shown in Listing 7.

Upon pressing a key, this routine does not merely issue a single frequency, but rather a sequence of tones decreasing monotonically in frequency. The number of cycles of each tone (i.e. its duration) and the number of tones in the sequence are variables dependent on the particular key pressed.

As done previously, line 110 of the Basic routine returns a number Z identifying the physical position of the key pressed. Line 120 calculates a 16-bit number, N, to be pressed to the machine language routine. The lower 8-bits, sent to register L, control the tone duration; the high 8-bits, passed on to register H hold the number of tones in the sequence. The sequence repeats itself over and over so long as the key is held down.

The calculation of N in line 120 insures a wide range of values; as a result, you will be amazed at the variety of sound effects generated by the same routine as different keys are hit. When the duration is relatively long, you will hear the notes distinctly, as if you swept your fingers across a piano keyboard. If the duration is shorter, the tones run together, resulting in bird-chirps, machine-gun, and phaser-like sounds. I'll leave it to you to figure out how the assembly language routine accomplishes all this.

If the examples presented in this article have whetted your appetite, a natural next step would be to write a routine to play a song, where notes have been stored as numbers in memory. If you do try this, one thing to remember is that the duration of each note should not be dependent on its frequency. The sound effects routine just discussed doesn't meet the criterion, since lower notes will last longer than higher notes. Hopefully, I've given you enough to tackle that and other interesting experiments. □



Drawing a grid, a moving ball, and more

Poke Graphics on the TRS-80

James P. Mac Lennan

Introduction

Graphics incorporated in computer programs are always a "plus" in any type of program. They considerably liven up any text program, be it a stoic and serious business program or a fast and frivolous game program. As the old saw says, "A picture is worth a thousand words!"

Most Level II TRS-80 programmers can use SET and RESET for graphs, random points, etc., in simple programs, but some more complex graphics, when drawn with SET commands, are too slow.

The Level II manual mentions POKE graphics for speeding up operation, but does not go much further into this subject. Using ASCII graphics codes, you may speed up graphics displays up to 6 times. The myriad possibilities for this type of graphics deserve looking into much more.

Begin With The Basics

In using POKE graphics, you assign ASCII graphics codes to video memory addresses. The beginning address is 15360, and the end address is 16383. Using the ASCII graphic code 191, which signals "all bits on," type in this short program to show how fast you can fill up the screen:

```
10 FOR X=15360 TO 16383
20 POKE X,191
30 NEXT X
40 GOTO 40
```

Be careful when POKEing any information! An indiscriminate POKE can disrupt your program or something even worse! Don't POKE beyond those limits!

To POKE graphically, you must know what the ASCII graphics codes are, and what character they stand for.

James P. MacLennan, 6073 Hudson Ave., San Bernardino, CA 92404.

Run this program:

```
10  FOR X=129 TO 191
20  PRINT X;:PRINT CHR$(X),
30  NEXT
40  GOTO 40
```

While line 40 is busily looping, a good idea is to copy all the codes with their corresponding numbers on a copy of the Video Display Worksheet (Appendix E of the Level II Manual). This comes in **very** handy for a chronic POKEr, as having to run the display program to find out which character you need gets too tedious to be practical.

Now, armed with a graphics code table and memory address locations, we delve deeper into the POKE graphics game.

Draw That Grid

For some, a basic grid might be drawn in the following manner:

```
10  FOR X = 0 TO 120 STEP 20
20  FOR Y = 0 TO 36
30  SET (X,Y):NEXT Y,X
40  FOR Y = 0 TO 36 STEP 6
50  FOR X = 0 TO 120
60  SET (X,Y):NEXT X,Y
70  GOTO 70
```

This seems a little slow, but by using POKE graphics, the process is speeded up immensely:

```
100  CLS:X=131
110  FOR A=15360 TO 16128 STEP 128
120  FOR Q=A TO A+59
130  POKE Q,X:NEXT Q,A
135  POKE Q,129
140  X=151
150  FOR A=15360 TO 16000 STEP 128
160  FOR Q=A TO A+50 STEP 10
170  POKE Q,X:NEXT Q,A
180  X=149
190  FOR A=15424 TO 16064 STEP 128
200  FOR Q=A TO A+60 STEP 10
210  POKE Q,X:NEXT Q,A
220  FOR A=15420 TO 16124 STEP 128
230  POKE A,X:NEXT*A
240  GOTO 240
```

For specific memory addresses, there is an easy method to get the correct number. Looking back to our video worksheet, we find each "print position" numbered. This numbering is used in determining what the number is that you need for using PRINT@ statements. By simply adding 15360 to this number, you get the correct memory address for that part of the screen.

A Moving Ball

Try this program:

```
5   CLS:DIM A(20)
10  FOR A=1 TO 20
20  READ A(A)
30  NEXT
40  FOR X= 1 TO 20
50  POKE A(X), 140
60  POKE A(X), 32
70  NEXT
80  GOTO 40
100 DATA 15446,15449,15452,15457,
      15460
110 DATA 15526,15593,15658,15722,
      15849
120 DATA  15974,16034,16031,16026,
      16022
130 DATA 15954,15888,15823,15695,
      15570
```

Now let's dissect it line by line.

Lines 10-30 spin an array of specific memory addresses.

Line 40 POKE a set of graphics blocks in the middle of each addressed area.

Line 60 uses the ASCII code for "space" to turn off these blocks.

Line 70 starts the whole process over again.

This method can be put to very interesting uses: spinning a roulette ball, bouncing around a billiard ball, turning a wheel and many more.

One Last Mind-Blower

Type in this program:

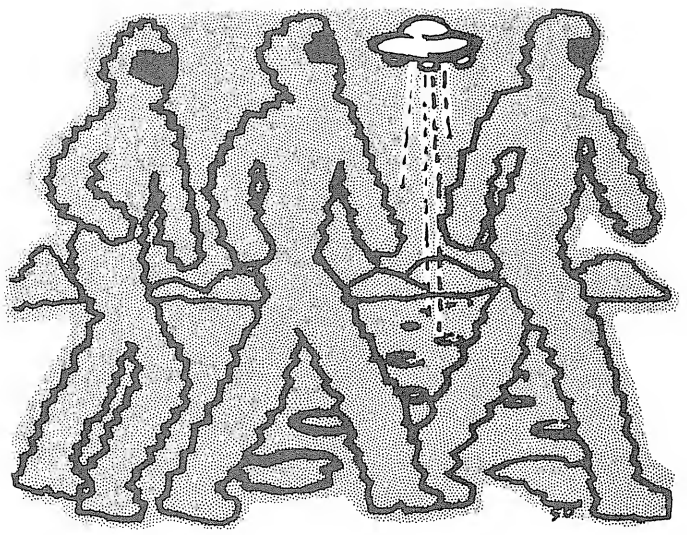
```
10  CLS:A=191:L=15360:B=32
20  R=PEEK(14420)
30  G=L
40  IF R=8 L=L-64:IF L<15360 L=L+64
50  IF R=16 L=L+64:IF L>16383 L=L-64
60  IF R=32 L=L-1:IF L<15360 L=15360
70  IF R=64 L=L+1:IF L>16383 L=16383
80  POKE L,A:IFR=0 THEN 20
90  POKE G,B:GOTO 20
```

Statements 40 through 70 are very important in this type of program. They are used to insure you are not POKEing where you shouldn't.

Here, the PEEK command is used to see what key is being pressed. It is similar to the INKEY command, but in this program you can hold the keys down to move the "paddle." The POKE command's power is exhibited best here. Try to substitute different ASCII values for A and B to draw pictures, etc.

Summary

POKE graphics are a very powerful asset to writing eye-catching programs and save time in drawing graphics. Not many people know how to use them, but they are an important part of Level II's capabilities and can be fun to work with. □



Animation in Level II Basic

Daniel Lovy

Good animation on the TRS-80 is usually limited to one or two moving points. Graphic blocks are SET, a new position is calculated and they are RESET. The result is a dot that appears to move on the screen. This makes for a nice missile or depth charge. Unfortunately, when a larger figure, say six blocks, is to be moved around things do not work out quite right. Six blocks must be SET and RESET and, more importantly, six new positions must be calculated. All this takes time. The animation becomes slow and choppy, even with Level II's faster graphics. For any real-time, arcade style programs, the programmer must either learn machine language or be content with missiles and depth charges.

Yet, do not despair. Level II Basic has a feature that is not documented in the manual. String variables can be used to store graphics. By doing so, space ships and star bases can be PRINTed rather than SET.

As you know (or maybe you don't, so this is a good time to find out), graphics are handled in the TRS-80 by the use of graphic codes. It uses these codes much like ASCII codes. POKEing a 67 into the video memory makes a C appear on the screen. POKEing a number between 129-191 makes a funny looking shape appear. By using the CHR$ function it is possible to store these shapes or groups of shapes as string variables.

EXAMPLE:

```
A$ = CHR$(191) + CHR$(191)
```

(Yes, it is legal)

Daniel Lovy, 2398 Hulett Rd., Okemos, MI 48864.

Now whenever A$ is PRINTed a block of 12 graphic blocks will appear. By using a variable that contains blanks you can get rid of the large square by PRINTing that variable in the same place. This is much easier and quicker that SETting and RESETting 12 blocks separately.

To animate a small figure, first design it and choose the proper graphic codes (more on that later) and store it in a string variable, then set up another variable of blanks to erase it. Use the PRINT@ statement to place the figure where you want it, calculate its next position, then erase it. Doing it this way saves having to individually position each block in the figure.

Using the PRINT@ statement to position and move the object has one drawback. It uses a single number to position characters on the screen. This makes calculation of movement a little cumbersome. A grid type system (two variables) is easier to work with. Here is an equation that will convert from a grid system to a linear one:

```
N = Y*64 + X
```

Y is between 0 and 15

X is between 0 and 63

Movement can be plotted in terms of X and Y, then converted to a single variable to be used with the PRINT@ statement.

Hoping to make things clearer, I've written a very short program which demonstrates this type of SET-less animation.

```
 1  CLS:X = 30:Y = 6: Y1 = 1
10  A$ = CHR$(166) + CHR$(132)
20  B$ = "  "  ' (2 spaces)
30  W = Y*64 + X
40  PRINT@ W,A$;
45  IF Y<2 OR Y>13 THEN 70
47  IF X<2 OR X>59 THEN 80
50  PRINT@ W,B$;
55  X = X + 3*X1:Y = Y + Y1
60  GOTO 30
70  Y1 = -Y1: R = RND(3):
    X1 = 2 - R:GOTO 50
80  X1 = X1: R = RND(3):
    Y1 = 2 - R:GOTO 50
```

Line 1 Sets things up

10 Puts the four block character into A$

20 B$ will be used to erase it

30 Converts from grid coordinates to a linear one

40 Prints all four blocks in one statement

45-47 Checks to see if it should bounce

50 Erases the figure

55 Moves it one more unit on the grid

60 Keeps it moving

70-80 Computes bounce

This type of graphics allows for more experimentation with the shape of the figures. Change line 10 to this;

```
10  A$ = CHR$(RND(62) + 129) +
    CHR$(RND(62) + 129)
```

Now each time the program is run a different shape will be used (delete line 20 and run it).

Not only graphics, but also characters can be animated in this way. Add these two lines:

```
10  A$ = "(-)"
20  B$ = "   "  ' (3 spaces)
```

The only problem that remains now is determining the proper graphic codes to use. Here is a short utility program that takes care of that:

```
10  CLS
20  PRINT@256, "INPUT NUMBER"
30  INPUT N
```

```
40 IF N>30 THEN 100
50 Y = INT(N*.1)
60 X = N - Y*10
70 SET(X,Y)
80 GOTO 20
100 FOR M = 15360 TO 15364
110 A = PEEK(M)
120 PRINT A
130 NEXT
```

Design your figure on a grid like this;

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29

When the program asks for a number, input the numbers from this grid. When you are finished enter a number greater than 30. The proper graphic codes will appear on the screen (32 is a blank).

This type of animation runs much faster than if SET and RESET are used. It gives more time to the other functions of the program. It also proves that the limitations of a computer are related only to how sneaky the programmer can get. □

Draw Art

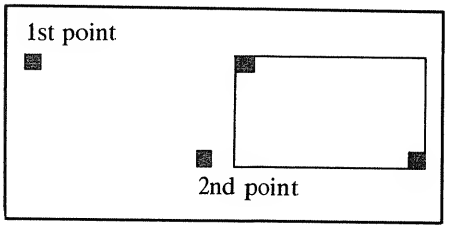
Brad Pitzel

The TRS-80 is not a great graphics machine, but with the help of the following short program, you can easily create some amazing displays. Drawart is a masterpiece in simplicity.

No longer do you have to sit down and plot each pixel that makes up a circle. Nor do you need to figure out the equation of a line just to draw it.

Drawart takes care of these problems and more. By using one-letter commands

Figure 1.



and the cursor control keys, Drawart makes your TRS-80 do all the tedious work; leaving you more time to be creative.

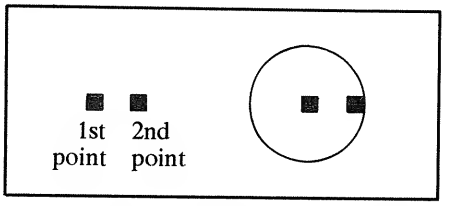
The commands are as follows:

Cursor Control: Pressing one of the arrow keys will move the cursor in the direction indicated on the key. You can't press more than one key at a time.

Draw/Erase Mode: The cursor in Drawart can be set in either the draw or erase mode. Pressing the spacebar reverses the current mode.

Lines: To enter the line mode, press L. Once you are in this mode, you must plot

Figure 2.



two points by using the spacebar and the arrow keys. Press G and these dots will be connected by a straight line in the order in which they were plotted.

Rectangles: Pressing R places you in the rectangle mode. By using the cursor control keys and the spacebar, plot two dots. After you plot the second dot, the computer will fill in the area between the two dots.

Circles: Press C to enter the circle mode. Plot two dots that have the same Y axis. The first point is the center of your circle and the second designates the radius.

Block Move: To store the ASCII values of the entire screen into the memory addresses 26000 to 27023, press P. You can then continue drawing. □

Listing 1.

```
10 CLS : CLEAR : DIM X(255), Y(255), Q(40), W(40)
20 Q=64 : W=24 : SET (Q,W) : X(32)=-1 : X(64)=1
25 Y(8)=-1 : Y(16)=1 : R=1
30 GOSUB 1000 : IF FK=1 THEN R=-R
40 A$=INKEY$ : IFA$="" THEN 30
50 IF A$="L" THEN 2000
60 IF A$=CHR$(31) THEN CLS
70 IF A$="R" THEN 4000
80 IF A$="C" THEN 5000
90 IF A$="P" THEN 6000
100 GOTO 30
1000 FK=0 : A=PEEK(14400) : IF A=0 THEN RETURN
1010 IF R=-1 THEN RESET (Q,W)
1020 Q1=Q : W1=W : Q=Q+X(A) : W=W+Y(A)
1025 IF Q<0 Q=Q+128 ELSE IF Q>127 Q=Q-128
1026 IF W<0 W=W+48 ELSE IF W>47 W=W-48
1030 FK=0 : IF A=128 FOR ZA=1 TO 30 : NEXT : FK=1
1040 SET(Q,W) : RETURN
2000 R=-1 : T=0
2010 GOSUB 1000 : IF INKEY$="G" THEN 2040
2015 IF R=1 AND (Q(T)<>Q OR W(T)<>W) THEN R=-R
2020 IF FK=0 THEN 2010
2030 R=1 : IF Q=Q(T) AND W=W(T) THEN 2010
2035 T=T+1 : Q(T)=Q : W(T)=W : IF T<40 THEN 2010
2040 IF T<2 THEN 30
2050 FOR GT=1 TO T-1 : X=Q(GT) : Y=W(GT)
2055 X1=Q(GT+1) : Y1=W(GT+1)
2060 GOSUB 3000 : NEXT : GOTO 30
3000 IF ABS(X1-X) < ABS(Y1-Y) THEN 3070
3010 FOR X2=X TO X1 STEP 2*(X1<X)+1
```

Brad Pitzel, 122 Meadowbrook Dr., Nova Scotia, Canada.

```
3020 SET(X2,(X2-X)*(Y1-Y)/(X1-X)+Y+.5)
3030 NEXT : RETURN
3070 FOR Y2=Y TO Y1 STEP 2*(Y1<Y)+1
3080 SET(.5+(Y2-Y)*(X1-X)/(Y1-Y)+X,Y2)
3090 NEXT : RETURN
4000 R=-1 : T=0
4010 GOSUB 1000
4015 IF R=1 AND (Q(T)<>Q OR W(T)<>W) R=-1
4020 IF FK=0 THEN 4010
4030 R=1 : IF Q=Q(T) AND W=W(T) THEN 4010
4035 T=T+1 : Q(T)=Q : W(T)=W : IF T=1 THEN 4010
4040 FOR I=Q(1) TO Q(2) STEP SGN(Q(2)-Q(1))
4050 FOR J=W(1) TO W(2) STEP SGN(W(2)-W(1))
4060 SET(I,J) : NEXT J,I : GOTO 30
5000 R=-1 : T=0
5010 GOSUB 1000
5015 IF R=1 AND (Q(T)<>Q OR W(T)<>W) R=-1
5020 IF FK=0 THEN 5010
5030 R=1 : IF Q=Q(T) AND W=W(T) THEN 5010
5035 T=T+1 : Q(T)=Q : W(T)=W : IF T=1 THEN 5010
5040 X=ABS(Q(1)-Q(2)) : Y=X*3/7
5045 KX=Q(1) : KY=W(2)
5050 T=Y*7/3 : IF X>T THEN T=X
5060 FOR HJ=0 TO 2*3.1415 STEP 1/T
5070 Z1=COS(HJ)*X+KX : Z2=SIN(HJ)*Y+KY
5075 IFZ1<0 OR Z1>127 OR Z2<0 OR Z2>47 THEN 5090
5080 SET(Z1,Z2)
5090 NEXT : GOTO 30
6000 FOR I=15360 TO 16383
6005 POKE I+10640, PEEK(I) : NEXT
6010 CLS : FOR I=15360 TO 16383
6015 POKE I, PEEK(I+10640) : NEXT
6020 GOTO 30
```

TRS-80 Graphics Made Almost Painless — Part I

John Crew

This is the first of a three-part series on graphics creation for purposes such as doodling, making limited resolution artwork, and designing graphics for your own programs.

The programs are intended to simplify graphics creation for the above purposes, and should be of interest to novice and expert computer users.

This article describes an etch-a-sketch program that I call Sketch/Print. The second article describes a program, called Vector Plotter, which draws lines between any two points on the screen that you specify. The last article describes a program, Graphics Manager, which stores whatever is on the screen. Graphics Manager can store up to nine pictures, which I call frames, that can have either standard or double width characters. Using Graphics Manager, frames can be saved on tape, loaded from tape, compressed, printed on paper, combined, and have the ASCII number of every byte in them listed.

Sketch/Print and Vector Plotter can be used independently, or you can add Graphics Manager to either. I recommend that you type them as shown alone, test them individually, and correct your typographical errors, and then if you desire, follow the instructions in Part 3 of this series to add Graphics Manager to one or both of the others.

Programmers who have more than 16K of free RAM may want to combine all three programs. That is a fairly simple task if you know Basic well, have about 18K of free RAM, and have an excellent line renumbering program.

System Requirements

All of the programs in this series are written in Level II Basic for an unmodified Model I, 16K, cassette system. All programs except Graphics Manager will easily fit into 4K of free RAM. Using Graphics Manager with one of the other programs requires at least 16K of free RAM, and it is a tight fit so there probably is not enough room for anything else except a very short program such as a simple key debounce program.

To print frames on paper using Graphics Manager you need an MX-80 printer (the basic model without the new features is adequate), or you will have to modify the program. The article on Graphics Manager will describe how frames are stored, which should be very helpful to those who want to modify the program.

Because I used POKE statements and some other tricks which are unique to the previously described system, the programs in this series will probably require modifications to work on a Model III, a differently configured Model I, or other computer.

I spent many hours developing and debugging the programs in this series so I do not think they contain any errors. The programs have many tests to reject clearly erroneous commands. The only

John Crew, 1106 Karin Dr., Normal, IL 61761.

problem I am aware of is that some parts of the programs are not written in the most efficient and neat manner. There are no syntax errors, so do not change any statements which look wrong to you. If you find statements which you think are unnecessary, please leave them alone; there is probably a good reason for their inclusion.

One good book which helped me gain the skills and knowledge which I needed to write these programs was William Barden's *Programming Techniques for Level II Basic* which is sold by Radio Shack. I highly recommend that book to intermediate Basic programmers who want to become more versatile.

Program Modification

The three main programs are written very compactly to save memory space and execution time. To this end I willingly sacrificed some legibility and ease of comprehension. For Basic programs they are rather fast. They could be made even faster if you rewrote them so the subroutines were put as close to the beginning as possible with the most frequently used subroutines first.

Another way to make the programs run faster if you have memory in the expansion interface is to set the memory size to 32769 so Basic will use the faster RAM in the keyboard. When you type them, omit all REM statements.

I strongly urge you to learn about the EDIT mode of Level II Basic before typing any program which has long lines since that knowledge will probably save you much frustration. Where you see what appears to be a long string of blanks, in the program I have used the ↓ key to start a new line on the screen; there are no long strings of blanks in my programs because they waste space and formatted output is better produced with STRING$ or TAB in a PRINT statement, or, of course, PRINT USING.

I used many tricks to save memory, some of which I have rarely or never seen used before. For example, there are only two cases in which a semicolon is needed in a PRINT statement: at the end of a PRINT statement to suppress a line advance, and to indicate the separation of two variable names. Many people use unnecessary semicolons in PRINT statements.

Figure 1 gives examples of compact PRINT statements which are designed to print the current values of two variables named A and B. Notice in particular that the first example will not work as intended; it will print a single value which is the value of a variable named AB if such a variable is used in the

Figure 1. Examples of Compact PRINT Statements.

Compact Form	Functionally Equivalent Form	Separator of Variable Names
PRINTAB	PRINTA B	none
PRINTA;B	PRINT A; B	;
PRINTA%B	PRINTA% B	%, type declaration character
PRINTA" AND "B	PRINT A " AND " B	literal character string
PRINTASIN(C)B	PRINT A SIN(C) B	function name
PRINTATAB(10)B	PRINT A TAB(10) B	Basic keyword

Listing 1. Characters Allowed in Alphanumeric Mode of SKETCH/PRINT.

```
10 CLS: PRINT "CHARACTERS WHICH CAN BE PRINTED IN THE ALPHANUMERIC MODE OF": PRI
NT "SKETCH/PRINT
20 DATA 32, 64, "BLANK SPACE, NUMERALS, AND SOME COMMON NONALPHABETIC SYMBOLS"
30 DATA 65, 90, UPPER CASE ALPHABET
40 DATA 91, 91, UPWARD ARROW
50 DATA 97, 122, LUWER CASE ALPHABET (CONVERTED TO UPPER CASE)
60 CLEAR50: DEFINTA-Y: DEFSTRZ
70 FOR I=1 TO 4: READ B, C, Z: PRINT: PRINT: PRINTZ
80   FOR J=B TO C: PRINT CHR$(J)" ";: NEXT
90 NEXT
99 GOTU 99
```

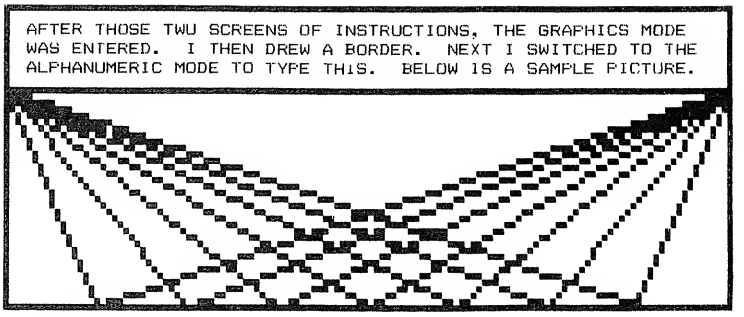
Listing 2. Example of SETting a Point in a Location Occupied by an Alphanumeric Character.

```
10 CLEAR 50: CLS: DEFINT X: DEFSTR A-B: A="THIS IS A TEST"
20 PRINT A: PRINTA: PRINT@ 960, "PRESS ANY KEY EXCEPT 'BREAK' TO CONTINUE.";
30 B=INKEY$:IFB=""THEN30
40 FOR X=0 TO 15: SET(X,5): NEXT
45 PRINT@ 960, CHR$(31);: REM  THIS STATEMENT ERASES THE SCREEN FROM POSITION #
960 TO 1023 (THAT IS, IT ERASES THE LAST LINE).
50 PRINT@ 896, "NOTICE THAT PART OF THE SECOND LINE WAS ERASED WHEN IT WAS
UNDERLINED.";
60 GOTO 60
```

Sample RUN of SKETCH/PRINT.

```
                SKETCH/PRINT BY JOHN CREW 12/24/81
                        GRAPHICS MODE
USE NUMERIC KEYPAD TO MOVE CURSOR AS SHOWN IN DIAGRAM.
USE THOSE KEYS WITH THE SHIFT KEY TO ERASE.
PRESS '.' TO SWITCH TO NONDESTRUCTIVE FLASHING CURSOR.
PRESS 'SHIFT .' TO RETURN TO NORMAL
CURSOR.
PRESS 'SHIFT ←' TO SWITCH TO
ALPHANUMERIC MODE.
PRESS 'S' TO STORE CURSUR LOCATION.
PRESS 'D' TO AUTOMATICALLY DRAW A
LINE FROM THE STORED LOCATION TO THE
PRESENT LOCATION.
PRESS 'E' TO DO SAME AS 'D' BUT ERASE.

                7   8   9
                4   5   6
                1   2   3

          PRESS ANY KEY BUT 'BREAK' TO CONTINUE
```

```
                    ALPHANUMERIC MODE

ALL PRINTABLE KEYBOARD CHARACTERS BUT LOWER CASE MAY BE USED.
PRESS '←' TO BACKSPACE AND ERASE LAST CHARACTER.
PRESS 'SHIFT →' TO RETURN TO GRAPHICS MODE.

                    BOTH MODES

PRESS 'SHIFT CLEAR' TO ERASE THE SCREEN.
PRESS 'ENTER' TO GO TO GRAPHICS MANAGER (IF IT HAS BEEN ADDED).

          PRESS ANY KEY BUT 'BREAK' TO CONTINUE
```



hypothetical main program. If AB is not used elsewhere, Level II will set aside space for AB and set its value to zero. Also notice that putting a space between the A and the B will not affect that PRINT statement, because Level II Basic almost always ignores spaces in statements. The other examples will properly print two values.

About Sketch Print

Most of the etch-a-sketch programs which I have seen range in quality from mediocre to dreadful. I decided to write a better program which would use a small graphics block instead of a large graphics character, which was compact and efficient, and which did not behave oddly when the edge of the screen was reached.

Sketch/Print can work in two different ways (modes): graphics mode and alphanumeric mode. When you run it, you will first see the instructions. Next you will see a small graphics block at the lower lefthand corner of the screen. When that block appears, you are in the graphics mode.

The Graphics Mode

In the graphics mode you can move the cursor using the keys 1-9 on the numeric keypad (or the numeric keys on the main keyboard) in a pseudo-joystick fashion. The 8 key moves the cursor straight up, the 9 moves it diagonally upward to the right, the 6 moves it to the right, etc.

Using the 5 key, you can turn on a graphics block at the current cursor location. The other keys (1-3, 4, 6, 7-9) first move the cursor and then turn on a graphics block. The keys in the numeric keypad will repeat as long as you hold them down. If you find the rate of repetition too fast, increase the value assigned in line 380 to the variable named T and insert this at the beginning of line 350:

T=#:GOSUB410:

(using a digit in place of #).

Listing 3. SKETCH/PRINT

```
100 CLEAR2:DEFINTA-Z:DIMID(8)
105 REM ** LINES 110-150 PROVIDE INSTRUCTIONS
110 CLS:PRINTTAB(13)"SKETCH/PRINT BY JOHN CREW 12/24/81
"TAB(24)"GRAPHICS MODE
USE NUMERIC KEYPAD TO MOVE CURSOR AS SHOWN IN DIAGRAM.
USE THOSE KEYS WITH THE SHIFT KEY TO ERASE.
PRESS '.' TO SWITCH TO NONDESTRUCTIVE FLASHING CURSOR.":@A$="ALPHANUMERIC MODE"
120 @A$="ALPHANUMERIC MODE":PRINT"PRESS 'SHIFT .' TO RETURN TO NORMAL
CURSUR.
PRESS 'SHIFT "CHR$(93)"-' TO SWITCH TO
"@A$".
PRESS 'S' TO STORE CURSOR LOCATION.
PRESS 'D' TO AUTOMATICALLY DRAW A
LINE FROM THE STORED LOCATION TO THE
PRESENT LOCATION.
140 PRINT"PRESS 'E' TO DO SAME AS 'D' BUT ERASE.":PRINT@358,7TAB(50)8TAB(61)9;:P
RINT@550,4TAB(61)6;:FORY=18TO33:SET(103,Y):NEXT:FORX=82TO122:Y=.370079*(X-38):SE
T(X,Y+2):SET(X,25):SET(X,49-Y):NEXT:PRINT@562,5;:PRINT@806,1TAB(50)2TAB(61)3;
145 GOSUB420:CLS:PRINTTAB(23)@A$"

ALL PRINTABLE KEYBOARD CHARACTERS BUT LOWER CASE MAY BE USED.
150 PRINT"PRESS '"CHR$(93)"-' TO BACKSPACE AND ERASE LAST CHARACTER.
PRESS 'SHIFT -"CHR$(94)"' TO RETURN TO GRAPHICS MODE.

"TAB(27)"BOTH MODES
PRESS 'SHIFT CLEAR' TO ERASE THE SCREEN.
PRESS 'ENTER' TO GO TO GRAPHICS MANAGER (IF IT HAS BEEN ADDED).":GOSUB420
153 REM ** END OF INSTRUCTIONS. THE MAIN PART OF THE PROGRAM FOLLOWS. LINES 155-
160 SET VARIABLES TO INTITIAL VALUES, CLEAR THE SCREEN, AND SET A GRAPHICS BLOCK
 IN THE LOWER LEFT-HAND CORNER
155 CLS
160 @A=0:X=0:Y=47:X1=X:Y1=Y:F=0:SET(X,Y):T=9:GOSUB410
165 REM ** LINE 170 ERASES PART OF THE KEYBOARD BUFFER SO THE COMPUTER DOESN'T I
NOW CERTAIN KEYS WERE PRESSED BEFORE
170 FORQE=0TO2:POKEQE+16442,0:NEXT
175 REM ** THE "PEEK" STATEMENT IN LINE 180 RETURNS A 1 IF THE "SHIFT" KEY IS DE
PRESSED, OTHERWISE IT RETURNS 0
180 S=PEEK(14464):@A$=INKEY$:IF@A$=""THEN180ELSEN=ASC(@A$)
183 REM ** LINE 185 CONTAINS TESTS TO INTERPET KEYBOARD INPUT IN THE GRAPHICS AN
D ALPHANUMERIC MODES. MOST OF THAT LINE IS USED FOR ONLY THE GRAPHICS MODE
185 IFS=1ANDN=31THEN155ELSEIF@ATHEN190ELSEIFN=24THEN@A=7ELSEIFN=83THENX1=X:Y1=YE
LSEIFN=68ORN=69THEN195ELSEIFN=46THENF=7:GOTO380ELSEIFN=62THENF=0:GOTO380ELSEIF32
<NANDN<58THENN=N+48*(48<N)+32*(N<42):IFN<10THEN210
187 GOTO170
189 REM ** LINES 190-193 ARE USED ONLY FOR PRINTING A CHARACTER AND MOVING THE C
URSOR IN THE ALPHANUMERIC MODE
```

```
190 IFN=250A=0:GOTO180ELSEIFN=8THENIF1<XTHENX=X-2ELSEIF2<YTHENY=Y-3:X=X+126
192 T=FIX(Y/3)*64+FIX(X/2)+15360:IFN=8THENPUKET,32ELSEIF31<NANDN<128THENPOKET,N+
32*(N>95):IFX<126THENX=X+2ELSEIFY<45THENY=Y+3:X=X-126
193 GOTO180
194 REM ** LINES 195-200 DRAW OR ERASE A LINE AUTOMATICALLY. IF THE REQUESTED LI
NE IS HORIZONTAL OR IF THE INTITIAL AND TERMINAL POINTS ARE THE SAME, LINES 195-
196 ARE USED. OTHERWISE, ONE OF THE TWO LOOPS IN LINES 197-200 DRAWS OR ERASES T
HE LINE
195 S=(N=68):IFY1<>YTHEN197ELSEFORN=X1TOXSTEPSGN(X-X1):(X=X1):IFSTHENSET(N,Y)ELS
ERESET(N,Y)
196 NEXT:GOTO180
197 A!=ABS((X-X1)/(Y-Y1)):IFA!>1THEN199ELSEA!=SGN(X-X1)*A!:B!=X1:FORN=Y1TOYSTEPS
GN(Y-Y1):IFSTHENSET(B!+.5,N)ELSERESET(B!+.5,N)
198 B!=B!+A!:NEXT:GOTO180
199 A!=SGN(Y-Y1)/A!:B!=Y1:FORN=X1TOXSTEPSGN(X-X1):IFSTHENSET(N,B!+.5)ELSERESET(N
,B!+.5)
200 B!=B!+A!:NEXT:GOTO180
205 REM ** THE FOLLOWING IS USED IN THE GRAPHICS MODE TO CHECK THE REQUESTED DIR
ECTION. ALSO, IF THE MOVEMENT IS LEGAL, THE CURSOR IS MOVED AND A GRAPHICS BLOCK
 IS SET, RESET, OR FLASHED). LINES 210-220 SET ILLEGAL DIRECTION FLAGS.
210 FORQE=0TO8:ID(QE)=0:NEXT:IFX=0FORQE=0TO6STEP3:ID(QE)=7:NEXTELSEIFX=127FORQE=
2TO8STEP3:ID(QE)=7:NEXT
220 IFY=0FORQE=6TO8:ID(QE)=7:NEXTELSEIFY=47FORQE=0TO2:ID(QE)=7:NEXT
230 REM ** LINE 250, IF THE REQUESTED MOTION IS ILLEGAL, THE COMPUTER GOES BACK
FOR ANOTHER COMMAND. IF THE DIRECTION IS LEGAL, THE CURSOR MOVEMENT IS PERFORMED
 BY THE LAST PART OF LINE 250 AND BY LINE 260
250 IFID(N-1)THEN170ELSEIFN<4THENY=Y+1ELSEIF6<NTHENY=Y-1
260 IFN/3=FIX(N/3)THENX=X+1ELSEIF(N+2)/3=FIX((N+2)/3)THENX=X-1
330 REM ** NOW THAT THE CURSOR HAS BEEN MOVED, A BLOCK WILL BE SET, RESET, OR FL
ASHED. IF THE FLASHING CURSOR IS ON, LINE 340 DIVERTS PROGRAM EXECUTION TO LINE
380
340 IFFTHEN380ELSEIFSTHENRESET(X,Y)ELSESET(X,Y)
350 GOTO170
360 REM ** LINES 380-400 FLASH THE CURSOR AND GO BACK FOR A NEW COMMAND
380 S=POINT(X,Y):T=1:FORQE=0TO2:IFSTHENRESET(X,Y)ELSESET(X,Y)
390 GOSUB410:IFSTHENSET(X,Y)ELSERESET(X,Y)
400 GOSUB410:NEXT:GOTO170
405 REM ** LINE 410 IS A DELAY SUBROUTINE. LINES 420-430 CONTAIN A SUBROUTINE WH
ICH WAITS FOR A KEY TO BE PRESSED
410 FORQF=0TO10*T:NEXT:RETURN
420 PRINT@971,"PRESS ANY KEY BUT 'BREAK' TO CONTINUE";
430 IFINKEY$=""THEN430ELSERETURN
```

Each time you press a key which moves the cursor, the direction is checked to be sure that it will not move the cursor off the edge of the screen. If the direction is illegal, then the cursor movement command is ignored.

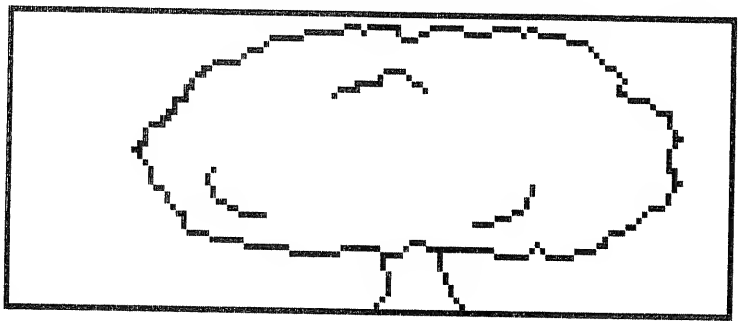
Holding the SHIFT key down while a numeric key is depressed will erase instead of turn on a block.

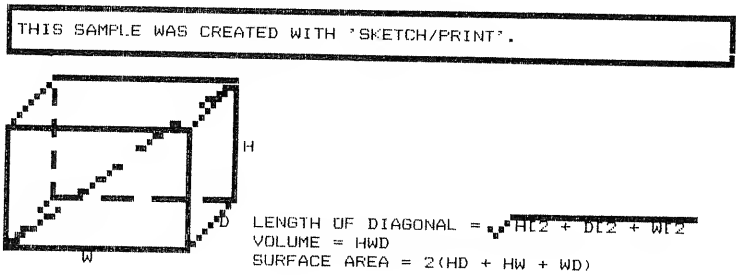
To move the cursor to a different position without erasing or drawing over existing graphics, press the decimal point key. This will switch to a flashing cursor which will not disturb your graphics characters, but you will have to be careful not to move through or under an alphanumeric character (which will be mentioned later in detail). To return to the normal cursor for the graphics mode, press the decimal point and the SHIFT keys simultaneously.

You can also draw and erase lines automatically while in graphics mode. Press the S key to store the current cursor location for later reference when automatically drawing lines. To draw a line from the last stored location to the present location, press the D key.

To erase a line from the last stored location to the present cursor location, press the E key. The stored location is initially set to the coordinates of the lower lefthand corner of the screen.

Two Sample Pictures Made with SKETCH/PRINT.





Switching Modes

To switch to the alphanumeric mode, press the SHIFT key and the ← key. To return to graphics mode, press the SHIFT key and the →.

The Alphanumeric Mode

In the alphanumeric mode you may type on the screen any displayable character which is accessible from the keyboard except the lower case letters. Listing 1 displays the characters which may be used. Any lower case letter which you use will be converted to upper case. The left arrow key may be used to backspace and erase the last character. You cannot go beyond the top or bottom of the screen.

Restrictions

There are two restrictions imposed on the program by the design of the Model I.

The first restriction is that, as noted above, lower case letters can't be used in the alphanumeric mode. This is because the unmodified Model I has only seven instead of eight bits for each location in the video memory (bit 6 is not stored). You will see a character with an ASCII number 64 less than the one you POKEd if you POKE a character with an ASCII code of 96-127 or 192-255. If you POKE an ASCII value less than 32, you will see a character with an ASCII number 64 greater than the POKEd code.

Line 192 contains the POKE statement which puts an alphanumeric character into video memory and also converts lower case letters to upper case. If you have a working lower case modification, you may want to modify this program to allow lower case letters. I believe that the only change needed is to use the EDIT mode of Level II to delete +32*(N>95) from the second POKE statement in line 192.

The second restriction is that you can't move through or immediately under an alphanumeric character while you are in the graphics mode of Sketch/Print. This is because the Model I has character graphics instead of a separate display mode for graphics such as the Color Computer has.

On the Color Computer you could write a program to draw alphanumeric characters while in one of the high resolution graphics modes, so you could freely draw over the alphanumeric characters. But, alas, on the Model I you are restricted to the characters listed in the C appendix of the Level II reference manual.

The only way to make two characters on the screen appear to overlap is to flash them alternately at high speed, a trick which this program cannot accomplish.

There are 1024 character positions on the Model I screen. Each position can hold either an alphanumeric character or a graphics character. Alphanumeric characters have a blank space below them while graphics characters may fill a character position. The blank space under an alphanumeric character is part of that character.

If you try to SET a point in a position which is occupied by an alphanumeric character, that alphanumeric character will be replaced by a graphics character. Because each alphanumeric character has a blank space below it which is associated with that character, you can't set a point in the blank space below the alphanumeric character.

Program Notes

I wanted the keys which move the cursor in the graphics mode to repeat. This can be accomplished in two ways: by writing a keyboard scan subroutine either in Basic using PEEK statements or in machine language, or by erasing the keyboard buffer used by Level II and using the INKEY$ function to scan the keyboard. I used the second method.

Level II maintains a buffer which contains a record of the last keys pressed. It occupies positions 16438-16444 in memory. This buffer is referred to after the keyboard has been scanned so the computer can determine which *new* key has been pressed. This provides keyboard rollover without elaborate hardware.

If zeroes are POKEd into the keyboard buffer, the computer does not know which keys were previously pressed and recognized. For a more complete description of how the keyboard is used by Level II see the book by William Barden which I mentioned earlier.

You may wonder why I used POKE instead of PRINT@ to put a character on the screen. I used POKE because even if you put a semicolon at the end of a PRINT@ statement the screen will scroll when you put a character at position 1023 (the lower righthand corner of the screen).

The multiple IF-THEN-ELSE statements in Sketch/Print are used to avoid using many GOTO statements to jump past the long list of tests. Using lines which have multiple IF-THEN-ELSE statements saves memory and makes the program run fast. Also, a program written that way will appeal more to programmers who like structured programs.

When you write multiple IF-THEN-ELSE statements on one program line, remember to put the highest priority IF-THEN tests first, and, if the logic is complex, make a flowchart.

Figure 2. Variables Used By Sketch/Print

Type	Name	Primary Use(s)
Integer	X	x-coordinate of cursor (0-127)
	Y	y-coordinate of cursor (0-47)
	QA	Alphanumeric mode flag
	F	Flashing cursor submode flag
	S	Shift key flag, temporary storage
	T	Temporary storage. Used as parameter to set length of time delay and to hold address of cursor in alphanumeric mode
	QE	Loop counter
	QF	Loop counter in delay subroutine
	N	VAL(QA$). A number, 1-9, indicating the direction of movement.
	X1	x-coordinate of stored location used as initial point in line-drawing submode.
	Y1	y-coordinate of stored location used as initial point in line-drawing submode.
	ID(0-8)	Illegal direction flags. Position 4 is always zero.
Single precision	A	Used in line-drawing calculations
	B	Used in line-drawing calculations
Character string	QA	Used for INKEY$ loop. It holds the character obtained from the keyboard.

One of the biggest problems I encountered in writing Sketch/Print was finding an efficient way of testing which directions of motion in the graphics mode were illegal. There are eight different illegal cases. Four of them occur when the cursor is in a corner of the screen. The other four occur when the cursor is at one edge of the screen but not in a corner.

At first I thought eight IF-THEN statements would be needed, but later I thought of a clever method that required only three IF-THEN-ELSE statements. An array with nine elements named ID is used to hold flags which indicate which directions are illegal. Each position in the ID array corresponds to a key in the numeric keypad. The fifth position in the array corresponds to the 5 key which does not move the cursor so it is never an illegal direction.

The IF-THEN-ELSE statement in line 210 checks to see if the cursor is at the left or right edge of the screen and if one of those situations exists, sets appropriate flags indicating which horizontal directions are illegal.

The IF-THEN-ELSE statement in line 220 checks to see if the cursor is at the top or bottom of the screen, and, if one of those situations exists, sets appropriate flags indicating illegal vertical directions.

In line 250, the direction of motion you request is compared with the list of illegal directions. If the direction is illegal, the computer goes back (to line 170) and awaits your next command. When the cursor is in a corner one illegal direction flag is set by both the test for illegal horizontal directions and the test for illegal vertical directions. That is a minor inefficiency.

If you plan to modify Sketch/Print or want to learn how it works, look at Figure 2 which lists the variables used in the program.

You might want to modify Sketch/Print so it would do one, several, or all of the following functions on command: reverse graphics; draw a border; scroll the screen left, right, up, or down; or automatically draw a triangle, rectangle, ellipse, circle, or other figure. □

TRS-80 Graphics Made Almost Painless — Part II

John Crew

This is the second article in a three part series. The first segment appeared in the January 1982 issue. Here we discuss some quirks of Level II Basic and describe Vector Plotter, a program that draws lines on the screen between any two points. Vector Plotter can produce random vectors, or you can supply the x, y coordinates of the initial and terminal points of a vector.

Many people don't know what *vector* means because it is often misused. The correct definition—used by mathematicians, engineers, and scientists other than biologists—is a line which has two properties, length and direction. Some programmers use vector to mean *array.* Airplane pilots and science fiction writers often use it in place of *direction.* Biologists use vector to mean "a disease carrying organism."

John Crew, 1106 Karin Dr., Normal, Il 61761.

Peculiarities of Level II

There are two problems with the VAL function, which are not mentioned in the reference manual. The first problem is that VAL doesn't recognize numeric character strings preceded by a minus sign if there are blanks before the minus sign. That problem was described in Radio Shack's "Microcomputer News" (Oct. and Nov. 1980 issues). The second problem is that if a percent symbol is the first nonblank character after a string of numeric characters, an SN (syntax) error message will be printed when VAL is used on that string.

Apparently the programmers at Microsoft were uncertain about what should be done in this case. They could divide the value by 100 to get a decimal equivalent, they could leave the number as a percentage, or they could have an error result and leave it to you to write an error handling subroutine to perform whichever calculation you want.

They chose the latter option, but for some reason they call it an SN error instead of an FC (illegal function argument) error. Listing 1 demonstrates both problems with the VAL function. Listing 2 shows the extra lines needed to make the program in Listing 1 work as desired.

If you want the decimal equivalent of a percentage instead of the percentage returned by VAL, then use an error handling subroutine like the one in Listing 2 but insert /100 after

```
LV=VAL (LEFT$(B,K-1))
```

in line 100.

Because I often want compact programs, I sometimes use IF-THEN statements with an implied THEN. The Level II reference manual doesn't say when THEN is unnecessary, so by experiment I discovered when it can be omitted.

Figure 1 shows different legal IF-THEN-ELSE statements, most of which use an implied THEN, that don't work

properly.

The first two examples in Figure 2 have the same error. In a compact IF-THEN-ELSE statement with an implied THEN, if the logical expression ends with a string constant, the THEN branch works properly, but the ELSE branch will be ignored. That problem can be solved by reversing the last comparison (so it is "YES"=IN$) or by inserting a comma, blank space, or THEN between the logical expression and the THEN branch.

Another solution would be to put parentheses around the logical expression. That problem is one of the few cases I know of which can be solved by inserting a space; Level II usually ignores spaces.

An odd IF-THEN-ELSE statement I found doesn't have a THEN branch. It is

```
IFA=BELSEPRINT"NOT EQUAL"
```

If the logical expression is true, nothing is done, otherwise the ELSE branch is taken.

I suggest you avoid using unusual forms of IF-THEN or IF-THEN-ELSE statements because line renumbering programs, Basic compilers, and other versions of Basic almost certainly won't allow such things. Besides those problems, using odd formats of Basic statements makes your programs hard to read or debug. Only use unusual forms of Basic statements if you desperately need to save memory or you want your program to be incomprehensible.

Multiple IF-THEN-ELSE statements

I often use complex, multiple IF-THEN-ELSE statements to consolidate a long process into one program line. This eliminates many GOTO statements so the program is easier to read and runs faster. The Level II manual isn't very helpful in explaining how to write complex, multiple IF-THEN-ELSE statements and it even gives an incorrect example on page 4/17. Contrary to the claim in the manual, you can't nest IF-THEN-ELSE statements within an IF-THEN-ELSE statement. (The writers of the manual were thinking of Fortran or perhaps PL/1.) ELSE is matched with the most recent THEN (or implied THEN) in that program line. See the following listings for examples.

How To Use Vector Plotter

In the instructions you will see the maximum number of pairs of endpoints which can be stored (which is the same as the number of vectors which can be stored). The maximum number of vectors is calculated in line 100 based on the

Figure 1.

Examples of Legal IF-THEN-ELSE statements

Example	Indicator of Separation of Logical Expression and THEN Branch	Notes
`IFA=BTHENC=0ELSEC=1`	THEN	
`IFA=B,C=0ELSEC=1`	Comma	A comma works like THEN. This is allowed because some other versions of Basic allow it.
`IFA%=B%C%=0ELSEC%=1`	Type declaration character ($,%,!,#)	
`IFA=1/(B+1)C=0ELSEC=1`	Parenthesis	
`IFA=BPRINT''SAME''ELSE PRINT''DIFFERENT''`	Basic keyword	
`IFA=B:C=0ELSEC=1`	Colon	The colon takes the place of THEN here, but as Figure 2 shows, this doesn't work if you want the THEN branch to be an implied GOTO branch.
`IFA=1B=-1ELSEB=10`	Transition from numeric character string to a variable name which doesn't start with E or D.	
`IFA=0,1000ELSE2000`	Comma	`IFA=0GOTO1000ELSEGOTO2000` is equivalent.
`IF10=A%90ELSE100`	Type declaration character	Equivalent to `IF10=A%GOTO90ELSE100`
`IFA=8/(B+1)100ELSE90`	Right parenthesis	Equivalent to `IFA=8/(B+1)GOTO100ELSE90`

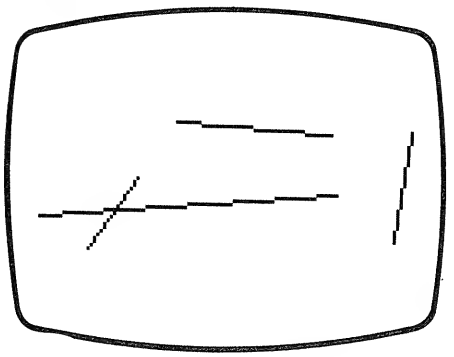
Figure 2.

Examples of IF-THEN-ELSE Statements Which Don't Work as Desired

Example	Separation Indicator	Notes
`IFIN$=''YES''AFFIRM=AFFIRM+1 ELSENEG=NEG+1`	Quotation mark	*
`IFIN$=''YES''PRINT''AGREE''ELSE PRINT''DISAGREE''`	Quotation mark	*
`IFA!=3E=1:PRINT''TRUE''ELSE PRINT''FALSE''`	None	**
`IFA#=8D=-1:PRINT''TRUE''ELSE PRINT''FALSE''`	None	**

First sample run of Vector Plotter.

```
DO YOU WANT RANDOM VECTORS? Y
HOW MANY RANDOM VECTORS? 4

PAIR # 1     PAIR # 3

PAIR # 2     PAIR # 4
```



amount of free memory. The formula was arrived at by experimentation. Some free space is left for use by Basic. The more free memory you have, the smaller the percentage used for the coordinate array.

After the instructions, you are asked if you want random vectors. Usually, my programs look for Y and anything else is treated as no, but in line 150 of Vector Plotter you must reply Y or N.

If you don't want random vectors, you are next asked how many vectors you want to enter. You should ask for as many or more than you expect to enter. The computer then asks for the coordinates of the initial and terminal points. After you enter each pair of endpoint coordinates, the number of endpoint pairs entered so far is printed. You may leave this loop early by pressing the S key instead of entering coordinates. To see instructions, press the H key instead of entering coordinates. H and S are recognized only when they are pressed before you have typed anything in response to an input request.

Entering Coordinates

If you want a nonzero x coordinate, type it first. Then if you want a nonzero y coordinate, type a decimal point followed by the y coordinate. If the y coordinate is only one digit, put a zero between the decimal point and the digit. If you press ENTER without typing anything, the x and y values are set to zero by default. To backspace and erase the last character, press the key as usual.

The x coordinate must be between 0 and 127. The y coordinate must be between 0 (bottom of the screen) and 47 (top of the screen)—the y coordinate is

IFA=BC=-1ELSESTOP	None	***
IFA=B C=-1ELSESTOP	None	***
IF0=TANDS=0STOPELSE100	None	There is a syntax error in the logical expression. The TAN is interpreted as a function name which isn't what I want.
IF0=T AND S=0) THEN80ELSE90	THEN	The logical expression is interpreted the same as: (0=TAN(DS))=0 which is equivalent to NOT(0=TAN(DS)) or 0<>TAN(DS).
IFI=9:10ELSE20	Colon	If true, a SN error results. If false, nothing happens.

* If true, the THEN branch is used; but if false, the ELSE branch is never taken. See article for a list of solutions to this problem.

** The ELSE branch is used if false. Because E or D if used before an exponent the assignment statement is considered part of the preceeding logical expression.

*** This is interpreted the same as ((A=BC)=-1) which has a logical expression within a logical expression. It is equivalent to A=BC.

Figure 3.

Variables Used in Vector Plotter

Type	Name(s)	Use(s)
Integer	X1	X coordinate of initial point.
	Y1	Y coordinate of initial point.
	X2	X coordinate of terminal point.
	Y2	Y coordinate of terminal point.
	N	The number of x,y coordinates stored so far.
	L	The number of x,y coordinates which will be stored.
	I	Count of characters entered in coordinate entry subroutine.
	MV	The maximum number of vectors which can be stored.
	X	Loop counter for drawing lines; it is the current x coordinate of the cursor. Temporary storage.
	Y	Loop counter for drawing lines; it is the current y coordinate of the cursor. Temporary storage.
Single Precision	A	Used in line drawing calculations. It is either the slope or the reciprocal of the slope. Temporary storage.
	B	Used in line drawing calculations, for x or y coordinate. Temporary storage.
	C(0→(MV-1))	Coordinate storage array.
Character String	QA	Last character entered.
	B	String of characters entered in coordinate entry subroutine.

Listing 1. Demonstration of the Two Problems with the VAL Function.

```
10 CLEAR 100:  DEFSTR A-B:  DEFINT I-Z
20 DATA12,-87,1 %,-100 %
30 CLS:  PRINT TAB(6)"EXAMPLE OF TWO PECULIARITIES OF THE VAL FUNCTION":  PRINT
TAB(19)"BY JOHN CREW  11/27/81":  PRINT
40 PRINT"STRING", "LENGTH", "VAL(STRING)"
50 FOR I=1 TO 4:  READ A
60   FOR J=0 TO 1:  B=STRING$(J,32)+A:  PRINT B, LEN(B),
70     V=VAL(B)
80     PRINT V
90   NEXT J
95 NEXT I
```

Listing 2. Extra Lines Needed to Make the Program in Listing 1 Work as Desired.

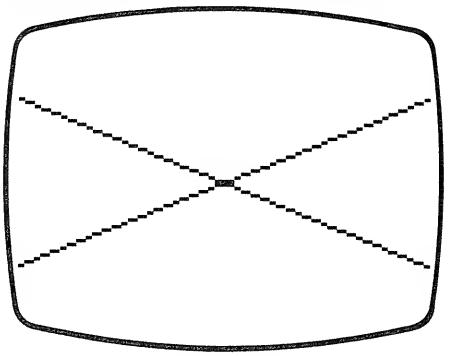
```
15 ON ERROR GOTO 100
64 REM LINES 65-66 REMOVE LEADING BLANKS FROM THE STRING NAMED "B". THIS WILL FI
X THE PROBLEM OF A NEGATIVE NUMERIC STRING BEING IGNORED IF THERE ARE BLANKS IN
FRONT OF IT.
65     IF LEN(B)=0 THEN 70 ELSE P=0:  FOR K=1 TO LEN(B):  IF MID$(B,K,1)<>" " TH
EN P=K-1:  K=256
66     NEXT K:  B=RIGHT$(B,LEN(B)-P)
96 END:  REM LINES 100-110 ARE AN ERROR HANDLING SUBROUTINE. ANY ERROR EXCEPT A
SN ERROR IN LINE 70 IS HANDLED IN THE USUAL WAY. LINE 110 AND THE "ELSE" BRANCH
IN LINE 100 ARE ONLY USED FOR A SN ERROR IN LINE 70.
97 REM IF A SN ERROR IN LINE 70 OCCURS, THE STRING NAMED B IS SEARCHED, FROM LEF
T TO RIGHT, FOR A "%" SYMBOL. THEN THE VAL FUNCTION IS USED ON THE PORTION OF TH
E STRING BEFORE THE "%" SYMBOL.
98 REM A REAL SN ERROR IN LINE 70 WON'T BE TREATED AS SUCH. MAKE SURE LINE 70 IS
 TYPED CORRECTLY BEFORE RUNNING THE PROGRAM.
100 IF ERR/2+1<>2 OR ERL<>70 THEN ON ERROR GOTO 0 ELSE FOR K=1 TO LEN(B):  IF MI
D$(B,K,1)="%" THEN V=VAL(LEFT$(B,K-1)):  K=256
110 NEXT K:  RESUME NEXT
```

Listing 3. Vector Plotter.

```
100 CLEAR12:DEFINTA-Z:RANDOM:MV=FIX((MEM-477)/9)*2!:DIMC!(MV-1):MV=FIX((MV+1)/2!
)
105 REM LINES 110 & 120 PRINT INSTRUCTIONS
110 QA$=" COORDINATES ":CLS:PRINTTAB(23)"VECTOR PLOTTER
"TAB(20)"BY JOHN CREW 11/27/81
THIS CAN WORK IN TWO WAYS:  THE COMPUTER CAN PLOT RANDOM LINES, OR YOU CAN ENTER
 THE X,Y"QA$"OF THE INITIAL AND TERMINALPOINTS.
MAXIMUM NUMBER OF LINES="STR$(MV)".
(0,0) ";
120 PRINT"IS THE LOWER LEFT-HAND POINT. (127,47) IS THE UPPER RIGHT-HAND POINT.
PRESS 'H' FOR HELP INSTEAD OF ENTERING THE"QA$"OF A
POINT.
PRESS 'S' TO STOP BEFORE ENTERING THE NUMBER OF LINES YOU
SPECIFY.
"TAB(13)"PRESS ANY KEY BUT 'BREAK' TO BEGIN":GOSUB446
140 REM LINES 150-220 EITHER GENERATE RANDOM VECTORS OR CALL THE X,Y COORDINATES
 ENTRY SUBROUTINE
150 N=0:CLS:PRINT"DO YOU WANT RANDOM VECTORS? ";
160 GOSUB420:IFQA$="Y"THENX=1:GOTO170ELSEIFQA$="N"THENX=2:GOTO180ELSEIF" "<=QA$P
RINTCHR$(8);
165 GOTO160
170 INPUT"
HOW MANY RANDOM VECTORS";L:IFL<1THEN150ELSEIFMV<LTHEN170ELSE200
180 CLS:INPUT"HOW MANY VECTORS DO YOU WANT TO ENTER";L:IFL<1ORMV<LTHEN180
200 IFX=2QA$="INITI":GOSUB370:B!=A!:QA$="TERMIN":GOSUB370ELSEB!=(RND(48)-1)/100+
RND(128)-1:A!=(RND(48)-1)/100+RND(128)-1
220 C!(N)=B!:C!(N+1)=A!:N=N+2:PRINT"
PAIR #"N/2:IFN/2<LTHEN200
230 CLS' CLEAR SCREEN BEFORE DRAWING VECTORS
233 REM LINES 235-350 DRAW VECTORS
235 FORI=0TON-1STEP2:X1=FIX(C!(I)):Y1=47-FIX(100*(C!(I)-X1+.002)):X2=FIX(C!(I+1)
):Y2=47-FIX(100*(C!(I+1)-X2+.002)):IFX1=X2IFY1=Y2:SET(X1,Y1):GOTO350ELSEFORY=Y1T
OY2STEPSGN(Y2-Y1):SET(X1,Y):NEXT:GOTO350
250 IFY1=Y2FORX=X1TOX2STEPSGN(X2-X1):SET(X,Y1):NEXTELSEA!=ABS((X2-X1)/(Y2-Y1)):I
FA!<=1A!=SGN(X2-X1)*A!:B!=X1:FORY=Y1TOY2STEPSGN(Y2-Y1):SET(B!+.5,Y):B!=B!+A!:NEX
TELSEA!=SGN(Y2-Y1)/A!:B!=Y1:FORX=X1TOX2STEPSGN(X2-X1):SET(X,B!+.5):B!=B!+A!:NEXT
350 NEXT
```

Second sample run of Vector Plotter.

```
DO YOU WANT RANDOM VECTORS? N
HOW MANY VECTORS DO YOU WANT TO ENTER? 2

ENTER INITIAL POINT'S COORDINATES?
ENTER TERMINAL POINT'S COORDINATES? 127.47
PAIR # 1

ENTER INITIAL POINT'S COORDINATES? .47
ENTER TERMINAL POINT'S COORDINATES? 127
PAIR # 2
```



```
360 GOTO360' AFTER DRAWING, LOOP HERE
365 REM SUBROUTINE SECTION FOLLOWS
368 REM LINES 370-410 ARE A SUBROUTINE WHICH GETS AND CHECKS X,Y COORDINATES FRO
M THE USER
370 PRINT"
ENTER "@A$"AL POINT'S ";
375 PRINT"COORDINATES? ";:B$="":I=0
380 GOSUB420:Y=ASC(@A$):IFY=13A!=INT(VAL(B$)*100)/100:RETURNELSEIFY=8IFI=0THEN38
0ELSEI=I-1:B$=LEFT$(B$,I):PRINTCHR$(8);:GOTO380
390 IFI=0IFY=72THEN450ELSEIFY=83IFN=0THENCLS:GOTO360ELSE230
400 IFI=60RY<460R57<YORY=47THENPRINTCHR$(8);ELSEA!=VAL(B$+@A$):IFA!<128ANDA!-FIX
(A!)<.471THENB$=B$+@A$:I=I+1ELSEPRINTCHR$(8);
410 GOTO380
415 REM LINES 420-445 ARE A SUBROUTINE WHICH GETS A CHARACTER FROM THE KEYBOARD
420 PRINTCHR$(95);
430 @A$=INKEY$
440 @A$=INKEY$:IF@A$<" "AND@A$<>CHR$(8)AND@A$<>CHR$(13)THEN440ELSEPRINTCHR$(8);:
IF" "<=@A$PRINT@A$;
445 RETURN
446 @A$=INKEY$' LINE 446 & 447 ARE A PAUSE SUBROUTINE
447 IFINKEY$=""THEN447ELSERETURN
449 REM LINES 450 & 460 PRINT REMINDERS WHEN THE USER ASKS FOR HELP
450 PRINT"
0<=X<=127 AND 0<=Y<=47
ENTER A NUMBER WITH X BEFORE, AND Y AFTER THE DECIMAL. IF Y IS
ONE DIGIT, PUT A ZERO BEFORE IT. FOR X=12 AND Y=8, ENTER '12.08' (WITHOUT QUOTATI
ON MARKS). TO STOP BEFORE ENTERING THE NUMBER OF LINES YOU SPECIFIED, PRESS 'S'."
460 PRINT"PRESS 'H'--FOR HELP.
":GOTO375
```

inverted from the standard Level II use by subtracting the requested y coordinate from 47.

The computer constantly checks to see if the value you typed has an x or y coordinate which is too high or too low. If either the x or y coordinate is too high or too low, the last digit entered is rejected; you aren't allowed to type an illegal coordinate. If you press a key which isn't used for coordinate entry, it is ignored. I have tried to make this program foolproof.

After all the coordinates have been stored, the vectors are drawn. There is a delay of a few seconds before the first vector is drawn.

To quit using Vector Plotter, press the BREAK key. □

TRS-80 Graphics Made Almost Painless — Part III

John Crew

The third in a three-part series, this article describes Graphics Manager, a program that stores whatever is on the screen when it is called. Stored screen images (which I call frames) can be combined, compressed, saved on tape, loaded from tape, and printed on the screen. Graphics characters and ASCII codes can be listed as well. Frames can have either single or double width characters.

John Crew, 1106 Karin Dr., Normal, Il 61761.

The maximum number of frames that can be simultaneously stored in memory depends on the current amount of free string space (frames are stored as strings). The absolute maximum which can be stored is nine because the subroutine which accepts frame numbers uses a single digit. Graphics Manager in Listing 1 clears 6553 bytes of string space which is just enough to hold six frames with single width characters. If some frames are compressed or have double width characters you may be able to store nine frames.

Graphics Manager requires at least 16K of free memory. If you have more, the program will work without modification. The computer for which the programs in this series were written was described in Part 1. You don't need an MX-80 printer unless you want to print frames on paper without modifying the program.

I wrote Graphics Manager to help me design graphics for programs, to manipulate frames, and to provide a

means of printing, recording, and loading frames. Graphics Manager can simplify the design of graphics for TRS-80 programs by allowing you to write a graphics creation program with the slow but versatile SET command.

Add Graphics Manager to that slow graphics creation program. Then list ASCII codes to see the character code and screen position of every character. Next, rewrite the graphics creation program using faster techniques such as POKEing character codes or printing strings of graphics characters.

If you have a favorite computer generated picture, Graphics Manager can print it on paper. If you add my Sketch/Print program (January 1983) you can doodle, make cartoons, create art work, or save screens filled with text and graphics.

The ability of Sketch/Print and Graphics Manager combined to record screens of text and graphics can be used to create a simple educational program. The teacher would type information on the screen as it would appear to the student. He would then record a series of screen contents on tape. Students would use a stripped-down version of Graphics Manager to load and view the frames.

Because Graphics Manager allows printing of some or all frames in forward or reverse order at a rate controlled by the user, students could review a screenful of information as many times as they liked until they understood and remembered it. A student could also print some frames on paper if he wanted to study them later.

This method of teaching merely uses the computer to replace a book and doesn't take advantage of the ability of the computer to ask and answer questions. This approach might be used to teach young children simple concepts by using graphics and words to present the material.

Mistakes In The Manual

Writing Graphics Manager was complicated by the poorly organized, sometimes unclear, sometimes incorrect, and often too brief Level II manual. The quality of the manual varies from section to section. The more I learned about Level II, the more I appreciated Microsoft Basic and the less I appreciated the manual. I'll mention just a few things the manual doesn't cover.

INPUT won't accept more than 240 characters at once, which should rarely be a problem.

The only place you can use TAB in a PRINT USING statement is between PRINT and USING (the only legal form is PRINT TAB (N) USING . .). When you

Listing 1. GM (Graphics Manager).

```
0 CLEAR6553:DEFINTA-Z:MF=8:DIMSC$(4,MF),CM(MF):OB$="PRECSGUDTAOL"
:REM *** THIS LINE SHOULD GO BEFORE ANY OTHER TO SET ASIDE THE STORAGE NEEDED BY
 GM. IF THIS ISN'T THE VERY FIRST LINE, AT LEAST MAKE SURE IT IS EXCECUTED BEFOR
E CALLING GM
32049 END
:REM *** THIS PREVENTS A PROGRAM FROM UNEXPECTEDLY ENTERING GM. IF YOU ARE SURE
THAT WON'T HAPPEN, DELETE THIS LINE
32050 GOSUB32680:IF0DTHEN32620ELSE0E=FC:GOSUB32630
:REM IF THERE IS ROOM, STORE THE CURRENT SCREEN'S CONTENTS
32080 REM *** LINES 32090-32100 PRINT THE MENU
32090 CLS:PRINTTAB(13)"GRAPHICS MANAGER BY JOHN CREW 2/3/82
"STRING$(64,143)"A - PRINT ASCII CODES
C - COMPRESS FRAME(S)
D - DUPLICATE A FRAME
E - ERASE FRAME(S)
G - REVERSE GRAPHICS
L - LOAD FRAME(S) FROM TAPE
P - PRINT FRAME(S)
32100 PRINT"Q - QUIT
R - RETURN TO MAIN PROGRAM
S - SAVE FRAME(S) ON TAPE
T - TRADE (SWAP) TWO FRAMES
U - UNITE (COMBINE) TWO FRAMES
"STRING$(64,143)TAB(15)CHR$(27)FRE("")"FREE BYTES OF FRAME STORAGE
COMMAND? ";:GOSUB32680
32110 REM *** LINES 32120-32134 WAIT FOR YOU TO PRESS A KEY, THEN IT IS CHECKED
FOR LEGALITY, AND, IF LEGAL, GM GOES TO THE CHOSEN OPTION. IF YOU TRY TO DO ANYT
HING BESIDES LOAD, QUIT OR RETURN WHEN NO FRAMES ARE STORED, YOU'LL GET AN ERROR
 MESSAGE
32115 REM *** IF YOU TRY TO UNITE OR LOAD FRAMES WHEN FRAME STORAGE IS FULL OR T
HERE ISN'T ENOUGH ROOM, YOU'LL GET AN ERROR MESSAGE. IF YOU TRY TO DUPLICATE FRA
MES, YOU'LL BE TOLD LATER IF THERE IS ENOUGH ROOM OR STORAGE IS FULL
32120 GOSUB32410:QA=ASC(QA$):IFQDANDQA=76THEN32620ELSEIFFC<2ANDQA=84ORQA=85)PRI
NT"
TWO OR MORE FRAMES MUST BE STORED":GOTO32440
32125 REM *** LINES 32130-32134 BRANCH TO THE SELECTED OPTION. IF THERE IS NO MA
TCH, YOU PRESSED AN ILLEGAL KEY SO GM GOES BACK FOR YOU NEXT COMMAND. IF YOU ASK
 FOR LOAD, THE 'ON-GOTO' LIST OF LINE NUMBERS WILL BE EXCEEDED SO THE NEXT LINE
WILL BE EXECUTED
32130 QB=0:FORQE=1TO12:IFQA$=MID$(QB$,QE,1)THENQB=QE:QE=12
32132 NEXT:IFQB=QPRINTCHR$(8);ELSEIFFC=QANDNOT(QA=76ORQA=82ORQA=81)GOSUB32670:GO
TO32090
32134 ONQB+1GOTO32120,32200,32397,32350,32390,32240,32388,32380,32195,32340,3227
0,32320
32135 REM *** LINES 32195-32399 PERFORM THE OPTIONS. LINES 32400-32690 ARE SUBRO
UTINES COMMONLY USED
32137 REM *** LOAD FRAMES
32140 QA$="LOAD FRAME(S) FROM TAPE":GOSUB32610:IFQDTHEN32090ELSEQD=(FRE("")-408)
/1024:IFQD+FC>MFTHENQD=MF-FC+1
32160 PRINT"
HOW MANY FRAMES DO YOU WANT LOADED";:GOSUB32400:QG=VAL(QA$):IFQG>1THENQE=QG-1ELS
EIFQD<QGTHENQE=QG+FC-2ELSE32190
32170 GOSUB32530:PRINT" ONLY ROOM FOR"QD"MORE FRAME(S)":GOTO32160
32190 PRINT"
INSERT TAPE AND PRESS PLAY BUTTON":GOSUB32550:FORQE=FCTOFC+QG-1:INPUT#-1,CM(QE),
SC$(0,QE):FORQF=1TO4:INPUT#-1,SC$(QF,QE):NEXT:FC=FC+1:PRINT"FRAME #"FC"LOADED":N
EXT:POKE16553,255:GOTO32430
:REM THE POKE STATEMENT CORRECTS THE READ-DATA BUG
32193 REM *** DUPLICATE A FRAME
32195 QA$="DUPLICATE A FRAME":GOSUB32610:IFQDTHEN32090ELSEQC=6:GOSUB32510:GOSUB3
2690:IFQH<QDORMF<FCTHEN32620ELSEFORQF=0TO4:SC$(QF,FC)=SC$(QF,QE):NEXT:CM(FC)=CM(
QE):FC=FC+1:GOTO32090
32197 REM *** PRINT FRAMES ON THE SCREEN OR ALSO ON A MX-80 PRINTER. WHEN USING
THE MX-80, YOU CAN HAVE NORMAL WIDTH OR COMPRESSED CHARACTER WIDTH. YOU'LL GET A
N ERROR MESSAGE IF THE PRINTER ISN'T READY
32200 CLS:PRINT"PRINT FRAME(S)":GOSUB32490
32210 QA$="COPY ON MX-80":GOSUB32610:QI=QD:PRINT:IFNOTQITHENIFPEEK(14312)<>63THE
NPRINT"PRINTER NOT READY":GOSUB32480:GOTO32210ELSEQA$="COMPRESSED CHARACTERS":GO
SUB32610:IFQDTHENLPRINTCHR$(18);ELSELPRINTCHR$(15);
32220 PRINT"
NOW AND AFTER EACH FRAME IS PRINTED, ";:GOSUB32550:FORQE=QATOQBSTEPQC:GOSUB32590
:IFQITHEN32228ELSELPRINTSTRING$(2,10):FORQF=15360TO16320STEP64:IFCM(QE)=2LPRINTC
HR$(14);
32224 LPRINTSTRING$((4-14*NOTQD)*(3-CM(QE)),32);:FORQG=0TO63STEPCM(QE):QH=PEEK(Q
F+QG):LPRINTCHR$(QH-32*(QH>127));:NEXT:LPRINT:NEXT:LPRINTSTRING$(2,10)
32228 GOSUB32420:NEXT:GOTO32090
32230 REM *** SAVE FRAMES ON TAPE
32240 QA$="SAVE FRAME(S) ON TAPE":GOSUB32610:IFQDTHEN32090ELSEGOSUB32490
32250 REM *** THE 'OUT255,4' STATEMENT IN LINE 32260 TURNS ON THE CASSETTE MOTOR
 (IF THE REMOTE JACK IS PLUGGED INTO THE RECORDER) SO BLANK SPACE IS LEFT AFTER
EACH FRAME
32260 PRINT"ADVANCE TAPE TO A BLANK SPACE":GOSUB32550:FORQE=QATOQBSTEPQC:PRINT#-
1,CM(QE),CHR$(34)SC$(0,QE):FORQF=1TO4:PRINT#-1,CHR$(34)SC$(QF,QE):NEXT:OUT255,4:
PRINT"FRAME #"QE+1"SAVED":GOSUB32480:NEXT:GOTO32430
32265 REM *** LIST ASCII CHARACTER CODES, SCREEN POSITION, AND MEMORY ADDRESS OF
 EACH SCREEN POSITION FOR EVERY CHARACTER IN A FRAME. THIS INFORMATION IS HELPFU
L IN DESIGNING GRAPHICS PROGRAMS USING 'PRINT@' AND/OR 'POKE'
32270 QA$="PRINT ASCII CODES":GOSUB32610:IFQDTHEN32090ELSEGOSUB32490:FORQE=QATOQ
BSTEPQC:CLS:PRINTTAB(27)"FRAME #"QE+1"
SCREEN POSITION"TAB(19)"SCREEN ADDRESS"TAB(38)"CHARACTER"TAB(54)"ASCII CODE"STRI
NG$(64,143);
```

```
32290 IFCM(0E)=2THENPRINTTAB(7)"DOUBLE WIDTH CHARACTER MODE (32 CHARACTERS/LINE)
   ONLY EVEN NOMBERED BYTES ARE OSED TO STORE THE CHARACTERS"ELSEPRINTTAB(22)"64
 CHARACTERS/LINE
32300 GOSUB32480:0H=0:FOR0F=0TO4:FOR0G=1TOLEN(SC$(0F,0E)):0A$=MID$(SC$(0F,0E),0G
,1):0D=ASC(0A$):IF0D<>192PRINTUSING"    ####";0H;:PRINTTAB(22)0H+15360;
32303 IF32<0DAND0D<192PRINTTAB(42)0A$;
32304 PRINTTAB(57)USING"###";0D:IF191<0DTHEN0D=0D-192ELSE0D=1
32305 0H=0H+CM(0E)*0D:NEXT:NEXT:GOSOB32480:NEXT:GOTO32090
32310 REM *** 0OIT (EXIT PROGRAM). THE HUGE AMOUNT OF STRING SIORAGE OSED BY GM
IS RELEASED AND ALL VARIABLES ARE ERASED. THE PRINTER IS SET TO 80 CHR/LINE IF I
T'S ON
32320 0A$="0UIT PROGRAM":GOSOB32610:IF0DTHEN32090ELSECLEAR50:IFPEEK(14312)<>15PO
KE14312,18
32330 END
:REM *** IF YOU WANT GM TO ERASE ITSELF, CHANGE 'END' TO 'NEW' (WITHOUT 0UOTATIO
N MARKS)
32335 REM *** TRADE (SWAP) 2 FRAMES BY EXCHANGING SUBFRAME STRING ADDRESSES
32340 0A$="TRADE TWO FRAMES":GOSUB32610:IF0DTHEN32090ELSE0C=5:GOSOB32570:FOR0E=0
TO4:0C=VARPTR(SC$(0E,0A)):0D=VARPTR(SC$(0E,0B)):FOR0F=0TO2:0G=PEEK(0C+0F):POKE0C
+0F,PEEK(0D+0F):POKE0D+0F,0G:NEXT:NEXT:0C=CM(0A):CM(0A)=CM(0B):CM(0B)=0C:GOTO320
90
32345 REM *** ERASE FRAMES. THIS RECOVERS THE SIORAGE USED BY THE ERASED FRAMES
32350 0A$="ERASE FRAME(S)":GOSUB32610:IF0DTHEN32090ELSE0D=FC-1:GOSUB32490:IF0B<0
ATHEN0C=0A:0A=0B:0B=0C
32360 0C=0B-0A+1:IF0B=0DTHEN32370ELSEFOR0E=0ATO0D-0C:CM(0E)=CM(0E+0C):FOR0B=0TO4
:0F=VARPTR(SC$(0B,0E)):0G=VARPTR(SC$(0B,0E+0C)):FOR0H=0TO2:POKE0F+0H,PEEK(0G+0H)
:NEXT:NEXT:NEXT
:REM *** THIS LINE MOVES FRAMES DOWN IN THE ARRAY IF NECESSARY
32370 FOR0E=0DTOFC-0CSTEP-1:FOR0B=0TO4:SC$(0B,0E)="":NEXT:NEXT:FC=FC-0C:GOTO3209
0
:REM *** THIS LINE ERASES THE FINAL FRAME(S)
32375 REM *** UNITE (COMBINE) TWO FRAMES BY PRINTING THE FIRST, MERGING GRAPHICS
 CHARACTERS AND PUTTING NONBLANK CHARACTERS FROM THE SECOND FRAME INTO CORRESPON
DING BLANK POSITIONS IN THE FIRST
32380 0A$="UNITE TWO FRAMES":GOSUB32610:IF0DTHEN32090ELSE0C=4:GOSUB32570:IFCM(0A
)<>CM(0B)PRINT"
BOTH FRAMES MUST BE THE SAME CHARACTER MODE":GOSUB32480:GOTO32380ELSEIFMF<FCORFR
E("")<1024/CM(0A)+408THEN32620
32382 0E=0A:GOSUB32590:0G=15360:FOR0E=0TO4:FOR0F=1TOLEN(SC$(0E,0B)):0D=ASC(MID$(
SC$(0E,0B),0F,1)):IF191<0DTHEN0H=0D-192:GOTO32386ELSE0H=1:0C=PEEK(0G):IF(0C=320R
0C=128)AND31<0DAND0D<128POKE0G,0D+32*(95<0D):GOTO32386
32383 IF0C=320C=128
32384 IF0D=320D=128
32385 IF127<0CAND0C<192AND127<0DAND0D<192POKE0G,0C0R0D
32386 0G=0G+CM(0B)*0H:NEXT:NEXT:GOTO32050
32387 REM *** REVERSE GRAPHICS CHARACTERS WHILE LEAVING OTHER CHARACTERS UNCHANG
ED
32388 0A$="REVERSE GRAPHICS":GOSUB32610:IF0DTHEN32090ELSEGOSUB32490:FOR0E=0ATO0B
STEP0C:GOSUB32690:IF0H<1024/CM(0E)-0DTHEN0D=0B:NEXT:GOTO32620ELSEGOSOB32590:FOR0
F=15360TO16383STEPCM(0E):0G=PEEK(0F):IF127<0GAND0G<192POKE0F,319-0GELSEIF0G=32PO
KE0F,191
32389 NEXT:GOSUB32630:NEXT:GOTO32090
32390 0A$="COMPRESS FRAME(S)":GOSUB32610:IF0DTHEN32090ELSEGOSUB32490:CLS:FOR0E=0
ATO0BSTEP0C:FOR0F=0TO4:0G=1:0I=0:0D=0:PRINT@0,"SUBFRAME POSITION"1:GOSOB32396
:REM *** LINES 32390-32396 COMPRESS FRAMES AND KEEP THE USER INFORMED OF PROGRES
S
32391 0H=ASC(MID$(SC$(0F,0E),0G,1)):PRINT@17,0G" ";:IF0H=320R0H=1280R0H=193THEN0
D=0D+1:IF0D=1THEN0I=0G:GOTO32393ELSEIF0D=630R0G=LEN(SC$(0F,0E))THEN0G=0G+1:GOTO3
2395ELSE32393
:REM *** SEARCH FOR A BLANK. WHEN FOUND, SEARCH FOR NEXT NONBLANK OR END OF STRI
NG
32392 IF1<0DTHEN32395ELSE0D=0
:REM *** THIS LINE IS REACHED WHEN A NONBLANK CHARACTER IS FOUND. IF THE COUNT O
F CONSECUTIVE BLANK CHARACTERS IS GREATER THAN ONE, COMPRESS THOSE BLANKS
32393 0G=0G+1:IFLEN(SC$(0F,0E))<0GTHENNEXT:NEXT:GOTO32090ELSE32391
:REM *** REPEAT LOOP ONTIL THE END OF THE LAST SUBFRAME IS REACHED
32395 SC$(0F,0E)=LEFT$(SC$(0F,0E),0I-1)+CHR$(192+0D)+RIGHT$(SC$(0F,0E),LEN(SC$(0
F,0E))-0G+1):0G=0I:0I=0:0D=0:GOSUB32396:GOTO32393
:REM *** THIS LINE DOES THE ACTUAL COMPRESSION
32396 PRINT@64,"LENGTH OF FRAME"STR$(0E+1)", SUBFRAME"0F+1"="LEN(SC$(0F,0E)):RET
URN
:REM *** TELL USER HOW LONG A SOBFRAME IS. THIS IS OSED BEFORE COMPRESSION AND E
ACH TIME THE LENGTH IS CHANGED
32397 0A$="RETURN TO MAIN PROGRAM":GOSUB32610:IF0DTHEN32090ELSEPRINT"
PRINT FRAME BEFORE RETORNING (Y/N)";:GOSUB32400:IF0A$<>"Y"THEN0E=-1:CLSELSEIFFC=
0THENGOSUB32670:0E=-1:CLSELSE0C=3:GOSOB32510:GOSUB32590
:REM *** POT FRAME ON SCREEN IF ASKED
32399 RETURN
:REM THIS LINE SENDS GM BACK TO THE GRAPHICS CREATION PROGRAM
32400 PRINT"? ";
:REM *** PRINT PROMPT. LINE 32400-32425 GET A CHARACTER FROM THE KEYBOARD. THIS
SUBROUTINE IS USED FREQUENTLY IN GM
32410 PRINTCHR$(95);
:REM *** PRINT CURSOR. ANOTHER CHARACTER COULD BE USED AND PROBABLY SHOULD BE SO
 YOU CAN TELL BY LOOKING AT THE CORSOR THAT A INKEY$ SUBROOTINE IS BE RON
32420 0A$=INKEY$
:REM *** THIS MAKES THE COMPUTER 'FORGET' ANY KEYS PRESSED BEFORE REACHING THIS
LINE
32425 0A$=INKEY$:IF0A$<" "THEN32425ELSEPRINTCHR$(8)0A$;:RETURN
32427 REM *** LINES 32430-32440 TELL THE USER WHEN LOADING OR RECORDING OF FRAME
S IS DONE
32430 CLS:PRINTCHR$(23):PRINT@538,"DONE"
```

add or delete lines from a program, Level II moves the rest of the program around as needed so that the pointers to the next line are always in ascending order. PRINT TAB (N) works much like PRINT STRING$(N,"") for N < 64.

The Edit mode of Level II can be used to find lower case letters.

FRE("") can be used instead of something like FRE ("A") which saves one byte. FRE(0) works the same way as MEM.

You can use an arithmetic expression such as ERROR N/10+2 after CLEAR or ERROR. If you use an arithmetic expression after ERROR and you are told there is a syntax error in the line where ERROR appears, don't pay attention to that message.

If you put spaces between GO and TO, they are removed so GOTO is always one word.

When you print a number, Level II won't print it on the current line if there isn't room for the entire number. When you record a string with leading blanks or one which contains a comma or colon, you should record that string with a quotation mark at the front. If you record a string on tape with a quotation mark at the beginning and a quotation mark before the end, you'll get an FD (bad datum in file) error when you try to load it, and only the part of the string between the first and second quotation marks will be loaded.

Using Graphics Manager

An external program that calls Graphics Manager must be added to Graphics Manager for it to work. Graphics Manager appears in Listing 1.

The program doesn't require you to press the ENTER key when you are typing in information. It quickly reacts to the pressing of a key and either accepts or rejects it.

The menu lists all primary options and tells you how much free frame storage is left, permanently reserving 408 bytes for workspace. When you see that 408 or 409 bytes of frame storage remain, only the workspace is left and no more frames can be stored unless you make more room by compressing or erasing some frames. To select one of the options listed in the menu, press the key which represents your choice. Next to each letter which represents an option is a short description. If you press a key other than the ones used to represent options, it will be ignored.

Storing, compressing, and reversing graphics characters, and uniting two frames are somewhat slow (longer than 10 seconds). Graphics Manager performs these operations visibly so you can

tell how near completion they are. As each character in a frame is stored, it is erased on the screen. The reversal of graphics characters and the union of two frames are also performed on the screen so you can see how much has been done.

For every option except the frequently used print option, you are asked to confirm your choice. This enables you to return to the menu if you pressed the wrong key or changed your mind.

Stored frames are referred to by using a number. The number of a frame is between 1 and the count of currently stored frames. When Graphics Manager asks you to enter a frame number, the legal range is printed in parentheses following the request for a number. Some functions ask for one frame number, some ask for two, others ask for the first and last frame numbers for the range of frames.

When you print, erase, compress, save (record on tape), list ASCII codes, or reverse graphics characters, you are asked to enter the starting and final frame numbers of the range of frames you want the function to work on. If the final frame number is greater than the first, you are asked if you want the function done to that range of frames in reverse order. You can have reverse order for any of the options listed above. If, for example, you had six frames stored, you could print 1-6, 6-1, 2-4, 3-3, or some other legal range of frames.

Frame Compression

Frame compression is one of the most important features of Graphics Manager and one of the most difficult to implement. It uses the seldom used space compression characters. Substrings of consecutive blanks (ordinary blank, graphics blank, or CHR$ (193)) are replaced by a compression character. Up to 63 blanks can be replaced by a single compression character. When a compression character is printed, it is expanded to a series of blanks. Frames without two or more consecutive blanks within a subframe are unchanged when you try to compress them. If you want to know how many bytes were gained by compression, note the free frame space before and after compression.

Compressing may give you enough room to store more frames. Printing or combining frames will be much faster if the frames are compressed. Most other parts of Graphics Manager will be slightly faster if some or all frames are compressed. If you want to decompress a frame for some reason, reverse its graphics characters twice. A compressed frame will expand to its original size when its graphics characters are reversed.

```
32440 GOSUB32480:GOTO32090
32470 REM *** LINE 32480 IS A DELAY SUBROUTINE
32480 FORQD=0TO1400:NEXT:RETURN
32485 REM *** LINES 32490-32500 GET THE STARTING AND FINAL FRAME NUMBERS FOR A R
ANGE OF FRAMES
32490 QC=1:GUSUB32510:QA=QE:QC=2:GOSUB32510:QB=QE:PRINT:QC=SGN(QB-QA):IFQC=0THEN
QC=1ELSEIFQC<0THENPRINT" YOU WANT REVERSE ORDER (Y/N)";:GOSUB32400:PRINT:IFQA$<>
"Y"THEN32490
32500 RETURN
32505 REM *** LINES 32510-32520 ARE A MULTIPURPOSE MESSAGE AND FRAME NUMBER ENTR
Y SUBROUTINE. THIS COMPLICATES THE PROGRAM BUT SAVES MEMORY
32510 PRINT:IFQC<4THEN32514ELSEIFQC=6THENPRINT"# OF FRAME TO BE DUPLICATED";ELSE
PRINT"#'S OF 2 FRAMES TO BE ";:IFQC=4THENPRINT"CUMBINED";ELSEPRINT"TRADED";
32511 GOTO32520
32514 IFQC=1PRINT"START";ELSEIFQC=2PRINT"END";ELSEPRINT"PRINTING AND RETURN";
32515 PRINT"ING WITH FRAME #";
32520 PRINT" (1-"STR$(FC)")";:GOSUB32400:QE=VAL(QA$)-1:IFQE+1>FCORQE<0GOSUB32530
:GOTO32510ELSERETURN
32525 REM *** LINES 32530-32540 ARE AN ERROR MESSAGE SUBROUTINE USED WHEN AN ILL
EGAL FRAME NUMBER IS ENTERED
32530 PRINT"
 FRAME #"FC"IS THE LAST
 NUMBER IS TOO ";:IFQE<0PRINT"SMALL"ELSEPRINT"LARGE
32540 RETURN
32545 REM *** LINE 32550 IS A PAUSE SUBROUTINE
32550 PRINT"PRESS ANY BUT A CONTROL
KEY TO CUNTINUE";:GOSUB32400:PRINT:RETURN
32560 REM *** LINES 32570-32580 ACCEPT TWO FRAME NUMBERS. THIS SUBROUTINE IS USE
D BY THE UNITE FRAMES & TRADE FRAMES OPTIONS. A COMMA IS AUTOMATICALLY PUT BETWE
EN THE TWO FRAME NUMBERS YOU TYPE
32570 GOSUB32510:QA=QE:PRINT",";
32580 GOSUB32410:QB=VAL(QA$)-1:IFQB+1>FCORQB<0THENQE=QB:GOSUB32530:PRINT"ENTER 2
ND FRAME #? ";:GOTO32580ELSERETURN
32585 REM *** LINES 32590-32605 PRINT A FRAME
32590 CLS:IFCM(QE)=2PRINTCHR$(23);
32600 FORQF=0TO3:PRINTSC$(QF,QE);:NEXT:PRINTLEFT$(SC$(4,QE),LEN(SC$(4,QE))-1);:Q
G=ASC(RIGHT$(SC$(QF,QE),1)):IF32<QGANDQG<192POKE16384-CM(QE),QG+32*(95<QGANDQG<1
28)
32605 RETURN
32607 REM *** LINE 32610 ASKS THE USER TO CONFIRM HIS CHOICE OF THE DIFFERENT UP
TIONS WHICH GM OFFERS
32610 CLS:PRINTQA$" (Y/N)";:GOSUB32400:QD=QA$<>"Y":RETURN
32615 REM *** LINES 32620-32625 ARE USED WHEN NO MORE FRAMES CAN BE STURED. IF F
EWER THAN 9 FRAMES ARE STORED BUT THERE ISN'T ENOUGH STRING SPACE TO HULD A FRAM
E, YOU'RE TOLD THERE IS 'INSUFFICIENT STORAGE'
32616 REM *** IF 9 FRAMES ARE STORED, YOU'RE TOLD 'FRAME STORAGE IS FULL'
32620 PRINT:IFMF<FCPRINT"FRAME STORAGE SPACE IS FULL"ELSEPRINT"INSUFFICIENT STOR
AGE"
32625 GOSUB32550:GOTO32090
32627 REM *** LINE 32630-32664 STORE A FRAME
32630 QD=15360:CM(QE)=1+PEEK(16445)/8:ONERRORGU1032664:QH=CM(QE)-1:FORQF=0TO4:SC
$(QF,QE)=STRING$((205-QH-(CM(QE)=2))/CM(QE),32):QI=VARPTR(SC$(QF,QE)):QI=PEEK(QI
+1)+256*PEEK(QI+2)+65536*(127<PEEK(QI+2)):FORQG=QH+1TO205STEPCM(QE):POKEQI,PEEK(
QD):QI=QI+1:POKEQD,32
32640 QD=QD+CM(QE):NEXT:IFCM(QE)=2OR(CM(QE)=1ANDQF=3)THENQH=1-QH
32650 NEXT:IFFC=QETHENFC=FC+1:RETURNELSERETURN
32664 IFERR=10ANDERL=32630THENQI=QI-65535:RESUMENEXTELSEONERRORGOTO0:
REM USED WHEN KEYBOARD/EXPANSION INTERFACE MEMORY BOUNDARY IS CROSSED
32669 REM *** LINE 32670 PRINTS AN ERROR MESSAGE
32670 PRINT"
FRAME STORAGE IS EMPTY":GOSUB32480:RETURN
32675 REM *** LINE 32680 EVALUATES A LOGICAL EXPRESSION WHICH CHECKS TO SEE IF A
 FRAME CAN BE STORED
32680 QD=MF<FCORFRE("")<1024/(PEEK(16445)/8+1)+408:RETURN
32685 REM *** LINE 32690 CALCULATES THE LENGTH OF A FRAME AND CALCULATES THE AMO
UNT OF STRING STURAGE WHICH CAN BE USED FOR STORING FRAMES
32690 QD=0:FORQF=0TO4:QD=QD+LEN(SC$(QF,QE)):NEXT:QH=FRE("")-408:RETURN
```

Combination Of Frames

Combination of two frames is done by printing the first frame and then merging graphics characters and putting nongraphics characters from the second frame into blank spaces in the first. You could think of combination as putting the second frame behind the first so the characters in the second frame show through holes in the first. The first frame number you type is the frame which will be printed first. The second is the number of the frame to be combined with the first. A comma is automatically put between the two numbers. The combined frame is stored as a new frame so you can't unite two frames if there isn't enough room to store the result.

Combining two frames may give different results depending on the order of combination. This is because a graphics character in the first frame takes precedence over a nongraphics character in the second, and an alphanumeric character in the first takes precedence over any type of character in the frame.

Combination is faster if the second frame is compressed more than the first. If you want to combine two compressed frames, estimate which is more compressed (the one with the most blank spaces) and type its frame number second. The more compressed frame will be quickly put behind the first.

Figure 1. Merging Graphics Manager with Another Program.

Do the following in the command mode:

1. CLOAD first program
2. FIN=16633: ST=16548
3. PRINT PEEK(ST); PEEK(ST+1). Write down the two numbers printed
4. BRW=PEEK(FIN)<2: POKE ST, PEEK(FIN)-2-256*BRW: POKE ST+1, PEEK(FIN+1)+BRW
5. CLOAD Graphics Manager
6. LIST 0
7. DELETE 0
8. POKE at 16548 and 16549 the two numbers printed in step 3
9. Retype line 0

Figure 2. Adding Graphics Manager to Sketch/Print

1. Merge programs
2. Remove CLEAR2: DEFINTA-Z: in line 100
3. Insert IFN=13ANDS=0,32050ELSEIFN=27,32090ELSE at the beginning of line 185
4. Change line 32399 to 32399 GOTO160

Figure 3. Adding Graphics Manager to Vector Plotter.

1. Merge programs
2. Change line 360 to 360 QA$=INKEY$: IFQA$=CHR$(13)THEN32050ELSEIFQA$=CHR$(27)THEN32090ELSE360
3. Change line 32399 to 32399 GOTO110
4. Remove CLEAR12:DEFINTA-Z: in line 100
5. Add :B$="" after NEXT in line 350
6. Insert IFQE>=0GOSUB32590ELSE before CLS in line 230
7. Add :QE=-1 to end of line 100

Figure 5. Bytes of String Spaces Required for an Uncompressed Frame.

	Subframe Number				
Character width	1	2	3	4	5
Single	205	205	205	205	204
Double	102	103	102	103	102

Figure 4. Variables used by Graphics Manager.

Type	Name	Main Use(s)
Integer	MF	(Maximum number of frames)-1.
	FC	The count of stored frames.
	QA	Number of first frame of range.
	QB	Frame number of final frame in range.
	QC	Increment for loop which processes a range of frames.
	QD	Number of consecutive blanks during compression. Loop counter in delay subroutine. Used to pass a value from main to subroutines and vice versa.
	QE	Loop counter for frame numbers.
	QF	Loop counter for subframe numbers.
	QG	ASC (final character in frame) when printing frames, reversing graphics. Position in subframe during compression.
	QH	ASC (character in frame) when listing ASCII codes, copying a frame on the MX-80, and compressing frames.
	QI	Compressed character flag when using MX-80. VARPTR (SC$(QF,QE)). Address of position in string during frame storage. Position of first blank in subframe during compression.
	CM(0→MF)	Character display mode for each frame. 1 for single width, 2 for double width.
Character string	QA$	Character entered by user, message sent to selection confirmation subroutine.
	QB$	String of letters which represent program options.
	SC$(0→4,0→MF)	Screen character storage; holds frames.

Quitting

You can, of course, quit when not saving or loading frames by pressing the BREAK key, but I suggest you use the quit option instead. When you use the quit option, all variables are erased and the large amount of string space used by Graphics Manager is released. If the printer is turned on, it is set back to 80 characters per line.

Returning To The Graphics Program

I call the graphics creation program the main program even though Graphics Manager may well be longer and more complex. If you want, Graphics Manager can put a frame on the screen before returning. If the graphics program takes advantage of the ability of Graphics Manager to return a frame, you can modify that frame and then store it if there is room. Sketch/Print or Vector Plotter can modify a frame sent by Graphics Manager.

Warning

Don't try to save on tape a frame which contains a quotation mark because you will get an FD error when you try to load it. You can store such a frame and do anything except load it

properly. This problem is a result of the way Level II handles string input.

Error Checking

Graphics Manager does much error checking to avoid having the program fail because you pressed the wrong key or asked for a function under the wrong circumstances. I tried to do a thorough job of making the program reject erroneous information and print a message when it detects an error. Depending on the error and where it occurs, Graphics Manager will either go to the menu or repeat the question you answered incorrectly.

If you ask for any option other than load, return, or quit when no frames are stored, the program will tell you that frame storage space is empty. If you try to store, load, duplicate, or unite two frames when nine frames are stored, the program says frame storage space is full. If there are fewer than nine frames stored and too little memory to hold another frame when you try to do one of the four things just listed, Graphics Manger will tell you that frame space is insufficient to store a frame.

If you ask Graphics Manager to print frames on paper, it checks the printer. If the printer isn't on-line, the program says the printer isn't ready and then asks you again if you want frames printed on the MX-80.

You also get an error message if you try to combine two frames which have different width chacters. Whenever you enter a frame number, Graphics Manager checks to see if it is in the proper range.

Adding Graphics Manager To Another Program

Figure 1 lists the steps you should follow to append Graphics Manager to another program you have recorded separately. These instructions are for a Level II cassette system but should work for most related systems. The program to which Graphics Manager is to be added should have line numbers greater than 0 and less then 32049. You can have line numbers greater than 32690 but those lines will have to be either typed after Graphics Manager has been added or appended to the combination of Graphics Manager and the low-numbered lines of the graphics creation program.

The common method of merging programs in Figure 1 works by setting the address of the beginning of Basic program storage to the address of the end of the program in memory. That moves the beginning of Basic program storage to just after your program so CLOAD, NEW, and LIST won't affect your hidden program. Graphics Manager is loaded next.

Line 0 of Graphics Manager is listed just so you can see it if you don't have a printed copy handy. Next, line 0 is deleted. Then the beginning of Basic program storage is set to its previous value (which you should have written down after step 3). Finally, line 0 is retyped so it is put before the graphics creation program.

The graphics program you combine with Graphics Manager must leave 408 bytes of string storage. If you use an ON ERROR GOTO statement in your graphics program, you must execute that statement every time you return from Graphics Manager because the program uses an error handling subroutine. Don't use any of the following variables: FC%, MF%, QB$, SC$ (0-4, 0-MF%), or CM%(0-MF%).

Figures 2 and 3 give complete instructions for adding Sketch/Print or Vector Plotter. Figures 2 and 3 refer to the merging process described in Figure 1. If you add Graphics Manager to Sketch/Print as shown, you can go to Graphics Manager and store a frame at any time after the instructions by pressing the ENTER key, or if you want to go straight to the menu, press the SHIFT and ↑(escape code) keys. The SHIFT, ↑, and ENTER keys are used the same way if you combine Vector Plotter and Graphics Manager as shown, but you must wait for all vectors to be drawn before those keys are recognized.

Listing 2. Lines which can be added to GM to test GM.

```
10 FORI=1TO10:CLS:IFI/2=FIX(I/2)THENFORQE=0TO127:QF=.370079*QE:SET(QE,QF):SET(QE
,47-QF):SET(QE,0):SET(QE,47):NEXT:FORQE=0TO47:SET(0,QE):SET(127,QE):NEXT:PRINT@3
1,I;:GOTO30
15 IFI/3=FIX(I/3)THENFORQE=1TO7:PRINTSTRING$(128,152);:NEXT:PRINTSTRING$(127,152
);:POKE16383,152:PRINT@31,I;:PRINT@667,CHR$(34)"  THIS FRAME CONTAINS QUOTATION
MARKS"CHR$(34);:GOTO30
16 IFI/5=FIX(I/5)THENPRINTCHR$(23);:FORQE=1TO15:PRINTSTRING$(32,178);:NEXT:PRINT
STRING$(31,178);:POKE16382,178:PRINT@30,I;:GOTO30
20 PRINTCHR$(23);:FORQE=1TO7:PRINTSTRING$(32,32)STRING$(32,146);:NEXT:PRINTSTRIN
G$(32,128)STRING$(31,146);:POKE16382,146:PRINT@30,I;:PRINT@64,CHR$(34)"THIS FRAM
E CONTAINS QUOTATION MARKS"CHR$(34);
30 GOSUB32050
40 NEXTI
```

Listing 3. Assembly Language version of subframe store and scan subroutines.

```
                00100 ;        SUBFRAME STORE, SUBFRAME SCAN (STOSCN)
                00110 ;                BY JOHN CREW
                00120 ;                62 BYTES LONG WHEN ASSEMBLED
                00130 ;                DATE 1/5/82
                00140 ;
                00150 ;*****************************************************
                00160 ;SUBFRAME STORE SUBROUTINE
                00170 ;
                00180 ;REGISTER        USE(S)
                00190 ;
                00200 ; A              CHR MODE INDICATOR, CHR FROM SCREEN, LSB OF
                00210 ;                SCREEN ADDRESS
                00220 ; B              LEN(SC$(QF,QE))
                00230 ; C              INCREMENT FOR SCREEN ADDRESS
                00240 ; DE             ADDRESS OF LOCATION IN SC$(QF,QE)
                00250 ; HL             VARPTR(SC$(QF,QE)), CURRENT SCREEN ADDRESS
                00260 ;*****************************************************
7FC2            00270          ORG     32767-61        ;(TOP OF 16K MEMORY)-61
7FC2 003C       00280 SCRADD   DEFW    15360           ;THIS WILL BE POKED BY GM
                00290          ;DETERMINE WHETHER SINGLE OR DOUBLE WIDTH CHARACTER
                00300          ;ARE BEING DISPLAYED
7FC4 3A3D40     00310          LD      A,(16445)       ;GET CHR MODE INDICATOR
7FC7 0E01       00320          LD      C,1             ;LOAD DEFAULT INCREMENT FOR
                00330                                  ;SCREEN ADDRESS
7FC9 B7         00340          OR      A               ;SET FLAGS
7FCA 2801       00350          JR      Z,CONT1         ;JUMP IF SNG WIDTH CHRS
7FCC 0C         00360          INC     C               ;MAKE INCR 2 INSTEAD OF 1
7FCD 2A2141     00370 CONT1    LD      HL,(16673)      ;GET VARPTR(SC$(QF,QE))
7FD0 46         00380          LD      B,(HL)          ;GET LEN(SC$(QF,QE))
7FD1 23         00390          INC     HL
7FD2 5E         00400          LD      E,(HL)          ;GET LSB OF STRING ADDRESS
7FD3 23         00410          INC     HL
7FD4 56         00420          LD      D,(HL)          ;GET MSB OF STRING ADDRESS
7FD5 2AC27F     00430          LD      HL,(SCRADD)     ;GET CURRENT SCREEN ADDRESS
                00440 ;FILL ONE SUBFRAME WITH CHARACTERS FROM THE SCREEN
7FD8 7E         00450 STORE    LD      A,(HL)          ;GET CHR FROM SCREEN
7FD9 3620       00460          LD      (HL),32         ;ERASE CHR ON SCREEN
7FDB 12         00470          LD      (DE),A          ;STORE CHR IN SC$(QF,QE)
                00480          ;INCREMENT SCREEN ADDRESS
7FDC 7D         00490          LD      A,L             ;GET LSB OF SCREEN ADDRESS
7FDD 81         00500          ADD     A,C             ;INCREMENT LSB OF ADDRESS
7FDE 3001       00510          JR      NC,CONT2
7FE0 24         00520          INC     H               ;ADD 1 TO H BECAUSE OF CARRY
```

```
7FE1 6F        00530 CONT2   LD      L,A             ;PUT NEW LSB IN L
               00540         ;ADJUST DESTINATION POINTER AND COUNTER
7FE2 13        00550         INC     DE              ;INC POINTER TO SC$(0F,0E)
7FE3 10F3      00560         DJNZ    STORE
7FE5 22C27F    00570         LD      (SCRADD),HL     ;STORE SCREEN ADDRESS
7FE8 C9        00580         RET                     ;RETURN TO BASIC
               00590 ;*****************************************************
               00600 ;SUBFRAME SCAN SUBROUTINE
               00610 ;
               00620 ;REGISTER       USE(S)
               00630 ;
               00640 ; A             CHR BEING SEARCHED FOR (A QUOTATION MARK)
               00650 ; BC            LEN(SC$(0F,0E))
               00660 ; DE            ADDRESS OF SC$(0F,0E), VARPTR(SC$(0F,0E))+2
               00670 ; HL            VARPTR(SC$(0F,0E)), ADDRESS OF SC$(0F,0E), A
               00680 ;               0 IS PUT IN HL IF A QUOTATION MARK IS FOUND
               00690 ;               OTHERWISE A NONZERO NUMBER IS LEFT IN HL
               00700 ;*****************************************************
7FE9 2A2141    00710 SCAN    LD      HL,(16673)      ;GET VARPTR(SC$(0F,0E))
7FEC 0600      00720         LD      B,0
7FEE 4E        00730         LD      C,(HL)          ;BC HOLDS LEN(SC$(0F,0E))
7FEF 23        00740         INC     HL
7FF0 5E        00750         LD      E,(HL)          ;GET LSB OF STRING ADDRESS
7FF1 23        00760         INC     HL
7FF2 56        00770         LD      D,(HL)          ;GET MSB OF STRING ADDRESS
7FF3 EB        00780         EX      DE,HL           ;PUT STRING ADDRESS IN HL
7FF4 3E22      00790         LD      A,34            ;ASCII FOR QUOTATION MARK
7FF6 EDB1      00800         CPIR
7FF8 2003      00810         JR      NZ,GOBACK       ;JUMP IF NOT FOUND
7FFA 210000    00820         LD      HL,0000H        ;INDICATE IT WAS FOUND
7FFD C39A0A    00830 GOBACK  JP      2714            ;RETURN RESULT TO BASIC
0000           00840         END
00000 TOTAL ERRORS

CONT1   7FCD
CONT2   7FE1
GOBACK  7FFD
SCAN    7FE9
SCRADD  7FC2
STORE   7FD8
```

Listing 4. Relocating Basic loader for subframe store and scan Machine Language subroutines.

```
5 CLS: PRINT"RELOCATING LOADER FOR STORE & SCAN MACHINE LANGUAGE SUBROUTINES":P
RINTTAB(24)"BY JOHN CREW":PRINTTAB(27)"1/6/82"
10 CLEAR 300: DEFSNG E: DEFINT L, M: PRINT: INPUT"ENTER MEMORY SIZE (IF YOU DO
N'T WANT IT TO BE 32706)";A$: IFVAL(A$)=0 THEN E=32706-2 ELSE E=VAL(A$)-2
20 GOSUB 95: POKE16561,LSB: POKE16562,MSB: CLEAR6553: REM SET MEMORY SIZE AN
D SET ASIDE STRING STORAGE
30 E=PEEK(16561)+256*PEEK(16562)+2: PRINT "MEMORY SIZE =" E: E=E+2: GOSUB 95
: E=E-2: PRINT: PRINT "ADD THIS TO THE BEGINNING OF LINE 32630 IN GM:": PRIN
TTAB(10)"POKE16526," LSB ":POKE16527,"MSB
40 S=0: FOR I=E TO E+61: READ N: S=S+N: GOSUB 105: NEXT: IF S<>4481 THEN CL
S: PRINT"THE SUM OF THE DATA IS SUPPOSED TO BE 4481; NOT"S: END: REM POKE MACHI
NE LAGUAGE PROGRAM INTO BEGINNING OF RESERVED MEMORY
50 GOSUB 95: I=E+20: GOSUB 100: I=E+36: GOSUB 100: REM ADJUST TWO MEMORY REFER
ENCES IN MACHINE LANGUAGE PROGRAM
60 PRINT: PRINT"MACHINE LANGUAGE PROGRAM HAS BEEN POKED INTO MEMORY STARTING AT
 LOCATION" STR$(E)"."
70 PRINT: PRINT"PREPARE GRAPHICS MANAGER FOR LOADING.
PRESS ANY KEY EXCEPT 'BREAK' WHEN YOU ARE READY TO LOAD GM."
80 A$=INKEY$
90 IF INKEY$=""THEN 90 ELSE PRINTTAB(22)">> NOW LOADING <<": CLOAD
95 MSB=INT(E/256): LSB=E-256*MSB: RETURN
100 N=LSB: GOSUB105: I=I+1: N=MSB: GOSUB105: RETURN: REM POKE UPDATED ADDRE
SSES FOR A LOAD INSTRUCTION
105 POKE I+65536*(32767<I), N: IF N<>PEEK(I+65536*(32767<I)) THEN PRINT: PRINT"E
RROR: DATUM WASN'T STORED": CLEAR: END ELSE RETURN
110 DATA    0, 60, 58, 61,   64, 14, 1, 183,    40, 1, 12, 42
120 DATA   33, 65, 70, 35,   94, 35, 86, 42,   196, 127, 126,54
130 DATA 32, 18, 125, 129,   48, 1, 36, 111,   19, 16, 243, 34
140 DATA196, 127, 201, 42,   33, 65, 6,  0,    78, 35, 94, 35
150 DATA  86, 235, 62, 34,   237,177, 32, 3,   33, 0, 0, 195
160 DATA  154, 10
```

Listing 5. Lines in GM which are modified to use the two Machine Language subroutines.

```
32240 0A$="SAVE FRAME(S) ON TAPE":GOSUB32610:IF0DTHEN32090ELSEGOSUB32490:POKE165
26,233:POKE16527,127:FOR0E=0ATO0BSTEP0C:FOR0F=0TO4:IFUSR(SC$(0F,0E))=0THENPRINT"
FRAME"0E+1"CONTAINS A QUOTATION MARK AND WOULDN'T LOAD RIGHT":0E=0B:NEXT0E:GOTO3
2440

32260 NEXT:NEXT:PRINT"ADVANCE TAPE TO A BLANK PLACE":GOSUB32550:FOR0E=0ATO0BSTEP
0C:PRINT#-1,CM(0E),CHR$(34)SC$(0,0E):FOR0F=1TO4:PRINT#-1,CHR$(34)SC$(0F,0E):NEXT
:OUT255,4:PRINT"FRAME #"0E+1"SAVED":GOSUB32480:NEXT:GOTO32430

32630 POKE16526,196:POKE16527,127:CM(0E)=1+PEEK(16445)/8:POKE32706,0:POKE32707,6
0:0H=CM(0E)-1:FOR0F=0TO4:SC$(0F,0E)=STRING$((205-0H-(CM(0E)=2))/CM(0E),32):0A$=U
SR(SC$(0F,0E)):IFCM(0E)=20R(CM(0E)=1AND0F=3)THEN0H=1-0H
```

Calling The Program

To add Graphics Manager to one of your own programs, you need to know how to get to Graphics Manager and back. This can be done in either of two ways: You can use GOSUB to jump to Graphics Manager and RETURN to get back, or you can use GOTO to get back. If you enter Graphics Manager at line 32050, the current screen contents will be stored if there is room. If you want to go directly to the menu, enter Graphics Manager at line 32090. Line 32399 contains the statement which goes back to the graphics creation program.

General Design

Graphics Manager is written entirely in Basic. It is very compact, and, I hope, efficient. It doesn't use any READ or DATA statements, so you can use them in your own graphics creation program without problems. It uses as few variables as practical. It also uses integer variables for storing numbers, because they use less memory and arithmetic is faster using them. I tried to use variable names which you probably wouldn't use in your graphics program.

Figure 4 lists all the variables used in Graphics Manager along with a short description of the use(s) of each. The variables QA, QB, QC, . . . QI are used for a variety of short-term purposes. You may use QA-QI in the graphics creation program, and if speed is your goal, you should use those variables to minimize the time spent by Level II looking through the variable storage area.

Graphics Manager uses zero positions in arrays so memory isn't wasted. The user isn't aware of this since frame numbers go from 1 to the count of stored frames.

How The Program Works

Frames are stored in an array called SC$ (screen characters). Level II allows character strings to be a maximum of 255 characters long. A frame with single width characters uses 1024 bytes of string memory so it must be stored as five strings (which I call subframes). A frame with double width characters takes 512 bytes, so it could be stored in three strings, but I had Graphics Manager store it in five strings so the program would be simpler. Figure 5 shows how characters are distributed among the five strings in each frame.

A subframe is stored by first reserving the needed space by assigning a string of blanks to that subframe using the STRING$ instruction. Then characters are PEEKed from video memory and POKEd into string memory. This method is fast because it minimizes string space

reallocation (also known as garbage collection).

When a frame is compressed, each subframe is compressed by itself and no characters are moved from one subframe to another. The first blank is searched for, and, if there is a blank, the next nonblank character or the end of the string is searched for. If there is a substring of two or more consecutive blanks in a subframe, it is replaced by a space compression character.

Frame erasure releases the memory used by the erased frames. If you want a range of frames which includes the final frame erased, then erasure is done by assigning a null string to each subframe of each frame to be erased. If you want some beginning frames erased but not the final frames, then the final frames are moved down over the frames to be erased.

Frames are moved by copying string addresses and lengths so no physical movement of characters is done. That method of moving strings is fast and avoids an OS (out of string space) error when space is tight and a long string is assigned to a variable which held a short string. After moving down any strings which need to be moved, the indicated number of final frames are erased.

If you are still confused by the method used to erase frames, consider this example. Suppose you have seven frames stored and you want 4 and 5 erased. The range of frames you want erased doesn't include the final frame, so frames 6 and 7 must be moved down over 4 and 5 respectively. At this point 4 is the same as 6 and 5 is the same as 7. You wanted two frames erased, so two frames at the end are erased. The count of stored frames is reduced by two. The result is as if frames 4 and 5 were taken out.

Trading (swapping) two frames is done by exchanging string addresses and lengths.

Duplication of a frame is done by assigning the subframe of the frame being copied to an end location in the frame storage array. You can't copy a frame if there isn't room to hold the copy.

I didn't copy a frame by copying string addresses and lengths because I was afraid Level II would later make an actual copy of the string. I experimented a little with copying a string by setting the pointer and length of the second string to the pointer and length of the first and found Level II won't make an actual copy when it does garbage collection.

I suspect that if two strings have the same pointer and length and you use the name of one of those strings anywhere in an assignment statement, an actual copy of the original string will be made. I didn't use that method because I didn't know if it worked under all circumstances. I leave it to you to experiment with that method. If it works, you could easily change Graphics Manager to copy frames that way.

Unlike some graphics reversal subroutines I have seen, mine is fast, efficient, and doesn't disturb nongraphics characters. A blank space or graphics blank is replaced by a completely white graphics character.

I had an odd problem with line 32388 in the program. Sometimes extra characters would appear at the end. This problem seems to occur when a line of about 250 characters is listed after a line of 255 characters. I think Level II doesn't clear the output buffer after listing a very long line so the next long line gets some characters from the previous one.

When this problem occurs, remove the unwanted characters from the line in which they appear using the edit mode. Then list a short line. Next list the line which had extra characters and you

Partial sample run of Graphics Manager.

```
                GRAPHICS MANAGER BY JOHN CREW 2/2/82
A - PRINT ASCII CODES
C - COMPRESS FRAME(S)
D - DUPLICATE A FRAME
E - ERASE FRAME(S)
G - REVERSE GRAPHICS
L - LOAD FRAME(S) FROM TAPE
P - PRINT FRAME(S)
Q - QUIT
R - RETURN TO MAIN PROGRAM
S - SAVE FRAME(S) ON TAPE
T - TRADE (SWAP) TWO FRAMES
U - UNITE (COMBINE) TWO FRAMES
          6552 FREE BYTES OF FRAME STORAGE
COMMAND? L
```

```
LOAD FRAME(S) FROM TAPE (Y/N)? Y
HOW MANY FRAMES DO YOU WANT LOADED? 0
 FRAME # 0 IS THE LAST
 NUMBER IS TOO SMALL
 ONLY ROOM FOR 6 MORE FRAME(S)

HOW MANY FRAMES DO YOU WANT LOADED? 2

INSERT TAPE AND PRESS PLAY BUTTON
PRESS ANY BUT A CONTROL
KEY TO CONTINUE?

FRAME # 1 LOADED
FRAME # 2 LOADED


                      DONE
```

```
                GRAPHICS MANAGER BY JOHN CREW 2/2/82
A - PRINT ASCII CODES
C - COMPRESS FRAME(S)
D - DUPLICATE A FRAME
E - ERASE FRAME(S)
G - REVERSE GRAPHICS
L - LOAD FRAME(S) FROM TAPE
P - PRINT FRAME(S)
Q - QUIT
R - RETURN TO MAIN PROGRAM
S - SAVE FRAME(S) ON TAPE
T - TRADE (SWAP) TWO FRAMES
U - UNITE (COMBINE) TWO FRAMES
          4504 FREE BYTES OF FRAME STORAGE
COMMAND? P
```

```
PRINT FRAME(S)

STARTING WITH FRAME # (1 - 2)? 2
ENDING WITH FRAME # (1- 2)? 1
 YOU WANT REVERSE ORDER (Y/N)? Y

COPY ON MX-80 (Y/N)? N

NOW AND AFTER EACH FRAME IS PRINTED, PRESS ANY BUT A CONTROL
KEY TO CONTINUE?
```

should see only the desired characters in that line. To avoid the problem, either use short lines or don't list the program unless you are willing to go through the corrective steps mentioned before.

In a few places Graphics Manager ends a loop early because some special condition is detected. This is done by setting the loop index to its final value and then executing a NEXT for that loop. This is done in line 32130, the search of the command string, if a match is found. It is also done when reversing graphics characters if a compressed frame would expand more than there is room for.

Modifying And Extending The Program

I grew tired of waiting for a frame to be stored and found that frames with quotation marks in them wouldn't be loaded properly, so I wrote the two assembly language subroutines shown in Listing 3. The first subroutine stores a subframe after space has been reserved for it. The second scans a string for a quotation mark. If a quotation mark is found, the subroutine returns a 0; if none is found, a nonzero number is returned. The machine language string scan subroutine is much faster than scanning a string in Basic, using a loop and the MID$ function to check every character.

To put the two machine language subroutines in memory you can either use an assembler to make a system format tape and then load it, or, if you prefer Basic, you can use the Basic program in Listing 4 to put the two machine language subroutines in high memory. The program in Listing 4 sets the memory size for you and lets you put the machine language program in memory starting at any high address. It checks each byte of the machine language program to make sure it was POKEd properly. If you have a bad memory location or you ask for the machine language program to be put in nonexistent memory, you are told that a datum (part of the machine language program) wasn't properly stored.

To make Graphics Manager work with the two machine language programs, lines 32240, 32260, and 32630 should be changed to match Listing 5. Delete 32640 and 32664. If you use the Basic program to put the machine language into memory, you can remove CLEAR6553: from line 0. The program in Listing 4 ends with CLOAD so Graphics Manager will be automatically loaded, so I suggest you record the modified version of Graphics Manager right after it.

The program or program segments which appear in Listings 3 through 5 are written for a system with 16K of free memory. If you know Basic well and know a little assembly language, you could easily modify them for a different amount of free memory.

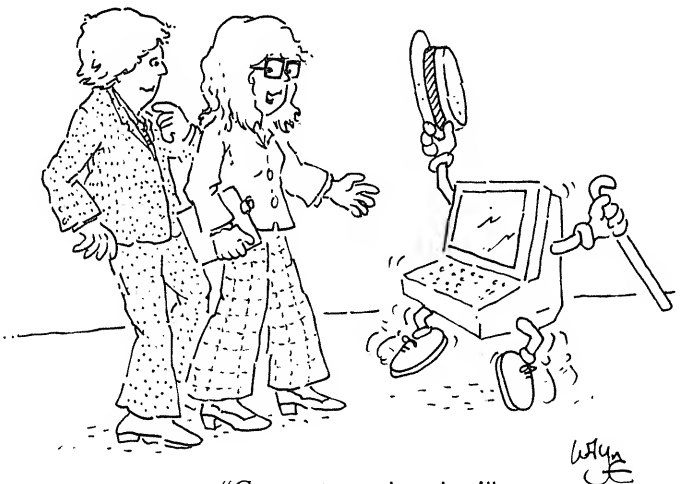
Extensions and Modifications

Some features you might want to incorporate into Graphics Manager are disk storage and retrieval of frames, and storage of more than nine frames at once (you would need to change the subframe number entry subroutines). You might also CLEAR more string space (If you have more than 16K of free RAM, I recommend CLEAR 1024*N+409 with N equal to the number of frames you want to be able to store at once); give each frame a name and search for a frame by name on tape, disk, and in memory; rewrite some of Graphics Manager in assembly language; allow storage and manipulation of partial frames; allow switching of frames from single to double width characters and vice versa; or write an assembly language program to load a frame containing a quotation mark.

Some more exotic features you might want to add are: top-bottom reflection of a frame; right-left reflection of a frame; shifting a frame right, left, up, or down; and rotating graphics about a user specified center. You might want to modify Sketch/Print and/or Vector Plotter so they could use double width characters.

Closing Notes

I hope you have found a useful program or learned something from this series. I worked extremely hard preparing it—experimenting, writing, and rewriting and I ask you to have the decency not to distribute my programs for your own profit. □



"Computer animation!"

Chapter VIII
Word Processing

Word Processing With the TRS-80

Ian Hodgson

The TRS-80 is a good choice for word processing. It is reasonable in price, has inexpensive, quality service available within a few miles of most of the U.S. population, there are at least three excellent software packages for word processing, and a wide selection of printers can be easily interfaced to the TRS-80.

Creative Computing magazine is produced by TRS-80 computers. We use an interface from Irwin Gretczko (G2 Enterprises, 255 W. 90th Street, New York, NY 10024) to connect a TRS-80 computer to an AlphaComp typesetter. The system also has an Omikron 8" drive system installed. (The interface boards, two 8" drives, CP/M operating system, MBasic, and Phoenix Word Processing system sell as a system for $1700 from Omikron (1127 Hearst Street, Berkeley, CA 94702.) We are able to prepare Electric Pencil files on our CP/M systems or Scripsit and Electric Pencil files on our TRS-80 systems and dump them directly into the typesetter. Nearly every article in the magazine has been prepared this way for the past year.

We are currently installing a micro-Composer interface ($1500 from Cove View Press, Box 637, Garberville, CA 95440) to connect another TRS-80 computer to a Compuwriter typesetter. In addition, we are installing an auto answer modem on another system to allow us to receive ASCII files over the telephone, edit them on the screen, and typeset them all by computer.

In this section, we review Scripsit, Electric Pencil, and Lazy Writer, the best known word processing programs for the Radio Shack computers. We also review two programs that add extra features to Scripsit, Scripmod and Superscript. We conclude the section with the experience of Bill Horvath, a consultant who uses a Radio Shack system for his business correspondence.—George Blank

The combination of the TRS-80 and Electric Pencil has been a very popular one for a couple of years now, and I have been using it for quite a while to produce manuscripts, and tests and reviews for my students. It is so much faster and less frustrating than a typewriter that I didn't begin to notice its shortcomings until recently.

One of Pencil's problems seems well known: that is, its tendency to drop letters when a fast typist reaches the end of a line, and the last word jumps to the beginning of the next. During the jump, the keyboard is ignored, so characters typed are lost. It can be quite frustrating to have to slow down at the end of each line. Some of its other limitations are just as annoying. For instance, if you type a line which has no spaces in it (say you want to underline a long title with dashes, for example), when you reach the 65th character the line will just disappear.

Sometimes, the program bombs in a manner reminiscent of a photon blast hitting a Klingon ship. This problem also occurs if you try to edit a Basic file (from disk) where the lines have been compressed to save time/space. Another limitation appears when you try to use Pencil to set up tables or charts that are more than 64 characters wide. In this case the lines wrap around, and it is very hard to tell where the columns are. The same limitation makes it hard to tell when a word is going to be at the end of a line, so that you may end up with a number on one line, and its units on the next, and you might get "60" and "mph" on the next. This doesn't cause much trouble in regular text, of course, but in technical writing it is awkward.

Another difficulty arises when you want to indent a block of text, or change printing format during a print run. You can't; at least, not without stopping the printing and resetting parameters. And have you noticed that if you end a paragraph by hitting Enter, just after the last word has wrapped around, the cursor returns to the beginning of the same line rather than the next?

Finally, how many times have you saved a long, long text file and then discovered the next day that you forgot to set the cursor to the beginning before saving and it is all gone?

Now none of these problems is insurmountable; in fact, the whole system is so much easier than a typewriter that I usually sing its praises loudly. But, occasionally, the thought would cross my mind that maybe, just maybe, there was something better. Now just as these thoughts were ripening, I looked through the Radio Shack catalogue, and lo and behold, there was "Word Processor Disk." The very brief blurb seemed to indicate that it would eliminate some of the problems I was having. Not only that, but it was, at $99.95, about $50 less than disk Pencil. But, would it work with lower case? I dropped by the local Radio Shack computer center and looked at the manual. Yes, if you have the offical Radio Shack lower case modification, Scripsit (as the program is called) will handle lower case. Now came the risk. It was going to cost me $100 to see if the official lower case mod was similar to the Pencil mod that so many of us have already done. The answer—YES! My money was not spent in vain. It worked perfectly.

For those of you who don't know about these lower case modifications: the character generator in the TRS-80 has both upper and lower case letters, but to save a bit of money, the folks at Radio Shack left out one memory chip (a 2102) thus leaving only 7 bits rather than 8 for screen

Ian Hodgson, 296 Malcolm Circle, Dorval, Quebec, Canada H9S 1T7.

memory. Bit 7 is used to indicate graphics mode, so only 6 are left for characters. This is not enough for a full 96-character ASCII set, so they "fake" the missing bit with a couple of gates, and all letters print in upper case. The lower case modification consists of deleting the fake bit and adding an extra memory chip, thus allowing a full character set to be displayed.

It takes a while to learn how to use Scripsit. In fact, the program comes with a set of six one-hour lessons on audio cassette, and the manual is designed around these lessons. This does make the manual a bit cumbersome for those of us who try to look things up without taking the self-teaching course, but there is a summary at the back which answers most questions.

Scripsit also comes with a set of stick-on labels for your keyboard, which make it much easier to remember what all the control keys are, and a quick reference list of all instructions. But let's find out what the program does. I will concentrate mostly on those characteristics which differ from Electric Pencil, but a more complete list of Scripsit functions is given in Figure 1.

The first thing I tried was to type at about 100 wpm and see how many letters were missed on wrap around. None! True, the last word did disappear for a second or so, but when it came back all the letters I had typed were there. Score one for Scripsit.

Then I wanted to see what horizontal scrolling was all about. In Scripsit you can define the screen width to be anything you want up to 132 characters. It is usually advantageous to set it the same as your printer width, if you are typing anything that needs special formatting. When you reach the end of the 64-character video line while typing, the entire screen scrolls to the left, and continues doing so until you reach the end of the defined line, at which point it jumps back to the right again. (Score one more—when you get to the end of a line with no spaces, Scripsit just continues on the next line.) So when you set up tables or exam questions, or want to see which word is at the end of a line, you can see just what you will get.

There is a WINDOW command which allows you to move the screen in any direction without typing characters. As a further aid in setting up columns, you can set tabs in as many positions as you want, and the tab locations are visible as dots in a solid line that run across the bottom of the screen.

The screen width command is also used for hyphenation. This command allows you to determine the best place to hyphenate words to tighten up lines. It is not a fully automatic system with memory tables, like some, but simply stops at each potential word and allows you to decide. To use the command you simply define the screen width to be the same as your printer width, define the block you want hyphenated, and enter the H command. The cursor will stop at the first appropriate word. You may hit "-" to insert a hyphen, left arrow and then "-" to insert one earlier in the word, or "enter" to leave it alone. The cursor will proceed from word to word until the block is finished. These hyphens are stored with bit 7 set so that they can later be recognized and deleted if you want to change the print width.

Another major advantage of Scripsit over Electric Pencil is that all output formatting is controlled by commands embedded in the text. Each such command line is preceded by a > sign, which must be the first character on the line. This means that you can change format as many times as you want during a printout. The format commands are listed in Figure 2. A sample text file with commands is shown in Figure 3a, while the resulting

Figure 2. Scripsit page formatting commands.

Set page length	Set left margin
Set right margin	Set top margin
Set bottom margin	Set line spacing
Justify text paragraphs	Set # lines between
Center text	Print flush right
Center vertically	Suppress printing
Enable printing	Widow suppress
Begin header on page...	Begin footer on page...
Turn on/off header	Turn on/off footer
Beginning page number	

Figure 1. Scripsit commands.

COMMAND GROUPS		OTHER COMMANDS
Cursor motions:	left	Set video width
	right	Set paragraph indent
	up	Tab set
	down	Upper case lock
	end of file	Window up
	top of display	Window down
Terminations:	end of line	Window left
	end of paragraph	Window right
	end of page	Global replace
Deletions:	character	Global delete
	word	Global search
	line	Repeat (works with all commands)
	blanks	>*comment line
	paragraph	define header block
	block	define footer block
	unmark block	define page # block
	to end of text	define hyphen block
Insertions:	character	define block
	line	end block
Exchanges:	words	remove hyphens
	paragraphs	save to disk
	blocks	save to tape
Labelling:	open block	save in ASCII form
	close block	load from disk
Information:	cursor line #?	load from tape
	length of document?	load and append
	memory left?	
Printing:	Print	
	Print to serial port	
	Print with page pauses	
	Print "invisible"	

printout is shown in Figure 3b. Important improvements over Pencil include vertical centering (good for writing letters), paragraph spacing (which can differ from line spacing), flush right mode (also called ragged left), automatic horizontal centering (good for title pages) and "window suppress." This last command forces the printer to a new page if the last line on the previous page would start a new paragraph.

Putting > * at the beginning of a line allows you to insert comments (which will not be printed) in the text.

Scripsit allows more flexible page titling than Pencil. In Pencil, you may use one line (less ten spaces for the page number) for a title which will appear at the top of each page, followed by the page number at the top right corner. Scripsit lets you have multiline (up to 16) headers and/or footers, and the page number can be anywhere you want at the top or bottom of each page. You can also format the page number, so you could have "PAGE 3," "page 3," "p. 3," or even "This is page 3" if you wish. The headers and/or footers may be on every page, on even pages, or on odd pages, and you can have a different header on even and odd pages, so that you could have, say, the chapter number on the right-hand pages and the topic on the left-hand ones, with the page numbers on every page. Footnotes can be done in Scripsit, but not easily, as it does not have specific provision for them. Headers and footers can be turned on and off or changed at will anywhere within the text.

```
>*COMMENT LINES CAN BE INCLUDED IN THE FILE, and they_
>*don't print in the final copy._
>VC=N_
SCRIPSIT allows you to change print format by imbedding
commands in the text. The default format has a left margin
of 12, right margin of 72, and is justified and vertically
centered. This paragraph is printed in the default format,
although it is not vertically centered._
_
>C=N J=Y LM=25 RM=59_
Long quotes are often inset from the body of a report as
shown in this example. Many text editors make it possible
to do this easily._
_
>J=N LM=12 RM=72_
Another format is "ragged right" (or flush left). Justifi-
cation is turned off, and, if desired, the lines can be
evened out by hyphenating words. This paragraph is printed
in the ragged right mode._
_
>FR=Y_
     Flush right (sometimes called ragged left) is another
print mode available on SCRIPSIT. It is most often used for
page headers or sometimes for labelling diagrams. Text
printed this way looks a bit unusual, but could be used for
decorative purposes._
_
>FR=N C=Y_
Centered text is useful for page numbering,_
title pages, and similar uses. It is not justified.
Justified centered text can be_
accomplished by setting appropriate left and right margins.
Notice that all_
of these changes, and many more, can be set during printing
by means of imbedded commands._
```

Figure 3a.

Entering text in Scripsit is very much the same as in Pencil, but whereas Pencil zeroes all of RAM for a buffer before you start, Scripsit does not. In fact, Scripsit starts you off with 60 blanks (spaces) in the text buffer and you type over them. When you use them up it gives you 60 more, and so on (the letter you are typing disappears for a tiny fraction of a second while this happens, but you don't lose it). You cannot advance the cursor past the buffer area. This leads to one slight inconvenience. If you have a terminated line, such as one ending with ENTER at the end of a paragraph, you cannot move the cursor past the end of that line unless you use the "insert line" command to put 60 blanks there. In fact, there is no multiple-insert mode as there is in Pencil, so to insert a word you press "insert line," then type in the word, then hit CLEAR to eliminate the excess blanks. It sounds a bit inconvenient, but you rapidly get used to it, and the insert mode in Pencil is not much better, especially if you are inserting at the beginning of a long file, in which case the entire file moves after each letter and insertions are painfully slow. In both editors, after you finish inserting the first line, more space automatically opens up one line at a time.

In Pencil you can define a block of text and then delete it or move it to another location. This allows you to rearrange the order of what you are writing and makes short work of editing. Scripsit allows you to do this with multiple blocks, as each is identified by means of a letter. There is also an "exchange block" command which is very useful.

Scripsit also allows you to exchange paragraphs or words. The procedure is the same in each case: simply place the cursor anywhere in the second paragraph (or word) and hit "exchange paragraph" (or "exchange word") and the job is done. This allows easy tidying of sentences that might read better if the words were reordered.

The Electric Pencil supports two printer drivers. If a line printer is connected to the parallel port, it will be driven more or less normally. I say "more or less" because Pencil supplies line feeds after carriage returns and the Radio Shack line printers will double space in this mode, which is fine if you want double spacing, but Pencil will then give you pages twice as long as normal too. There is a special command (SX) to fix this, but it allows single-spacing only, and requires reloading of Pencil to change once it is set. (Pencil seems really to be set up to use the TRS-232 interface, which does software timing and outputs 300 baud ASCII via the cassette port. It will automatically switch to this mode if there is no line printer.

There are three print commands in Scripsit. "P" will send output to the line printer, in completely Centronics (i.e. Radio Shack) compatible form. "P,S" will send the output to the RS-232 port on the expansion interface. Adding the ",I" suffix to either of these will print the entire file, including formatting commands and comments. Unfortunately, Scripsit assumes that your serial printer supplies its own line feeds. Mine doesn't, and using double-line spacing to overcome the problem gives you pages half the normal length; just the reverse of Pencil's problem. Score -1 for Scripsit.

On the plus side, though, if you want to use single sheets of paper (for example, letterhead) in your printer, adding a ",P" suffix to any of the print commands will cause output to stop at the end of each page while you put in a new sheet. One nice feature is that Scripsit always scans the "clear" key, which is used to abort commands in progress. This key is even

scanned during printing, while the program waits for the line printer "busy" line to clear, so you can regain control even if your printer stops, runs out of paper or is unselected. Normally this would require rebooting and would imply loss of your file. (This is what happens with Pencil.)

When you load a file the first time from disk, Scripsit remembers its filespec and drive number until it is changed. Saving the modifed file is just a matter of typing "break" then "S". Scripsit will even remember the drive number on which it found the file if you didn't specify it, and append that number to the filespec so that when you save the file again it will be saved on the diskette from which it originally came. The cursor does not have to be moved to the beginning of the file! It quickly becomes a habit to type "break S" every few minutes to back up the file onto disk in case of power failure or what have you. Similarly, if you have botched the file since last loading it, "break L" will reload the original. The current file name can be ascertained with the "break ?N" command. Commands for saving to and loading from cassette are also included.

Although there are ways to do it, Pencil is hard to use with Basic files. If you try to load a Basic program or data file into Pencil it doesn't load, and DOS ERROR 22 appears. (The Pencil fix included with the VTOS operating system overcomes this problem, and also allows Pencil to operate with NewDOS, which it doesn't normally do either). The problem of lines longer than 64 characters without a space still remains even if you do get the file loaded, though. On the other hand, Scripsit operates perfectly with Basic program and data files (except that I haven't found a way of entering an up-arrow for exponentiation) or any other ASCII files. There is even a special ASCII save command in Scripsit so that Basic-compatible files will be produced. This is necessary because all command characters (such as paragraph ends, etc.) are normally stored in Scripsit files with bit 7 set, which would cause confusion since Basic would be looking for hex OD to terminate lines and would see hex 8D instead. In the ASCII mode, bit 7 is not set. When you load an ASCII file into Scripsit it will determine the control characters from the context, and set bit 7. (Scripsit did not originally work properly with NewDOS, failing to load the last record of a file, but patches to Scripsit are now supplied with NewDOS 80.)

Incidentally, Scripsit works fine with Pencil files, although you will have to add the formatting commands. And the VTOS-fixed Pencil works on Scripsit files too, within its limitations.

Thoughts on Scripsit

The following is part of a letter sent to John I. Snodgrass, Jr., manager of Radio Shack Computer Services, by Peter J. Brennan, Worldtech, P.O. Box 834, Gracie Station, New York, NY 10028.

I find that I can type a great deal faster with Scripsit than Pencil. It does not drop characters at the beginning of the line and does not have any keybounce whatever, which in Pencil is a big problem for two-fingered typists like myself. The hypenation feature is useful though cumbersome. Other features that undoubtedly make the program very versatile also make it far more difficult to learn to use than Pencil. Formatting is driving me up the wall. Indeed, I have made some standard formats and saved them on disk. When I want to type, I load in the format, change the date on the header and start typing.

There are some problems and also some features lacking that I would like to see added:

1) ?M gives me 32,608 characters free on start-up of my 48K two-drive system. The manual says the maximum figure, which I think I have, should be 36,909. That's quite a difference and one that matters to me since my standard manuscript will not fit entirely in memory with Scripsit but fits fine in Pencil. I wish I understood the difference.

2) I have an Anderson-Jacobson AJ-841 parallel printer, which, as you know, has Z-80-based electronics and an IBM Selectric front end. It runs beautifully with the TRS-80, or has. It doesn't run very well with Scripsit. Apparently it drops the first line feed after carriage-return-with-line-feed. LS=2 gives single spacing but the computer counts double so the page ends in the middle of the sheet. To make one blank line, one must make two line delimiters. LS=3 gives double spacing but also a miscount that ends the page far up the sheet. With a long manuscript that I had to get out in acceptable form, I was tearing my hair out. I finally set page length at 35 and set the printer mechanically to double space. It worked but is not an acceptable long-term solution. Single-space works fine.

3) I can build and save a much larger text than I can load back into the machine from disk. It saves fine, but then "L" gives me "NO MORE ROOM." Fortunately, I have "Superzap" and was able to identify and zero out the last disk sector of the text and then load the truncated version. What does one do if one does not have a way to get into the disk? The manual should emphasize this problem. Perhaps the program could be modified to flash a warning as the text in memory approaches a quantity that could not be safely saved and reloaded.

4) The instructional tapes with the manual are very helpful for getting one started. But the manual alone is not very good. It needs a lot more descriptive text and more examples of different types of formatting, problems that can arise and the like.

5) DOS allows one to work directly from the disk. You can print and read actively off the disk. I had hoped that some of this capability would be carried into Scripsit so that one would not be confined to working only with what is in memory, as is also the case with Electric Pencil and the main criticism of the program. I want to be able to take addresses and file copy from disk and mix and match it with form text on another disk or in memory to print out customized documents. I feel the TRS-80 disk system has that capability. I am disappointed that Scripsit has not capitalized on it. Are there plans to do so in some later, updated version?

Despite my carping, I am on the whole satisfied with Scripsit. I feel, though, that such advantages as it has over Pencil come at the cost of far greater complexity. I would like to see all the formatting commands and instructions put in a separate subprogram instead of as part of the text, for example, displayable on command on that bottom line.

I am also satisfied with the TRS-80, more than satisfied. — *Peter J. Brennan* □

Well, I seem to be running the risk of writing an instruction manual here, so I guess I'll call it quits. Which one should you use? If you have both you may find Pencil fine for short jobs—say, typing a quick letter—and, of course, it lets you use the TRS-232 interface. It also allows you a bit more freedom in moving the cursor around to insert text past the end of a line. But, for those of you with no editor yet, I can only see one reason for buying Electric Pencil: it is a smaller program (6K or so compared to 10K) and allows you more text space (4K is about one single-spaced typewritten page). This is no problem at all on a 48K system, but is worth considering if you only have 32K. Pencil has several bugs, doesn't do half of what Scripsit does, and costs 50% more. With Radio Shack software often rumored to be, it comes as a pleasant surprise to find a dynamite text editor for a modest price right in their catalogue. □

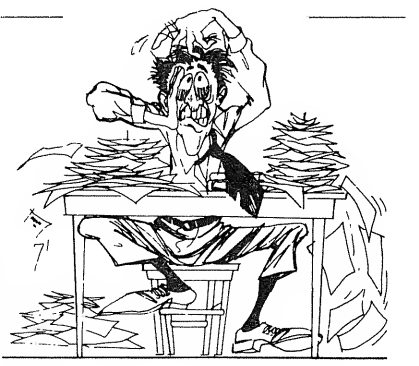
Editor's Note: Creative TRS-80 readers please note that the Electric Pencil has been replaced with the Electric Pencil 2.0. For more information on this latest Pencil, see the review later in this chapter.

```
SCRIPSIT allows you to change print format by imbedding
commands in the text. The default format has a left margin
of 12, right margin of 72, and is justified and vertically
centered. This paragraph is printed in the default format,
although it is not vertically centered.

          Long quotes are often inset from
          the body of a report as shown in
          this example. Many text editors
          make it possible to do this
          easily.

Another format is "ragged right" (or flush left). Justifi-
cation is turned off, and, if desired, the lines can be
evened out by hyphenating words. This paragraph is printed
in the ragged right mode.

      Flush right (sometimes called ragged left) is another
print mode available on SCRIPSIT. It is most often used for
     page headers or sometimes for labelling diagrams. Text
printed this way looks a bit unusual, but could be used for
                                        decorative purposes.

       Centered text is useful for page numbering,
   title pages, and similar uses. It is not justified.
            Justified centered text can be
accomplished by setting appropriate left and right margins.
                  Notice that all
of these changes, and many more, can be set during printing
            by means of imbedded commands.
```

Figure 3b.

A Complete Word Processing System

Getting It Together on the TRS-80



Bill Horvath

Radio Shack is finally beginning to put all of the pieces together! I discovered this recently when converting my old faithful 4K Level I into a 16K Level II machine outfitted for word processing.

My Level I had served me well in the 15 months since I originally purchased it to "give it a try." I considered it paid for not long into its life when it did wonders (with the help of Radio Shack's Math I program and some efforts of my own) for my seven-year old son who was having considerable difficulty comprehending fundamental math relationships.

Bill Horvath, 2459 Haymaker Rd., Monroeville, PA 15146.

The Decision to Upgrade

My work as a consultant, with increasing demands for reports and documents, led to my decision to upgrade. As a start, I pored through the most recent Radio Shack flyers and catalogs as well as the current personal computer magazines.

I developed my shopping list:

1 16K-Level II conversion (without keypad)
1 Lower case mod
1 Scripsit word processor (tape)
1 Line Printer II with cable

Assorted items such as tapes, ribbon, and extra paper.

I would also ask for the cassette-tape-loading modification to the board, the KBFIX tape to eliminate keybounce, and the mod to my CTR-80 cassette recorder (manufactured before March of 1979) to eliminate the possibility of glitches on the tape if the recorder was stopped for some reason during the save or load process.

My first visit was to the local Radio

Shack outlet. No, I was told, you must purchase the keypad with the Level II modification! Since my exclusive use of the computer would be in wordcrafting, the added baggage of the keypad on my board would be next to useless.

Since the regional Radio Shack Computer Center was not too many miles distant, I decided to visit before placing an order. "Yes," I was told, "we will install all of the mods without the keypad." (Savings: $79 for the pad plus $15 for installation.)

A few days later, I dropped off my 4K board and cassette deck. It was a Wednesday morning. The following afternoon I received a call that all was ready and could be picked up the following morning.

The Store Visit

Bright and early Friday morning, I arrived at the Computer Center to pick up my modified equipment. Before I made a decision on purchasing the Line Printer II, would I look at the new Line Printer IV which had just arrived from the warehouse? Indeed, I would. The demonstration proved that the printer had surprising capabilities. Yes, indeed, I would be very interested in the new printer, since I had saved nearly $100 by eliminating the keypad installation! It turned out to be a very good decision, as I will outline later.

What about my KBFIX tape, I asked? No longer needed, was the reply. Everything was handled with the modification.

OK, I replied, somewhat skeptically, checking to be certain all my manuals and conversion tapes were present before I left the store.

The Home Experience

Back home, I plugged all the components together and powered up the computer. An immediate shocking discovery! Memory read 15770 for 16K rather then the customary 15772. Was there a defective chip? Panic set in! A quick call to the Radio Shack Computer Center provided an answer and alleviated my anxieties.

In its latest conversion kits, Radio Shack is using a new ROM that reduces free memory space by two bytes. But this is certainly a welcome tradeoff for what the new ROM accomplishes, including an end to frustrating keybounce and cassette loading problems.

With the latest ROM you get MEM SIZE? on power up. Assuming you do not wish to protect any memory and push "Enter," the computer comes back with "R/S L2 BASIC" and the familiar "READY ." From here, you can type in "SYSTEM" and load machine language or go directly to Basic.

The new ROM also provides a number of other features such as the outputting of control characters, performing multiple "Print @s" using a single print statement, and the ability to specify file name using CLOAD under disk basic.

That was the good news, now the bad. If you have a second tape drive with Expansion Interface, you can now only CLOAD or CLOAD? with tape drive number 1. At least ten older Radio Shack programs—including MicroChess, Algebra I, and Editor/Assembler—may not load properly with the new ROM because of timing of the material on the tapes. Further, the new ROM is not directly interchangeable with the old—requiring you to shell out $150 for a new kit if you are interested.

Nevertheless, my reborn Level I hasn't changed its personality with the new mod. The occasional keybounce I once encountered (remedied with a dab of alcohol on a cotton swab, is now completely absent. Best of all, the cassette recorder thinks it is still feeding the Level I baud rate. I have tried a variety of tapes from my own to Radio Shack's to other software producers'. No problems! With the new mod, it's a case of set it and forget it.

Living with Lower Case

Radio Shack's lower case modification requires entering a machine language driver program from tape or disk before writing or running your own program utilizing lower case. Once you've loaded, you are ready to operate in normal Basic.

The modification manual warns that non-Radio Shack software may not load properly with the mod installed, and lists two of their own programs (Accounts Receivable and Level II Cassette Payroll) that won't work either. No doubt the problem is a memory-space clash—the mod uses the area of 32208 to 32767.

However, I have loaded a number of programs from various sources and have yet to note a problem. My advice is simple—make sure that the software you purchase won't run into the memory space area reserved if you have the upper/lower case mod installed.

Adventures with Scripsit

Scripsit, the Radio Shack word processing system, has the lower case driver included. Lower case is invaluable if you intend to do any serious word processing and have a printer capable of printing lower case. One of the best features are the "below the line" descenders which make the text on the video display very easy to read.

The tape version takes about four minutes to load. I find no difficulty with this, as it allows me time to organize my notes—and my thoughts—before beginning work. Once the machine language tape is loaded, you bring up the lower case by pushing the control (@) key and shift key at the same time. I have found this to be a somewhat tricky proposition sometimes requiring several tries.

Before beginning a letter or document, you enter a format line providing such instructions as lines and paragraph spacing, width of text, and whether or not you want the copy justified. From that point, you can begin writing text.

Although I've worked for years with manual and electric typewriters, this was my first experience with electronic word processing. It was a pleasant surprise! I am a fairly speedy typist, and the keyboard (the older TRS-80 version) felt very good to the touch. No matter how fast I went, the computer and screen kept up. Nowhere was there any sign of keybounce. Once in a while I would glance at the screen and would see the computer pause a fraction of a second before displaying a letter. I figured it was trying to discover what my flying fingers were up to on the keys—but it was right every time!

It took me less than an hour to learn the basic workings of Scripsit. At that point, I was able to turn out a decent-looking letter addressed to a friend. I realize that Scripsit holds a great deal more potential and have begun working with its manual and recorded lessons to gain as much skill as I can in using all the program features.

With my 16K machine, I am able to run roughly three double-spaced pages of text before running out of memory space. So, with my longer documents, I simply dump the text to tape, give a command to erase text from beginning to end, and begin again without having to reload Scripsit.

The only disadvantage here is the inability to move paragraphs any significant distance within the document.

There are a few frustrations. Although you can request the computer to give you a reading on remaining text memory space, it is a bother. You may be down to the last few words of a paragraph when the computer signals "NO MORE ROOM." I do wish for a signal that might tell me "FIVE MORE LINES" to help me to conserve space in wrapping up the material.

Another oddity is the occasional sudden dropout of lower case—usually while I am in an editing or special command mode. At this point, no matter what efforts I put forth, I am unable to bring lower case back short of reloading the program. I suspect I am accidentally hitting a wrong

key combination when this occurs. I hope Scripsit developers will provide a more "fail-safe" protection for lower case in future versions.

Line Printer IV

I have saved the best for last. Radio Shack's Line Printer IV (a Centronics 737 in disguise) is a truly versatile machine and a perfect companion for the TRS-80 equipped for word processing.

When I went shopping for a printer to go with my upgraded system, I looked at several in the $800 to $1,000 bracket. Among them were the Anadex (of the infamous purple ribbon), the Paper Tiger and the Okidata. The latter two incorporated graphics capabilites. They were all well-designed machines, but I decided to stick with tried-and-true Centronics and its Radio Shack version, Line Printer II.

What concerned me most about Printer II was its inability to produce true lower case descenders. Especially irritating were the tiny "Q" and the miniature "G"—simply upper case letters reduced in size. I realized that it might make proofreading a nightmare.

Nevertheless, I made it my first choice based on the Centronics' reputation for reliability and service. But as luck would have it, the updated Line Printer IV arrived at the Radio Shack Computer Center at the same time as I did.

The Line Printer IV is the same size and shape as its predecessor, but there is a great deal more on the inside, and its cost is the same as Printer II's was only a short time ago.

First, there is a new dot-matrix print head that can form up to nine vertical and six to eighteen horizontal dot placements, in contrast to Printer II's 7 x 7 head.

The result: three basic type styles, each with an elongated version, plus the capabilities of underlining, bold or overprint characters, superscript and subscript. The three basic styles are normal (ten characters per inch), condensed normal (16.7 cpi), and proportional. Each of these styles may be printed in an elongated version.

Although many users will prefer the normal face for most applications, the proportional text is a very clean and easy-to-read face, difficult to distinquish from the faces on some of the finest electric typewriters.

When using Basic, it is simple to shift from one print mode to another as your text needs change. As an example, the command "LPRINT CHR$ (27); CHR$ (14)" will instruct the printer to select the proportional characters rather than the normal.

Scripsit, on the other hand, will only permit selection of the character format once. When using Line Printer IV with Scripsit, any special function control command must be entered in Basic before typing "SYSTEM" and loading the Scripsit tape. The only way to change is to turn off the computer and reload the word processor—an effort I would make only in the most dire circumstance. I hope that later versions of the current Scripsit (Version 1.0) will allow the user to take full advantage of Line Printer IV's capabilities.

IN ANY WATER SOLUTION $[H_3O^+]X[OH^-]$ ALWAYS EQUALS 10^{-14}

Bi-Directional Printing allows the use of superscript and subscript.

Line Printer IV Character Styles.

PROPORTIONAL NORMAL

```
 !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~
```

PROPORTIONAL ELONGATED

```
 !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVW
`abcdefghijklmnopqrstuvwxyz{|}~
```

10 CPI NORMAL

```
 !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~
```

10 CPI ELONGATED

```
 !"#$%&'()*+,-./0123456
@ABCDEFGHIJKLMNOPQRSTUV
`abcdefghijklmnopqrstuv
```

CONDENSED (16.7 CPI) NORMAL

```
 !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~
```

CONDENSED (16.7 CPI) ELONGATED

```
 !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
`abcdefghijklmnopqrstuvwxyz{|}~
```

The printer has three controls located at the bottom of the front of the case. At the right is the on-off power switch. On the left-hand side are the ON LINE/LOCAL and PAPER FWD/REV controls.

When powering up or down, the left control should always be in the "local" mode to avoid any stray signal input from the computer. The switch must also be in this position if you wish to move the paper manually using the paper feed control. The two controls, used in conjuction, make paper handling a breeze. Also, if you should have a paper jam, throwing the right lever into "local" allows you to fix things without losing characters already in the printer's buffer.

Printer IV uses the same Möbius-loop ribbon-tray system as its forerunner. I do suspect that, despite occasional loading problems, you can use a great deal more ribbon this way in contrast with the traditional reel-to-reel method.

There is still the convenience of three paper feeds—roll paper, fan-fold, or single sheets. Alas, some improvements have been neglected in the transition of Printer II to Printer IV. The power-on LED indicator light is hidden deep in the innards of the machine, visible only through the upper paper slot (if you look carefully and in the right direction). On the first day the printer was in residence, I powered down everything else but missed the printer, which sat with power on for two hours. It could have been days or weeks! Please, Centronics folks, place the LED so it shows on the outside of the case.

The print-head guide also remains a problem. Single sheets and roll paper must be positioned very carefully when first loading. Otherwise, there is a tendency for the print head guide to snag the edge of the paper as its begins its rightward travel. Anything from small nicks in the sheet to torn sheets and snagged print heads can result.

The print head itself contains a four-position lever accessible by opening the top cover. The load position is used to retract the head while inserting paper. The user can select the other positions—1, 2 or 3—depending on thickness of paper, number of carbons (maximum of about three), or heaviness of impact.

The plastic "tear-off" strip, part of the top cover, leaves something to be desired when separating roll paper sheets. To get a clean cut, you must act gingerly, using a finger to keep a straight cutting edge. I've tried a quick-tear method, but this usually results in moving the paper from its straight-and-narrow path—which then requires paper repositioning.

I consider these to be very minor faults. After using Line Printer IV, I consider it to be the finest of its class if your main objective is word or number output. As of now, its features, with respect to paper variations, type styles and printing modes are unequalled.

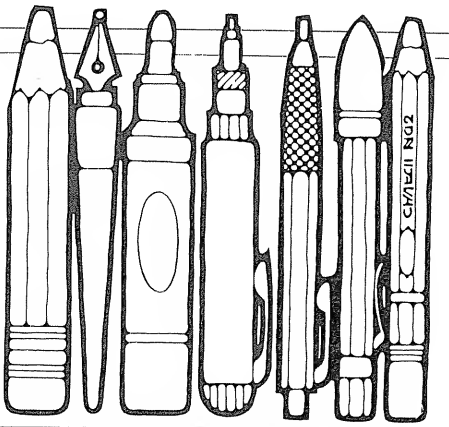
For years I have worked at electric typewriters whose motors and gears constantly whirred. As I pressed each key, there would be a mechanical clank of machinery in motion. Now, there is only the sound of the keyboard as I type. The printer sits quietly until I am ready to use it. And when I am, it simply purrs away quickly, quietly and efficiently—and the job is done.

Satisfied with my upgrade to Level II, Line Printer IV, and word processing? You bet! □

Editor's note: Creative TRS-80 readers please note that the Line Printer IV is no longer available. It has been replaced by the Line Printer VIII which has all the features of LPIV plus graphics.

Scripmod

Improving on Scripsit



Stephen Kimmel

I discovered the deficiencies of Scripsit about halfway through chapter 15 of my third novel *Lord of High Places*. The alien hero, a green, semi-intelligent, dinosaur-like creature, is trying to bluff his way out of certain disaster. Standing on his rear four legs he screams "Rhapaka is nothing compared to the Star Lord!" I decided that bit of deathless prose needed added emphasis and should be underlined.

Although I had dutifully listened to the six-hour course that came with Scripsit, I couldn't remember how to do that. After three more hours of searching through the manual, the reason became clear. I couldn't remember how to underline because Scripsit doesn't allow underlining.

"Ah come on!" I bellowed. "You sold me a $100 program that won't make the $2000 printer you sold me underline? Or backspace. Or print a copyright character. Or. Or. Or."

Filling the vacuum

Two programs have apeared to fill this vacuum. Superscript is reviewed elsewhere in this issue. Less well-known (read "not as heavily advertised") is Scripmod, $39.95

Stephen Kimmel, 4756 S. Irvington Place, Tulsa, OK 74135.

from MG Products, P.O. Box 7544, Tulsa, OK 74105). It is a single program that modifies your Scripsit each time you use it. While the Scripmod system may seem awkward, I have no difficulty with it. I simply put the program as an Auto statement and it takes perhaps two extra seconds. I can't tell the difference.

Scripsit's deficiencies come in two basic classes: the missing directory from within Scripsit and the lack of ability to send the various printer control codes. Scripsit doesn't bother to send anything but the minimal set of control codes to the printer. If you are going to write a program to modify Scripsit, you have to solve the problem of how to tell the printer where to start underlining and where to stop all the other things that printers do in their heart of darkness.

Enter Scripmod

In Scripmod you type <Control> <T> at the point you want to begin or end a special function and a British pound sign (a fancy L) appears. (You can't get that character from within Scripsit under normal circumstances.) When the computer encounters the sign it goes to a special control character instruction that you have built into the text to find out what the pound sign means this time. For example, the instruction

>CC=O,N,O,N will tell the program to send control code 14 (O is the fourteenth letter) to the printer the first time it encounters the special instruction, 13 the second, 14 the third and 13 the fourth. On my particular printer, a Daisy Wheel II, a 14 control code means to start underlining and a 13 means to stop. So,

```
CC=O,N
LScripmodL
```

will print the word "Scripmod" and underline just that word.

Scripmod can send any control code that your printer is capable of receiving since it isn't limited to just those built into the program. That's also why there are no extra drivers; the one program will support anything that can be supported by a TRS-80.

O and N represent underline-related instructions on the Daisy Wheel II and the Line Printer IV but they may well be subscripting on a Qume or a Diablo. With Scripmod you have to figure out which functions are represented by what letters. It's a learning process and the authors encourage you to experiment. Perhaps your printer has control codes the manufacturer neglected to tell you about.

As for the other deficiency; to display a directory from within Scripsit, use a <Control> <M>. The program then asks which drive you want and gives you the appropriate directory.

Future Modifications

There are still more features that would be useful for a word processing package. My Daisy Wheel printer is capable of printing twice as many characters as Scripsit will show. How about somebody writing a modification to do that? I don't know about anybody else but I frequently do articles too long for even a 48K machine. Scripmod's next version will enable one section of text to load the next one, thus permitting an infinitely long document. I doubt that I'll need one that long, but I could use one capable of handling 300 pages of text. I suspect that they'll use a final control command at the end of a section to load the next section.

Let's go wild. What else would be nice to have? TRSDOS has a date function. Why not permit Scripsit access to that? Have the letter date itself. Or how about letting Scripsit access external mailing lists? Has anyone done these thing yet? □

If Only Scripsit Could . . .

With SuperScript, It Can!

Carl Iseli

In general, the Scripsit word processing system for the TRS-80 Model I from Radio Shack has received rave reviews. It is just about the most versatile such program available for anything near its price. If you plan to use a word processor professionally, however, "just about" doesn't quite make it. Scripsit comes up lacking in several areas—some of them major, others relatively trivial. But that was before Acorn Software Products, Inc., 634 North Caroline Ave., SE, Washington, D.C. 20003, introduced a dandy modification called *SuperScript* by Richard Wilkes.

Carl Iseli, 2108 Kingshouse Rd., Silver Spring, MD 20904.

Underlining, superscripting and subscripting, special characters and boldface type are some of the more obvious abilities *SuperScript* adds to your word processing. Others, like the ability to prodce either "slashed" zeroes(Ø) or the regular variety (0), forced spacing, keyboard text insertion during printout and type pitch selectability, can come in handy more often than you might expect. Finally, *SuperScript* has added a couple of features that I can credit with having saved what's left of my sanity!

Until you've experienced the thrill of trying to save a twenty-page manuscript on disk, only to have the "DISK AREA FULL" message pop onto the bottom of the screen, you won't fully appreciate the ability of *SuperScript* to read a disk directory and kill files. Don't be smug, it *will* happen to you—probably at 3 a.m.—and your spare diskette box *will* be empty. After modifying Scripsit with *SuperScript*, you will be able either to find a disk with enough free space or kill some unnecessary files. And all the while your text remains safe and sound, waiting to be dumped whenever you are ready.

SuperScript is not a word processing system by itself, nor does the disk supplied

even contain a complete Scripsit program. It combines with and modifies your copy of Scripsit, giving it all the capabilities you've wished for. The *SuperScript* disk contains the basic Scripsit modification, driver routines for several popular printers and a file transfer utility, with a versatile, general duty lower-case driver thrown in as a free extra. You transfer your Scripsit/LC program to this disk—using the file transfer option if you have only one disk drive. Then specify which print driver you will use, and the program will combine Scripsit, the modification and the print driver into one program, called "SCRIPT/CMD." SCRIPT can now be copied onto any disk using the COPY command if you have two or more drives, or the file transfer utility on *SuperScript* for single drive owners.

In order to use *SuperScript*, you must have the disk version of Scripsit and your TRS-80 must have a lower-case modification installed. The reason for the first restriction is that the tape version of Scripsit uses different memory locations than does the disk version. The lowercase requirement shouldn't bother too many people; after all, a word processer without lowercase is like Perrier without bubbles! To utilize the printing capabilities fully, you need one of the daisy-wheel, thimble type, or other letter-quality printers: Radio Shack's Daisy-II and Lineprinter IV (and its clone, the Centronics 737), Diablo (parallel or serial), NEC Spinwriter (5510 or 5530, Qume, Sanders, and Anderson-Jacobson. *SuperScript* will support other printers with underlining and slashed zeros only, provided the printer is capable of backspacing.

One of my questions in evaluating any modification is: Does it accomplish its purpose without introducing bugs in the target program? In this, *SuperScript* really shines. The modification is almost completely invisible to the Scripsit user; you can use Scripsit exactly as described in the Radio Shack documentation and ignore the added capabilities if you like. The only exceptions to this are in the copyright notice that appears at the beginning of the program, and in the "L" (load file from disk) command. Normally, you needn't use a disk filename when you specify the "L" command, assuming that you have previously loaded or saved a file using its name. Since "L" is also used to find the length of the document in memory, it is conceivable that you might accidentally load a file—thus destroying whatever is in memory—while simply trying to find the length of the current file. Under *SuperScript*, you *must* specify a file name each time you load it.

SOFTWARE PROFILE

Name: SuperScript
Type: Word Processing Utility
System: TRS-80 Model I, III
Format: Disk
Language: Machine Language
Summary: Enhancement to Scripsit
Price: $50.00
Manufacturer:
Acorn Software Products. Inc.
634 N. Caroline Ave., S.E.
Washington, DC 20003

Printing Options

Using the new features is simplicity itself. The print modifying commands are embedded into the text using the "@" key as a "control" key. For example, to start underlining a section of text, you press the "U" key while holding down "@" (control). At the point where you wish the underlining to end, just press "control-U" again. Similar control codes are used for other print operations (Table 1). Each of the new control codes has a unique graphic block associated with it that becomes embedded in the text you see on your screen. Even these blocks have been chosen to indicate, as well as possible, the functions they perform. The underlining indicator is a small square below the print line. Superscripting and subscripting indicators are tiny rectangles above and below

Table 1. SuperScript Functions.

CONTROL CHARACTERS

Letters enclosed in brackets indicate that they are typed while holding down the "control" (@) key.

Control	Function	Remarks
[U]	Underline	Used to begin & end function
[Y] [U]	Boldface	Used to begin & end function
[T]	Superscript	Used to begin & end function
[B]	Subscript	Used to begin & end function
[Y] ∅	Slashed Zero	[Y] must precede each zero to be slashed
[N]	Required Space	See text
[O]	Left Bracket [	Appears as up-arrow
[P]	Right Bracket]	Appears as left-arrow
[K]	Left Braces {	
[L]	Right Braces }	
[I]	Caret ^	Appears as right-arrow
[Y] [B]	Set 10-Pitch	
[Y] [T]	Set 12-Pitch	
[Y] [Y]	Insert Text from Keyboard During Printout	

COMMAND FUNCTIONS

These are added functions that can be used by pressing the BREAK key, followed by the command.

Command	Function
?D	Displays the disk directory and amount of free disk space. Text is restored by pressing ENTER or space bar.
K filename	Kills the named file from the disk and frees its space. "filename" can be any valid DOS name: TVPROD/SCR:0, etc.

the line, respectively. The indicator for boldface even resembles a "b" in block graphics.

Active text insertion allows you to put in text from the keyboard while you are printing a Scripsit file. Thus you can insert names or other information in a form letter, etc. This can only be done in unjustified text, and is accomplished by typing "control-Y" twice. When these characters are encountered during a printout, the printer pauses and the bottom line of the screen (the "Command Line") awaits keyboard entry of the text to be inserted; after typing the desired insertion, pressing ENTER causes it to be printed and the printout to resume. Neat!

The purpose of the "required space" command is a little less apparent. When printing justified text, Scripsit randomly inserts spaces as necessary to make the lines come out even. Occasionally, this puts two or more spaces between words that really need to "go together," like "Atlantic City" (which looks strange if it is printed as "Atlantic City"). By typing "Atlantic <control-N> City," SuperScript sees one 13-letter word (where one of the letters happens to be a space) rather than an eight-letter word, a space, and a four letter word. It will always print with only one space. Another place where the required space comes in handy is when you want to be sure that a group of words appears on the same line. Since the group is seen by the program as one big word, it will print on the same line no matter how you format your text.

Other print control codes allow the printing of brackets ([ ]), braces ({ }), and carets (^). With some printers, *SuperScript* lets you toggle between ten characters per inch (10-pitch) and twelve characters per inch (12-pitch). Though you would normally set your type pitch at the beginning of the text, this, combined with the printout pause capability, lets you do some pretty fancy printing: You could start printing in 10-pitch, then pause the printer, switch type-wheels, change to 12-pitch, and print indented material with a completely different look than the main text (see Figure 1).

Directory and Kill Commands

SuperScript gives you many clever printing options, but the special commands available during text entry/editing will be used even more often, perhaps. These are the Directory and Kill commands, and add immense ease-of-use and power to Scripsit.

You can see the disk directory at any time by simply pressing the BREAK key (to put you in command mode) and typing "?D." The directory will replace the text on screen and will show all files with regular and "invisible" attributes (but not "system" files), plus an indication of the amount of disk space available in grans. Great, you say, but what happens to the text that was replaced by the directory? Press either the space bar or the ENTER key and the directory disappears, to be replaced by your text—even the cursor is just where you left it.

Killing files without leaving Scripsit is even simpler than killing them from DOS. Just enter the command mode by pressing BREAK, then type "K filename." The disk whirs in the drive and it's done. The beauty of this command is that you can make room for your text to be stored on disk without the Scripsit Catch-22: with unmodified Scripsit, you must exit the program—and thus lose the text—to make room on the disk for the copy you just lost! The combination of the Kill and Directory commands alone make *SuperScript* worth its price.

When Radio Shack first released Scripsit, there was a tremendous flurry of interest among business people. The furor was quickly dampened, however, when people discovered that the limitations of Scripsit were simply unacceptable in a business environment. Had *SuperScript* been available at that time—and recommended by Radio Shack personnel—sales to the business and professional communities would have increased significantly.

Disadvantages

But is everything about *SuperScript* just peaches-and-cream? Well, almost. There are some areas that could stand improvement. The program documentation, while very complete, is not easily digestible by computer novices and needs to be read a couple of times before it can be fully understood. I would dearly love to have seen included a separate card with a summary of the new commands and control characters. A set of stickers to use in labelling the new control keys would also be nice. It's little touches like these—however trivial—that make life easier for the end-user of a fine product like *SuperScript*, especially if that end-user is not a computer hobbyist.

Summary

All told, *SuperScript* represents a very significant improvement over Radio Shack's unmodified Scripsit. But even amid this wealth of added capabilities, there have been a few perfectionists who still suffer from the dread "if only..." syndrome. To satisfy them, Acorn Software has released an expanded (and more expensive) version of *SuperScript*. It includes user-defined special characters and control codes; variable repeat speed; automatic saving of the window width, tab stops, and paragraph indentation settings; an indication of how many

Figure 1. Example of SuperScript Text.

Here are some of SuperScript's special functions: With SuperScript, any text -- even if justified -- can underlined. Horizontal lines _____ can also be drawn easily. **Likewise, you can specify boldface for any section of copy and underline the boldface text, if you like.** For research works or mathematical papers, you can call for superscripting or subscripting, and to make the distinction between the letter "O" and the number "0" perfectly clear, you can specify slashed zeros:

$$X^2 + Y^3 = \text{Log}_e 3.1000331$$

For special emphasis on indented text material, SuperScript allows you to "toggle" between 10- and 12-pitch type, and to insert a printer pause to change type wheels.

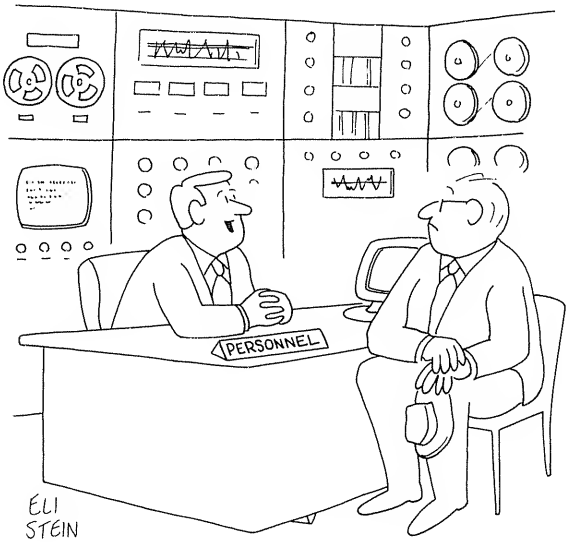
> The resulting document can be very impressive looking when you use appropriate **SuperScript** features. But don't overdo it -- like this example -- or it will simply be distracting and difficult to read.

All the features of SuperScript have not been used in this example, but it should offer enough of an example to show the utility of this clever modification to SCRIPSIT.

grans of disk space will be needed to save the current text; and the ability to load both Scripsit and the desired text file directly from DOS by typing "SCRIPT filename."

Additionally, Acorn has included a separate program titled, appropriately, "OOPS." This little gem will allow you to recover all or most of your Scripsit text after you accidentally press the reset button, issue the "END" command, or if the system "hangs up" for just about any reason before saving your copy on disk.

If you don't need the extra bells and whistles, the original version of *SuperScript* continues to be a super bargain at $49.95. On the other hand, the combination of Radio Shack's Scripsit and Acorn Software's expanded *SuperScript* is a word processing system that is hard to beat at any price. □



"Well... so you're a mimeograph machine operator. What's a memeograph?"

Scripsit In a Blue Cape

Super Scripsit

Dan Robinson

It's *Scripsit* in a blue cape. It can't leap tall buildings in a single bound, and it's still slower than a speeding bullet, but *SuperScripsit* is well named, for it is truly a super program.

Radio Shack's new word processor teams the TRS-80 with the Daisy Wheel II printer to produce proportional, justified printing that approaches book quality. Priced at $199, the program is much easier to use than its earlier Clark Kent version and contains most of the state-of-the-art features found in modern word processing programs.

The *SuperScripsit* package includes a pair of disks for the Model I and a pair for the Model III, eight half-hour lessons on audio cassettes, a 109-page training manual, a quick reference card, and a 144-page reference manual.

SuperScripsit may be called from TRSDOS together with the name of the text file to be processed, and fires up with the first of many menus which make the program user-friendly.

The Display

When entering text or editing, 14 text lines at a time are displayed above a format line. This line depicts margins and tab settings in inches and contains a ghost cursor that keeps pace with the text cursor. Below is the status line which contains the document name, the printed page and line numbers, and the position of the cursor in inches. The print pitch and line spacing for the current paragraph are shown together with indicators when certain modes are active.

Note the distinctions: although the screen may give equal space to every character, format, and print symbol, the status line is smart enough to consider only the space the characters will require in print, even given the variable width of proportional-font characters. Second, note that each paragraph may have its own margins, tabs, and line spacing.

The status line is further used as a mini-menu in answer to certain instructions and if an improper command is given, the status line is replaced by a flashing error message.

Text Control

SuperScripsit uses the familiar @ key for control in conjunction with letter

Dan Robinson, 1625 Higgins Way, Pacifica, CA 94044.

and symbol keys. Unlike the original *Scripsit*, all of the combinations are logical: Control-D is used for delete, Control-I for insert and so on.

SuperScripsit offers a large number of cursor movements with the arrow keys. By themselves, the arrow keys move left and right a character at a time and wrap around to the next line when they reach the end of the current one. The up and down arrows move one screen line at a time and when shifted move to the beginning or to the end of the text file. A shifted left arrow moves to the left margin while a shifted right arrow moves to the next tab stop.

An arrow pressed in conjunction with H or F moves to the header or footer page, while an arrow pressed with W, G, P or V moves to the next/previous word, paragraph, printed page, or video page. An arrow with N permits an input to move to a specified printed page, and with an L to a desired line number on the current printed page. Finally, an arrow with S will move the cursor to a specified search string.

Other common commands are a little more sparse. For example, there is only one way to delete text: hold the control key and D to eat a character at a time. The alternative is to designate the text to be deleted as a block, call the block action mini-menu, select Delete, and finally answer an "Are You Sure?" prompt to erase the text.

Insertions are a little simpler. A control-I opens the text for unlimited insertion. When the insertion is completed, a delete command closes the text.

Special Features

A unique feature of *SuperScripsit* is the align tab. When this function is called, the text moves to the left of the cursor as it is entered until the decimal point is keyed, and then the remaining input flows normally to the right. The result is an easy-to-use way to ensure that lists of figures will remain neatly in column even if the format of the paragraph is later altered. The align tab symbol may be changed from the default decimal point to any character, if needed to maintain a columnar presentation with product or part numbers and the like.

Another unusual feature offered by *SuperScripsit* is the tab line. Each paragraph contains its own tab and margin settings and tab and margin settings can be changed up to 50 times in a document. The tab line contains the left and right margins, tab settings, and an Indent tab to position the beginning of each new paragraph. The Indent tab may be set beyond the margin to provide reverse or hanging indentations for outlines.

SOFTWARE PROFILE

Name: SuperScripsit
Type: Word processor
System: TRS-80 Model I, III
Format: Disk
Language: Machine
Summary: Mild-mannered Scripsit now bends steel in its bare hands
Price: $199
Manufacturer:
Tandy/Radio Shack
1800 One Tandy Center
Fort Worth, TX 76102

The paragraph may use the system tab or call upon one of ten special tab formats stored by the program, all of which are programmable by the user. A quick-set command may be called to set margins and the indent tab for individual paragraphs. A separate tab Help Menu appears if blunders are made while formatting tab lines.

If the print width is greater than the 64 characters available on the TRS-80 screen, the text may be "windowed" in eight-character increments to the left or right. Unlike the original *Scripsit,* the new program with its wraparound cursor movements makes windowing an easy-to-use aid to ensure that the printed copy will appear as the user intended.

Search and Block Commands

The search routine in *SuperScripsit* presents a menu with find/delete/replace choices. The program may be instructed to make searches independent of whether the string is in upper or lower case, or optionally will respect the case of the string. A search may be specified to locate only complete words which equal the string or to find the string even if it is embedded within a longer word. Searches are made from the cursor position to the end of the text file, and may be set to pause after each match to permit selective deletions or replacements or continue through the text uninterrupted.

Many of the functions of *SuperScripsit* are performed through block commands. A block of text may be marked by moving the cursor to the beginning and then the end of the block, or by a Quick command which prompts for block identification by word, sentence, paragraph, printed page, from the cursor to the end of the text file. A block action command then presents a mini-menu from which a choice may be made to perform the block functions.

If the cursor is moved to a new position, Adjust gives the marked block the same format as the paragraph in which the cursor is located. Copy writes the block to disk and then duplicates it in the new location, while Move performs the move and also deletes the copy from the old position. Recall inserts the block from disk at the current cursor location, even if it was saved from a previous file. Print delivers a hard copy of the marked block, and Linespace alters the spacing of that block only.

Freeze makes the block impervious to change until it is unfrozen so as to preclude alteration if the text is later manipulated as part of a larger block. There is a Search option which functions like the global search but is confined to the marked block.

The Hyphen option brings the cursor to the segment of the word on the following line which would fit in the current line if hyphenated. The cursor can then be moved to the proper break point, and when a hyphen is struck, the word is broken and a hyphen inserted. The text is reformatted and the cursor moves on to the next hyphenation opportunity.

Other Features

A Control-C commands centering of the paragraph where the cursor rests, and whenever the cursor is moved over a centered paragraph, a Cen is displayed in the status line.

Text display may be toggled with a view mode to show the text as it will appear in print or to display format and print symbols as an aid in editing.

Ten user keys can be programmed to function with the control key in a sequence of up to 127 keystrokes each. A key sequence could, for example, insert a company letterhead or logo with a dot matrix printer, or type the complete standard signature block. The user keys can chain to one another and can loop back on themselves to provide such functions as scrolling.

SuperScripsit can be made to recognize widows so that the first line of a new paragraph doesn't end a page, or the last line of a paragraph hangs by itself at the beginning of a new one. When ordered to do so *SuperScripsit* will avoid widows by printing one more or one less line at the end of a page.

Headers and footers may contain up to 768 characters each. They may be identified to appear on odd, even, or all pages, and a menu option permits identifying the first page on which they will appear.

Printing

SuperScripsit has built-in functions to support the ability of the Daisy Wheel II to use sub- and superscripts, bold printing, strike-through, underscore, and double underscore. A top-of-form command sends reverse linefeeds equal to the number of lines printed on the current page to support two-column printing. Proportional or mono spacing or no justification at all can be specified, and the print pitch can be set to match the current wheel. Line spacing may be set for single to triple spacing with half-lines supported.

The printer menu permits commanding a page pause to insert a new sheet of paper at the end of each page. The program also supports a pause print command which can be placed anywhere in text to enable the operator to change daisy wheels before resuming printing.

SuperScripsit supports the Daisy Wheel II: Lineprinter IV, V, VI, and VIII; DWP 410, DMP 200, 400 and 500; and offers a general serial driver, as well. Several sample text files are provided on the disks to lead the new user through the lessons.

The CLEAR key is used to denote print codes embedded in the text, with the following character interpreted by the printer driver to activate the function. The *SuperScripsit* user can program both the shifted and unshifted number keys to operate in conjunction with the CLEAR key to send up to 20 sequences of special printer codes.

Menus

SuperScripsit calls up several menus to support user action such as opening a document, printing, or setting system parameters, and another pops up to offer options for Search. Seven screens of Help menus may be called in addition to the special Tab Help menu. Mini-menus appear for block actions, entering headers or footers, and Quick setting of linespacing or margins. Most menus call for a simple response, but some require editing to establish new *SuperScripsit* default parameters.

Form Letters

SuperScripsit contains a file merge capability which allows inserting of data from a variables file into the text for form letter production. For example, if the program encounters /NAME/ in the letter it will insert the data from the NAME field in the variables file. One copy of the form letter will be printed for each record in the file. *Profile III* may be used to create variable data for merged form letters, or they may be written using *SuperScripsit*, with the variable field names entered as the first record.

Documentation

Both the Figures training book and the Reference Manual are tabbed for quick access, and both are clearly written. The reference manual contains a copy of the Daisy Wheel II printer driver source code as an aid to Model III users who need to write a driver for their own printers. Model I users will need a bit of experience to write a driver, as the differences are not defined.

The training manual, cassette instructions, and sample text files on disk provide a painless introduction to *SuperScripsit* operation. Even a beginner in word processing will find that the course makes learning word processing easy following the one-step-at-a-time format. Emphasis is on the Daisy Wheel II, however, and users of other printers will miss many of the finest features of the program.

Utilities

The program has a utility module to compress a *SuperScripsit* file to its minimum size, although there must be sufficient space on the disks to contain both versions simultaneously. Other utilities convert *SuperScripsit* files to ASCII and back again to retain compatibility with old *Scripsit* files and Basic programs saved with the A option.

The Proofread option uses the Scripsit Dictionary in a second drive on the Model III with a 73,000 word vocabulary. The Model I requires three drives and is limited to 34,000 words.

The Warts

The *SuperScripsit* text buffer holds nearly 12,000 characters, and the text file is written to disk automatically when the buffer fills or when the user is ready to terminate the session. For safety's sake, the user may empty the buffer to disk at any time. Since *SuperScripsit* writes the file to disk as it grows, the length of a document is limited only by disk capacity, and with a second Model III 40-track, double density drive that quantity is awesome.

Manipulation of very large files is possible, but can be a slow process when dealing with more than 12,000 characters. The user must wait for the text to be read from and written to disk even to move from one end of the file to another. Old habits of dealing with small segments and then combining them for printing will have to be changed, for there is no way to merge even two small text files. Smaller segments must be stored one at a time as a block and later inserted into a second file.

SuperScripsit must be used without a write-protect tab. TRSDOS will write marked blocks to drive 0 when identified for a move or copy, although text files may be identified to be written to other drives as a part of their file specification. The program has its own directory command. This is not supported on the Model I however, and my efforts to make *SuperScripsit* function with operating systems other than TRSDOS failed.

Surprisingly, three common features found in other word processors are absent: the program does not support keyboard entry during printing, nor does it allow chain printing of lengthy documents or support insertion of "boilerplate" or standard paragraphs from disk. The more versatile delete commands found in other programs would be a blessing.

Summary

But that's being picky, for *SuperScripsit* does just about everything and does it very well, indeed. Users can be certain that Tandy programmers or outsiders will quickly have patch programs on the market to provide the few features that *SuperScripsit* lacks as well as the added utilities that are only now beginning to appear to enhance older word processors.

The program truly is *Scripsit* in a blue cape: it's *SuperScripsit.* □

Electric Pencil 2.0

Jim Klaproth

Back in the early days of the TRS-80, there existed a much needed piece of software called the *Electric Pencil.* It was written by Michael Shrayer and it performed the magical feat of turning an inexpensive microcomputer into a expensive device known as a word processor. It, being the only product of the sort on the market, soon became one of the most popular programs for the Tandy wonder machines (and a multitude of others). Then along came several competitors, each offering more capabilities than the previous ones. When Radio Shack announced Scripsit for $50 less than *Electric Pencil* and with more features, it set a new de facto standard for word processors, and several people gave up their *Pencils.* In early 1981, *Electric Pencil* died a timely death due to the increasing ground lost to other word processors. IJG, Inc. decided to revive it by releasing *Electric Pencil 2.0.,* which allows more sophistication in text processing and print formatting, yet maintains the simplicity of the original.

Electric Pencil now comes in three versions: tape, disk, and stringy floppy. The disk version, used for this evaluation, sells for $89.95 and comes supplied on a single density, 35 track, Model I format diskette, which contains a transfer utility to transfer it to your system diskette. The operating systems supported are TRSDOS 2.3, NEWDOS 2.1, NEWDOS/80 1.0, NEWDOS/80 2.0 for the Model I and TRSDOS 1.2, TRSDOS 1.3, and NEWDOS/80 2.0 for the Model III. In addition, the disk version supports cassette and stringy floppy I/O. Model III users must CONVERT all four *Pencil* files, meaning that single drive owners must find a two drive system to use to accomplish this task. The tape versions sell for $79.95 and require only a 16K Model I or III, both of which are on the same tape. The Model III version supports both 500 and 1500 baud, while the Model I runs only at 500. The stringy floppy version supports only the EXATRON stringy floppy for I/O.

Jim Klaproth, 2012 25th Ave., S.E., Puyallop, WA 98373.

SOFTWARE PROFILE

Name: Electric Pencil 2.0

Type: Word Processing

System: TRS-80 Model I and III, LNW
Any operating system

Format: Cassette or disk

Language: Machine (transparent to user)

Summary: Comprehensive, user friendly word processing package

Price: $89.95

Manufacturer:
IJG, Inc.
1260 West Foothill Blvd.
Upland, CA 91786

Text Entry

Typing PENCIL from DOS READY loads the main module, then displays the banner page. Pressing any key clears the screen and a flashing block cursor appears in the upper left corner. Characters are entered as on a typewriter and when the cursor reaches the end of a line, if the word being typed will not fit on that line, it drops down to the next line with the cursor. In testing this new version, no characters were lost as in the original *Electric Pencil.* The Cursor Control commands move the cursor in various ways. The arrow keys move it in the corresponding directions anywhere on the screen, as well as cause the text to vertically scroll. Additionally, the cursor can be sent to home position, to beginning of file, to end of file, to beginning of the current line, to tab 8 spaces right, or to scroll vertically (at 5 different speeds, and with pauses).

The Editing Commands are all two keystroke commands. These include delete character, insert character, delete current line, insert line, erase to end of line, back space and erase character, and insert and delete block of text (a "block" is one to several thousand characters marked with a block marker). The Special Character Keys allow the typing of a form feed, an underline character, and a paragraph terminator. The CLEAR key functions as the Control key and this may be used in addition to the modified control key that had to be installed on machines running the original *Electric Pencil.* There is also a upper/lower case lock (one minor irritation was that the default condition is upper case only), and a useful feature (for secretaries who must take dictation) called DICT-A-MATIC. This allows the user to remotely start and stop a cassette player by using two keystrokes. This toggles the cassette motor port in the computer, thereby turning on or off the cassette player attached to the computer in order to allow typing of the material on the tape.

The Utility Functions include a string search of from 1 to 39 characters, a search and replace function in order to replace the found string with a new string, a repeat function that repeats any function a specified number of times, and a "wild card" feature for coded search and replace opera-

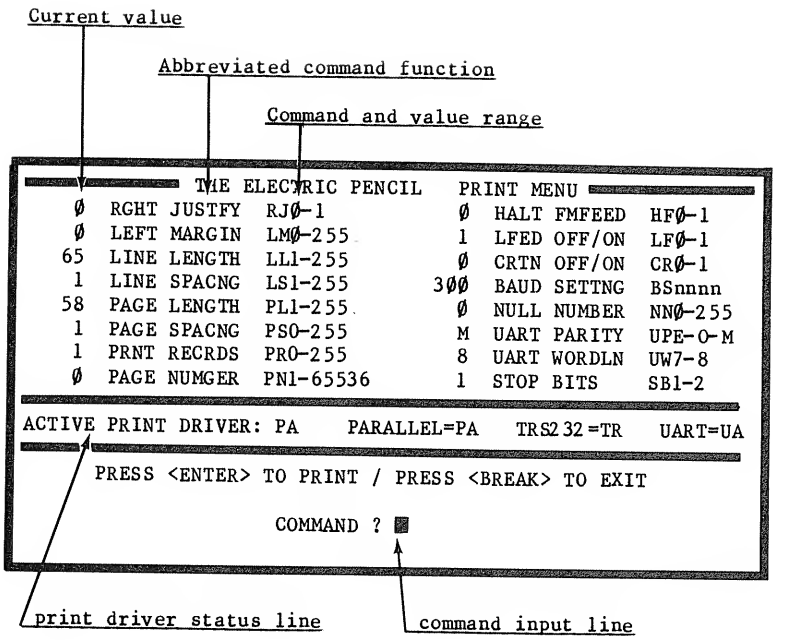
tions. Also in this group of features are the keystrokes to call up the System Menu and the Print Menu.

The System Menu

The System Menu, when invoked, replaces the normal text region with a full page of information. To return to the text entry mode, the user simply presses the BREAK key. The menu displays the number of words used, the number of records used (a "record" being any block of text that is terminated by a form feed or a paragraph terminator), and the amount of free memory left. The File Commands allow you to load, save, and verify tape files; load, save, and erase stringy floppy files; and, load, save, and kill disk files. In addition, all *Pencil* files with the /PCL extension can be displayed in the diskette directory. The System commands enable you to set the cursor speed and the baud rate for tape, to save and load the print driver, to clear all text before and after the cursor and from memory, and to return to DOS READY. There are also three Special Utility Commands, not displayed, that allow you to save the program configuration (disk only) so that the defaults concerning cursor speed, print values, and tape speed will be used automatically in subsequent sessions.

The Print Menu

After the text has been entered into the text buffer, it must be properly formatted and then sent to the printer. It is this area of word processing that separates excellent software from the mediocre. The criterion that this reviewer uses when evaluating word processors is, "Does the package allow full use of all the features built in to the printers available to the general population of computerists?" In other words, does the word processor support proportionally spaced, right justified printing, underlining, subscripting, superscripting, bold face, double width, italics, font change, graphics, and special symbols. *Pencil* supports only a few of these functions with the basic module. However, IJG plans to market several companion programs such as *Pencil Tip,* which allows embedded codes to dynamically format your printed output, *Blue Pencil* and *Red Pencil,* which are spelling checkers, *Electric Type,* which is a typesetting interface, and several print drivers for various printers. They even have plans for a talking word processor for visually handicapped persons and a graphics addition to the standard package.



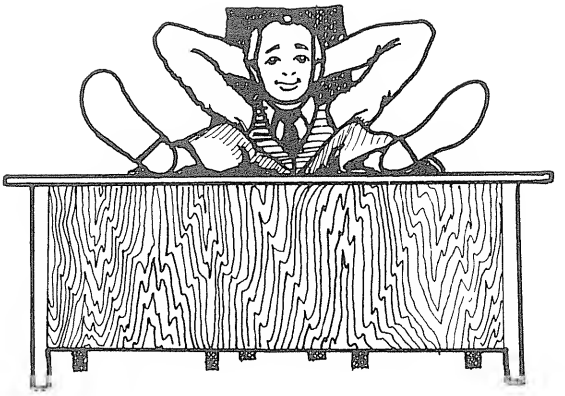
The Print Menu does allow control over most print values such as margin control, line length, line spacing, number of copies, number of lines per page, page numbering, right justification, and number of records to print. Various printer control commands control line feeds, carriage returns, pause on form feeds, RS232 parameters, and type of printer driver. The three supported are the standard Centronics parallel driver, the RS232 serial port (UART), and the Small Systems Software TRS232 serial printer interface. Title headings and page numbering are supported but not footers. Underlining is very clumsy and is supported only on printers that are capable of executing a carriage return without executing a line feed. Two lines of text are required to underline; one to display the underline characters (generated with a Control-O), and the other to display the text, and the underline characters appear above the text.

No direct method is given to embed escape codes into the text in order to control intelligent printer functions. There is a crude method of doing this that involves generating an ASCII text file with a list of escape sequences. This file is loaded first, then the normal text file. By using the block move command, the desired escape codes can be inserted into the proper text locations. This may be clever, but it takes more time and effort to do it. There are also a couple of "tricks" that allow double strike bold facing and extra-bold face letters, but again it is a clumsy process. No provision is made for right justifying proportionally-spaced printing or for subscripting, superscripting, italicizing, or generating special characters that are not on the keyboard.

Conclusion

The documentation consists of an attractive softcover manual. It contains about 120 pages and includes a quick reference card. It is clearly written and well layed out with a table of contents, a glossary of terms, and several appendices. There is a section on proper diskette handling and one on backup procedures. Also included are techniques on editing Basic programs and Visicalc files on disk systems.

My overall impression was that *Electric Pencil* is a well thought out package that certainly invites comparison by those who are considering Scripsit. In light of Radio Shack's policy of little or no ongoing support for their software, *Electric Pencil* may be a better choice for those wanting more features in the future. Pencil fans may also wish to purchase version 2.0 in order to have a modern product that maintains compatibility with their old *Pencil* files. It is my opinion that until the authors come up with a better printer formatter package, those that have more demanding word processing tasks will continue to choose more powerful packages such as Newscript or Lazy Writer. For more information, write IJG, Inc., 1260 W. Foothill Blvd., Upland, CA 91768. □



Lazy Writer

George Bond

SOFTWARE PROFILE

Name: Lazy Writer
Type: Word Processing
System: 32K Model I, III TRS-80, Disk Drive
Format: Disk
Language: Machine Language
Summary: Very easy to use
Price: $175
Manufacturer: SSM, Inc.
6250 Middlebelt
Garden City, MI 48135

This is being written on a TRS-80 Model I computer with ease and flexibility that previously had been available only on larger computers with much more expensive word processing programs. What has moved the Model I up into at least the Triple-A League, if not the true Big Leagues, of writing is a modestly priced word processer called Lazy Writer.

The best thing about Lazy Writer, from the viewpoint of a professional writer and editor, is not that it does exotic things, undreamt of by users of Electric Pencil and Scripsit. It is that it does the same things in a much simpler way.

Gone is the vexing problem of how to get a control key on the minimal keyboard of the TRS-80—control keys are not needed. More important, gone is the need to remember what the control key does, or to stick those sticky little reminder labels on the keys. And gone is the need to have the manual dexterity of a Mississippi River cardsharp to manipulate control-shift-character key combinations.

The program comes on a TRSDOS disk, which can be backed up as many times as necessary. Lazy Writer can also be run using the Dosplus 3.3D double-density operating system with a Percom Doubler board in the expansion interface. It also seems to run with the Percom DBLDOS system (which comes free with the doubler board), but I have not checked this combination extensively yet. It also works with NewDOS/80 version 2. Unfortunately, it will not work with NewDOS80 and DoubleZAP/II for double-density operation.

After booting the disk, the main menu for the program is loaded by pressing the "L" key. The main menu gives the choice of starting in "Text Entry" or "Edit," or to recover text from memory.

Switching between Text Entry and Edit is the key to Lazy Writer's simplicity. To switch from one to the other, you need only press the break key. The 15th line of the screen carries a message noting that the program is in Text Entry and the break key will move it to Edit. The 16th line in Text Entry lists the number of characters in your text, the maximum number of characters your computers memory can handle, the location of the cursor and the length of the line the cursor is on.

Entering Text

To write, start the program in Text Entry and begin. The only basic differences from writing on a typewriter are that instead of using shift-lock (which doesn't exist on the TRS-80 keyboard) to stay in capital letters, shift-clear is used, and the right-arrow key provides a five-space indentation after the enter key marks the end of a paragraph. That's it, all you need to know about Lazy Writer for bread-and-butter banging out of The Great American Novel.

Things are a bit more complicated in Edit, but not much. Typing "H" calls up a help file, which explains in detail what functions are available and how to use them (and you can add your own messages to fill in any personal blind spots about the system's operation).

The screen remains split in Edit, but the 15th line become a dashed separator and the 16th line tells only cursor position and line length, until one of the editing commands is used. Then the 15th line disappears and the function—"INSERTING," "DELETING," etc.—is printed as the 16th line.

The editing commands are extremely simple: "i" to insert, "d" to delete, "o" to overtype, "c" for change case (to make "DOG" into "dog" type the lower-case c over the D, O and G and there it is). Pressing the enter key makes your changes permanent; pressing break cancels them. The cursor can be moved up, down, right and left as usual with the arrow keys.

One of the really nice features of Lazy Writer allows skipping the cursor from word to word by pressing the space bar, sentence to sentence by pressing the period (".") and paragraph to paragraph by pressing the lower-case "p." The cursor can be moved backward in the text by using the shift key and the space bar or period.

These same functions can be used when deleting—"d" and space bar erase the next word, "d" and "." erase the next sentence, "d" and "p" erase the next paragraph. Single characters can also be deleted.

Equally simple commands allow Lazy Writer to search for strings of characters, words or strings of words; to search and delete; and to search and replace. Moving blocks of type is easy, too, and requires few keystrokes.

When using the print functions—you can select standard Centronics parallel, serial or special drivers from the keyboard—there are two choices for establishing the format of what you print.

The simpler method is to use a menu that appears when the print function is selected. Using the menu, you can change

George Bond, 328 8th St., SE, Washington, DC 20003.

margins, line spacing, page length and such, as well as order multiple copies of your printing.

The more complicated method of entering printing instructions requires putting them directly into the text—"imbedding" them. This has several advantages, chief among them that the commands are permanent. This means that if you need to send out promotional form letters, for example, with some regularity, they will be uniform in appearance without any extra effort by you after the initial set-up. You also can imbed commands to tell your printer—if it is capable of doing so—to switch to double-wide characters, underline, overstrike for bolder letters, change line width and margins in the middle of text, and work all sorts of graphic magic. With enough patience, you could even set up little indented boxes for later insertion of illustrations.

Another nice feature of Lazy Writer is an auxiliary program to send what you write over telephone lines (using, of course, an RS-232 port and modem) to other computers. It also allows taking a read-only look at files from disk—in effect, listing them as you would from the DOS level—without leaving the editing program or destroying text in memory.

Documentation

All the functions of Lazy Writer are covered well in its documentation, which ranks among the best I have seen. It is on a par with the construction manuals for Heathkits and Dynakits.

Finally, a covering note in the documentation from Dave Welsh, who wrote Lazy Writer, bodes well for its future: "We view widespread misunderstanding (of how to use the system) as a program or a documentation fault to be corrected if possible."

If more authors, and computer people in general for that matter, would take such an attitude, life among the bits and bytes would be infinitely more enjoyable. □

The Copyart Word Processor

Jim Klaproth

Copyart is a full-featured word processor from Simutek, the people who brought us the Zbasic compiler for the Models 1 and 3, Written using the Zbasic compiler, it has all the speed of a machine language word processor. Why another word processor to compete with Scripsit, Lazy Writer, Newscript, Superscript, Pensadyne, Electric Pencil II, Query, and the others? Well, *Copyart* is a little different. It is useful as a word processor, but its true power is in its ability to do high resolution graphics easily and quickly, and in combination with your normal text.

Copyart is patterned exactly after Scripsit in screen layout, with 14 lines reserved for text entry, the 15th line is a horizontal bar, and the bottom line is reserved for messages. A large, flashing block cursor is standard. The cursor is positioned with the 4 arrow keys and a line is ended with the ENTER key. Automatic line wrap-around occurs when the end of the line is reached. The video line width can be set from 64 to 255. The "at" symbol key functions as the control key. No provision is made for entering this symbol in text, which could be a problem for those that need it for invoices. The CLEAR key functions to tighten up text after an inserted line leaves a gap.

Jim Klaproth, 2012 25th Ave., S.E., Puyallop, WA 98373.

SOFTWARE PROFILE

Name: CopyArt
Type: Word Processor
System: 48K TRS-80 Model I, III
Format: Disk
Language: Machine
Summary: Good word processing system, with efficient hi-res graphics capabilities
Price: $149.95
Manufacturer:
Simutek, Inc.
4877 E. Speedway Blvd.
Tucson, AZ 85712

The ease of learning and using *Copyart* is due primarily to the use of simple control keys that use the first letter of each function. Some of the functions are [d]elete, [i]nsert, [q]uick line delete, [t]ab set, [k]ill tab, [s]tatus, [h]elp, [f]ill screem from buffer, [g]raphics on [c]ontrol code, and [w]ord delete. Simplicity is the keyword here. I found that I could use *Copyart* like an expert after a quick trip through the manual. The mnemonic commands make remembering the control keys a breeze.

Boldface and underlining are included for printers that have these capabilities. Italics can be done on the Epson printers with Graftrax by inserting the proper control codes into the text, as can any other special function. Find and replace functions are included, with a repeat function in

order to change several or all occurrances of a certain word. Block move is done a little differently on *Copyart*. Each line of text to be moved is placed in a buffer and then popped out wherever you wish to move it to.

Another nice feature is horizontal scrolling. Whenever the video line width exceeds 64 characters, the screen scrolls beyond the 64th position in order to enable more text to be entered. It is much faster than the corresponding function in Scripsit.

Printer format commands are used throughout the text to tell the printer how to format the text properly. The commands include page length, top and bottom margin, right and left margins, headers, footers, page numbering, single or double spacing, justification (on, off, or ragged right), emphasized print, change character size, and proportional spacing. Proportional spacing is supported only on the Centronics 737/739/Line Printer IV, Right justification of other proportional printers such as NEC, Diablo, C. Itoh, and Smith Corona is not supported at the present time, but is being worked on. The main printers that are supported are the Epsons, the Okidatas, and the Centronics 737/739 series. However, any parallel printer that has the ability to turn off automatic linefeeds after carriage returns can be used. Serial printers are not supported at all at this time.

One hardware limitation is the mandatory installation of a lower case modification in the machine. *Copyart* will not work without one. The manual provides instructions on how to add this to your Model 1, or you can have Radio Shack or Simutek install one for about $50. Another novel feature of *Copyart* is that it is a protected program. However, you can make all the copies you want and even give the copies to all of your friends. The catch is that the software is matched to a hardware "key" and will not run without it. The "key" is a potted DIN connector that plugs into the cassette port of the computer. Simutek claims that there are 18 versions of the key, each with a different code in the serial number. If you lose your key, you may obtain a replacement for $40 and only one extra key is alloted per customer. Pretty slick!

Graphics is the area where *Copyart* really excels. By simply hitting control-g you are in graphics mode. At this time there are 3 modes: 1) draw, which leaves a lit pixel wherever you move the cursor, 2) erase, which leaves an unlit pixel wherever the cursor is moved, and 3) move, which simply allows movement without change. The cursor has two speeds: fast and slow. There is also a built-in graphics character set which creates large graphic characters automatically. You can control the horizontal and vertical size of the letters, the direction of the letters (horizontal or vertical), and regular or inverse video background. The letters are actually quite attractive. Graphics can be intermixed with normal text and can even be used in headers or footers, like a logo.

Printing graphics is done very nicely by the Epsons and Okidatas, but what about the others? Well, *Copyart* has pseudographics that allow you to do things like bar graphs, posters, and other low resolution graphics. This is accomplished by overstriking two characters to form a solid dot. The built-in graphics characters do not print that well on my Line Printer IV, but simple graphics look pretty good. Actually, any printer with the ability to turn off linefeeds can do pseudographics.

The thing that really impressed me about *Copyart* was the ease of learning the system. This was definitely the easiest word processor to learn that I have ever used. There are some features that are missing, like subscripts and superscripts, but how often does one use these bells and whistles? The main features of boldface, underlining, and double wide are uncomplicated. The things that were missing in Scripsit are included in *Copyart,* such as a directory function, a help function, and the ability to kill, append, and chain files, and having a program that runs without patching your favorite operating system. Add the ability to do graphics, and you have a first rate package for a $149.95 selling price.

Simutek is certainly one of the pioneers in the ever-changing world of TRS-80 software. My prediction that we would soon see some first rate software created with the Zbasic compiler is already coming true.

For more information, write Simutek, Inc., 4877 E. Speedway Blvd., Tucson, AZ 85712. □

The making of a magazine . . .

Typecasting

David Lubar

"We're going to be doing all our typesetting right here, on TRS-80's." That was just one of the many statements I heard back in June of 1980 when I started with *Creative*, and I didn't pay much attention to it at the time. I had no hint that computerized typesetting and I would become linked in a turbulent affair which would span months of frustration, triumph and despair.

The prototype system arrived a few days later. Gathered with a crowd of onlookers, I watched as the three crucial components were integrated. There was the familiar TRS-80 Model I with expansion interface and two disk drives, there was an Alpha Comp typesetting machine, and lying between them, the G2 interface. It was, indeed, a prototype, housed in a cardboard box and hand labelled. The interface took ASCII data from Electric Pencil or Scripsit files and translated it into codes understandable by the Alpha Comp. The box was the heart and brains of the system. The box was the crucial link. The box almost worked.

Irwin Gretsco, father of the G2 interface,

David Lubar is a past associate editor for *Creative Computing* magazine.

gave us a demonstration. "Now is the time for all good men to come to the aid of their party," he typed on the TRS-80. This was followed by a few control codes, and a few instructions to the Alpha Comp. "Here it comes," he said. We all craned forward, staring at the single-line LED display of the typesetting machine. There were hums and whirrs. Letters appeared, showing the text being set. "Now is the time for all goo men tocome tothe aid of their party."

Irwin mumbled something and proceeded to make a few solder changes in the interface, ignoring the suggestion that he might want to turn off the power first. Another runthrough produced similar results. We were informed that there must be heavy industry in the area fouling up the power lines. Since our heavy industry neighbors at the time were a pizzeria and a deli, this didn't seem likely. The trouble was finally traced to a bad cable, and glitch number one vanished; making way for glitch number two. Fortunately, these early glitches soon gave way to transient problems which, while harder to trace, did less damage.

We began typesetting on premises (and on the premise that a new interface would take care of the problems in the prototype unit). The system still garbled an occasional line, but worked well enough to cut down on the amount of work being sent out for typesetting. A new interface was delivered within a few weeks. This one had a metal case, and wreaked no havoc on misquotes of Thomas Paine. The typesetters were getting used to the system, learning the meaning of DOS ERROR 22, and the value of triple backup disks. One could become an instant hero by reviving a dead disk. The people in typesetting and software discovered the meaning of synergy. Technology had finally caught up with us.

Still, the box had a habit of breaking down just before an issue deadline, producing frantic trips to the "professional" typesetter. The third box had even more bugs ironed out, and everything finally seemed to run smoothly. Well, not quite. Now that the box was working, it was time for the Alpha Comp to go flaky. I had the misfortune of being present during the first paper jam.

The fix involved turning the monster on its side, removing innumerable screws, and carefully peeling away pieces of paper from a razor-sharp knife poised on a spring control. Volunteering for the job once, I was blessed with it for life. Meanwhile, the typesetters were learning new joys, such as end-of-paper lights that didn't go on, fonts that couldn't tolerate any dust, and other random problems. But the thrill of seeing type roll out of the processor somehow made up for these minor aggravations.

There is a happy ending to this phase of the story. The system works almost all the time, allowing us to set the entire magazine, along with *Microsystems* and *SYNC*, right down the hall from the editorial offices. Between magazines, the typesetters also manage to set many of the new books published by Creative Computing Press, and all the documentation for Creative Computing Software packages. In-house typesetting definitely gives a boost to productivity.

Soon after this, the company moved to larger headquarters, taking over a building that had previously been a printing plant. In one of the rooms, as if a reminder of how far we'd come, sat a huge beast known as a hot-lead machine. This combination furnace and die caster creates type from molten metal, and probably doubles as a sauna. After a consultation with our efficiency expert, we decided to ignore the machine and stick with computers.

Eight Inch Blues

So, the system could take anything written under Electric Pencil or Scripsit and turn the text into typeset strips. That was fine for the typesetters, but left the editorial staff with one small problem. Most of us use systems with eight-inch disks. For example, the system I had inherited from my predecessor was an Altair running Electric Pencil under CP/M; the editor uses a SOL, the publisher has an Imsai, and two other eight-inch CP/M systems were lurking about. We had been told that the typesetting system might be able to handle eight-inch disks, but had no clue as to how to achieve that goal. Dual Omicron drives had been connected to the TRS-80 when the system was first set up. This allowed data to be read into

FROM TEXT TO TYPE

Text can either be entered directly into the Alpha Comp, or placed on disk first. The disk storage is preferrable for articles since it simplifies changes. The text contains embedded commands for the typesetting machine. For example, the equal sign indicates the start of a paragraph, and the percentage sign marks the end of a paragraph. The obvious question from here is, what if you want to print one of the reserved signs? This is taken care of by the memory capability of the Alpha Comp. It can store up to 1024 reserved characters. These memory fills are designated in the text with the symbols @N@, where N indicates which character to use. For instance, if a % is needed in the text, it can be designated as memory fill number 1. Then, whenever the text contains the symbols @1@, it will print a percentage sign. While this might seem to be a bit of a bother to enter into the text, don't forget that text is entered under Electric Pencil. Global search and replace takes the drudgery out of such tasks. The font is also controlled by text commands. A typical font disk for the Alpha Comp (this is the delicate item, dust being attracted to the combination of glass and film) contains three typefaces, usually standard, italic, and bold. With a command embedded in the text, the font can be changed at any time. One sentence can be in normal type. *The next can be in italics*. And another sentence or word can be in **boldface** type.

After the text is entered, hardcopy is produced and sent around for final editing. The changes are made on the disk, and it's time to typeset. First, the Alpha Comp is turned on, and any memory fills are defined. Then the command mode of Electric Pencil is used to set print parameters. Next, with a simple control-P, the characters start flowing through the interface and into the Alpha Comp. The flow continues without interruption unless a word won't fit on the end of a line. In such cases, the Alpha Comp beeps and waits patiently for the operator to hyphenate the word. Optional hyphens can be inserted allowing the machine to select the break which will produce the most even line of print. The text appears in a small window, and is sent to special paper inside the machine. After the text is finished, the paper is removed and put through a processor. This device, which resembles a mangler and is filled with chemicals that can strip the flesh from mortal bones, magically turns strips of paper into useable type. *Voila!* The article is typeset and ready for the art department. The entire process takes place within these very walls. From manuscript to camera-ready boards, we do it all for you.

the computer, but didn't seem to catch the attention or interest of the interface. Our resident hardware man at the time was sure he could effect a simple solution. Unfortunately, his efforts, over a period of a month, left us with a fix that did nothing whatsoever. He is no longer with us.

Determined to continue using the Altair, I got together with a software pro and decided to trash the original approach, starting fresh. Together, we came up with an idea that actually worked; just send the file right to the interface using the CP/M TYPE command. The next day, text was streaming off eight inch disks in typesetting. Those of us using Altairs, Imsais, and other vintage models breathed a sigh of relief. Those four-thousand word articles no longer had to be split into several disk files, and DOS ERROR 22 no longer reared its ugly head. Now, if I could only find out what's wrong with the top area of RAM in the Altair....

Changes

A few months after the eight-inch problem was resolved, the now-flaky TRS-80 was replaced with a seemingly more reliable LNW. This killed the eight-inch interface, making it necessary to download ASCII files from the Altair to a TRS-80, using LDOS, then take these files to the LNW. (If that description sounds to you like alphabet soup, you aren't alone.)

While the above may suggest that there are a few problems with the system, there are also definite advantages. The ability to keep track of a manuscript from start to finish is a great asset for any magazine. Also, duplication of work is avoided. When one of the staff writes an article, it doesn't have to be re-typed by the typesetting department. They can take the text right from the disk. Those horrendous monthly deadlines can be extended slightly because of the time saved here, so articles that would have been two weeks late are now only one week late. Eventually, we plan to set up a system that will translate files from any disk format. Already, some manuscripts for *Microsystems* and *Creative Computing* are taken straight from the author's disks, though it might be some time before the process is applied to *SYNC* magazine. The next step might be modems; we'll keep you posted. □

Sample text with embedded commands.

```
The font is also controlled by text commands. A typical font disk for the Alpha
Comp (this is the delicate item, dust being attracted to the combination of
glass and film) contains three typefaces, usually standard, italic, and bold.
With a command embedded in the text, the font can be changed at any time. One
sentence can be in normal type. <The next can be in italics>. And another
sentence or word can be in #boldface> type.%
```

"How fast can you process words?"

Chapter IX
Education

A Basic Generative Grammar Program

Kimball M. Rudeen

The article "Grammar as a Programming Language" (Jan/Feb 1978) contained examples of how to use the REPLACE command in Logo to generate sentences from syntactical rules. The following article shows how a similar approach can be done in Basic on the TRS-80.

This program was inspired by the article "Grammar as a Programming Language" in the Jan/Feb 1978 issue of *Creative Computing*. Since I had studied formal grammars in college, the idea of a program able to generate statements according to a given grammar was familiar and fascinating. I decided to write my own as soon as I had my own computer. The program described in this article is, I think, a decent attempt. It has all the necessary basic capabilities, plus a few frills which I found convenient.

The program, written in Basic for the TRS-80, is separated into five modes—Grammar, Generate, Edit, Read and Write. The capacities and operation of these modes are described in the following sections.

Grammar Mode

Grammar mode is used to input the rules of the chosen grammar. Rules are of the form

STRING = SUB1/SUB2/SUB3.../SUBN

where STRING is the string of characters to be replaced, and SUB1...SUBN are the possible replacement strings, separated in the rule by the "/" character. Recursion is allowed, and replacement strings may of course, be the initial strings of other rules. "Null" replacement strings, which replace an initial string with zero characters, are indicated by a double slash "//." Every time the Grammar or Read modes are entered, all previously defined rules are lost. An existing grammar may be modified by the edit commands (see Edit mode). Rules may be input by the user or read from tape, rules may be saved on tape (see Read/Write Mode). Figure 1 illustrates grammar input.

Kimball M. Rudeen, 57 Taft Ave., Lexington, MA 02173.

Figure 1.

```
MODE - GRAMMAR, GENERATE, EDIT, READ, WRITE, TERMINATE:
GRAMMAR
INPUT RULE - USE END TO HALT:
SEN=SUB VER OBJ
SUB=ARTICLE NOUN/NAME
VER=LIKES/HATES/SEES/RUNS AFTER
OBJ=ARTICLE NOUN
NAME=BILL/TOM/HARRY/JOE
END
```

Figure 2.

```
MODE - GRAMMAR, GENERATE, EDIT, READ, WRITE, TERMINATE:
GENERATE
INPUT START SYMBOL LIST - USE PERIOD TO END LIST AND END TO EXIT
GENERATE MODE.
1 SEN AND SEN
2 SEN
.
1 SUB VER OBJ AND SUB VER OBJ
1 NAME VER OBJ AND NAME VER OBJ
1 NAME SEES OBJ AND NAME RUNS AFTER OBJ
1 NAME SEES ARTICLE NOUN AND NAME RUNS AFTER ARTICLE NOUN
1 JOE SEES ARTICLE NOUN AND HARRY RUNS AFTER ARTICLE NOUN
2 SUB VER OBJ
2 ARTICLE NOUN VER OBJ
2 ARTICLE NOUN HATES OBJ
2 ARTICLE NOUN HATES ARTICLE NOUN
1 JOE SEES ARTICLE NOUN AND HARRY RUNS AFTER ARTICLE NOUN
2 ARTICLE NOUN HATES ARTICLE NOUN
FINISH
INPUT START SYMBOL LIST - USE PERIOD TO END LIST AND END TO EXIT
GENERATE MODE.
END
```

Generate Mode

Generate mode is used to input the list of strings to which the program will apply the grammar defined in the grammar mode. Each entry string may contain substrings not defined as an initial string in any grammar rule. Such substrings will not be changed by the program.

Each entry string will be treated separately. The program will continue to try the rules of the defined grammar against a given string, printing out intermediate results, until no more replacements can be made. If no grammar rule applies to a list entry, it will be left unchanged.

At the end of the process, the final result for each list entry will be printed out. The output subroutine is designed to avoid ending a line in the middle of a substring when possible. When this occurs, the subroutine will backtrack to a blank in the current string and end the line there. If no blank exists, the string will be printed as is. Note: a recursive grammar rule can generate an infinite loop. Figure 2 illustrates the operation of the Generate mode.

Generate Program Listing

```
10 REM     GENERATE: A GENERATIVE GRAMMAR PROGRAM
20 REM               BY KIMBALL M. RUDEEN
30 CLS
40 CLEAR 3000
50 RANDOM
60 DEFINT A-Z
70 DIM COM$(50), ND$(25)
75 REM  SELECT OPERATING MODE
80 PRINT "MODE - GRAMMAR, GENERATE, EDIT, READ,
   WRITE, TERMINATE     :"
90 INPUT M$:IF LEFT$(M$,3)="TER" END
100 IF LEFT$(M$,3)="GRA" GOTO 160
110 IF LEFT$(M$,3)="EDI" GOTO 2000
120 IF LEFT$(M$,3)="GEN" GOTO 220
130 IF LEFT$(M$,3)="REA" GOTO 4000
140 IF LEFT$(M$,3)="WRI" GOTO 5000
150 GOTO 80
155 REM  GRAMMAR RULE INPUT
160 CC=0
170 PRINT "INPUT RULE - USE END TO HALT:"
180 INPUT CC$
190 IF CC$="END" GOTO 80
200 GOSUB 1000
210 GOTO 180
215 REM  STRING GENERATION: INPUT START SYMBOLS
220 PRINT "INPUT START SYMBOL LIST - USE PERIOD
    TO END LIST AND END TO EXITGENERATE MODE."
230 CSD=0
240 INPUT ISD$
250 IF ISD$="." GOTO 280
260 IF ISD$="END" GOTO 80
265 REM  STRING GENERATION: SCAN STRINGS
270 CSD=CSD+1:ND$(CSD)=ISD$:GOTO 240
280 CD=0
290 CD=CD+1:IF CD>CSD GOTO 820
300 SD$=ND$(CD):CF=0:CL=0
310 CF=CF+1:CN=0
320 IF CF>CC GOTO 810
330 LN=VAL(COM$(CF))
340 M=4
350 IF LN>9 M=M+1
360 SUB$=MID$(COM$(CF),M,LN)
370 PS=0
380 LSD=LEN(SD$)
390 PS=PS+1
400 IF PS>LSD-LN+1 GOSUB 690:GOTO 310
410 NN=LN+PS-1
420 IF SUB$=MID$(SD$,PS,LN) GOTO 440
430 GOTO 390
435 REM  REPLACE STRINGS
440 LC=LEN(COM$(CF)):CL=1
450 M=LC-LN-4:IF LN>9 M=M-1
460 CN=VAL(RIGHT$(COM$(CF),M))
470 CM=INT(CN*RND(0))+1
480 PSD$=LEFT$(SD$,PS-1)
490 L1=LN+7
500 IF LN>9 L1=L1+1
510 IF CN>9 L1=L1+1
520 SN=0
530 FOR I=L1 TO LC
540 IF MID$(COM$(CF),I,1)="/" SN=SN+1
550 IF SN=CM GOTO 580
560 NEXT I
570 PRINT "ERROR";CM,SN:GOTO 820
580 FI=I+1:SM=0
590 FOR I=FI TO LC
600 CH$=MID$(COM$(CF),I,1)
610 IF CH$="/" GOTO 650
620 PSD$=PSD$+CH$
630 SM=SM+1
640 NEXT I
650 IF SM=0 NN=NN+1
660 I=LSD-NN
670 IF I>0 PSD$=PSD$+RIGHT$(SD$,I)
680 SD$=PSD$:ND$(CD)=SD$:PS=PS+SM-1:GOTO 380
685 REM  OUTPUT GENERATED STRINGS
690 IF CN=0 GOTO 800
700 LSD=LEN(SD$):M=1
710 LD=63
720 IF LSD-M+1>63 GOTO 750
730 PRINT MID$(SD$,M,LSD-M+1)
740 GOTO 800
750 FOR I=1 TO 63
760 IF MID$(SD$,M+LD,1)<>" " LD=LD-1:NEXT I
770 IF LD<>0 PRINT MID$(SD$,M,LD) ELSE PRINT
    MID$(SD$,M,63)
780 IF LD<>0 M=M+LD+1 ELSE M=M+63
790 GOTO 710
800 RETURN
805 REM  CONTINUE GENERATION OR TERMINATE
810 IF CL<>0 GOTO 300 ELSE GOTO 290
820 PRINT:CN=1
830 FOR CD=1 TO CSD
840 SD$=ND$(CD):GOSUB 690
850 NEXT CD
860 PRINT "FINISH":GOTO 220
870 END
895 REM  REPLACE RULE CF IN LIST
900 SC=CC
910 CC=CF-1
920 GOSUB 1000
930 CC=SC
940 RETURN
995 REM  ADD GRAMMAR RULE TO LIST
1000 CC=CC+1:CE=1:CA=0
1010 FOR I=1 TO LEN(CC$)
1020 IF MID$(CC$,I,1)="/" CE=CE+1
1030 IF MID$(CC$,I,1)="=" CA=I
1040 NEXT I
1050 COM$(CC)=STR$(CA-1)+"/"+LEFT$(CC$,CA-1)
1060 COM$(CC)=COM$(CC)+"/"+STR$(CE)+"/"
1070 COM$(CC)=COM$(CC)+RIGHT$(CC$,LEN(CC$)-CA)+"/"
1080 RETURN
2000 PRINT "EDIT - LIST, ADD, DELETE, MODIFY, QUIT:"
2005 REM  EDIT GRAMMAR RULES
2010 INPUT "EDIT:";E$
2020 IF E$="LIST" GOTO 2080
2030 IF E$="ADD" GOTO 2100
2040 IF LEFT$(E$,3)="DEL" GOTO 2130
2050 IF LEFT$(E$,3)="MOD" GOTO 2220
2060 IF E$="QUIT" GOTO 80
2070 GOTO 2010
2080 FOR I=1 TO CC:PRINT I,COM$(I):NEXT I
2090 GOTO 2010
2100 INPUT CC$
2110 GOSUB 1000
2120 GOTO 2010
2130 INPUT SUB$
2140 LN=LEN(SUB$)
2150 GOSUB 3000
2160 IF CF=0 PRINT "NOT FOUND":GOTO 2010
2170 FOR I=CF+1 TO CC
2180 COM$(I-1)=COM$(I)
2190 NEXT I
2200 CC=CC-1
2210 GOTO 2010
2220 INPUT CC$
2230 LN=0
2240 FOR I=1 TO LEN(CC$)
2250 IF MID$(CC$,I,1)<>"=" LN=LN+1:NEXT I
2260 SUB$=LEFT$(CC$,LN)
2270 GOSUB 3000
2280 IF CF=0 PRINT "NOT FOUND":GOTO 2010
2290 GOSUB 900
2300 GOTO 2010
2995 REM  LOCATE GRAMMAR RULE
3000 FOR CF=1 TO CC
3010 IF LN<>VAL(COM$(CF)) GOTO 3040
3020 M=4:IF LN>9 M=M+1
3030 IF SUB$=MID$(COM$(CF),M,LN) GOTO 3060
3040 NEXT CF
3050 CF=0
3060 RETURN
3995 REM  INPUT GRAMMAR RULES
4000 CC=0
4010 INPUT#-1,CC$
4020 IF CC$="END" GOTO 80
4030 CC=CC+1:COM$(CC)=" "+CC$:GOTO 4010
4995 REM  WRITE GRAMMAR RULES
5000 FOR CF=1 TO CC:PRINT#-1,COM$(CF):NEXT
5010 PRINT#-1,"END":GOTO 80
```

```
MODE - GRAMMAR, GENERATE, EDIT, READ, WRITE, TERMINATE:
EDIT
EDIT - LIST, ADD, DELETE, MODIFY, QUIT:
EDIT:LIST
 1                3/SEN/ 1/SUB VER OBJ/
 2                3/SUB/ 2/ARTICLE NOUN/NAME/
 3                3/VER/ 4/LIKES/HATES/SEES/RUNS AFTER/
 4                3/OBJ/ 1/ARTICLE NOUN/
 5                4/NAME/ 4/BILL/TOM/HARRY/JOE/
EDIT:DELETE
NAME
EDIT:MODIFY
SUB=ARTICLE ADJECTIVE NOUN
EDIT:ADD
ARTICLE=A/THE
EDIT:ADD
NOUN=CAT/DOG/BIRD/HORSE
EDIT:LIST
 1                3/SEN/ 1/SUB VER OBJ/
 2                3/SUB/ 1/ARTICLE ADJECTIVE NOUN/
 3                3/VER/ 4/LIKES/HATES/SEES/RUNS AFTER/
 4                3/OBJ/ 1/ARTICLE NOUN/
 5                7/ARTICLE/ 2/A/THE/
 6                4/NOUN/ 4/CAT/DOG/BIRD/HORSE/
EDIT:QUIT
```

Figure 3.

```
MODE - GRAMMAR, GENERATE, EDIT, READ, WRITE, TERMINATE:
GENERATE
INPUT START SYMBOL LIST - USE PERIOD TO END LIST AND END TO EXIT
GENERATE MODE.
SEN
.
SUB VER OBJ
ARTICLE ADJECTIVE NOUN VER OBJ
ARTICLE ADJECTIVE NOUN HATES OBJ
ARTICLE ADJECTIVE NOUN HATES ARTICLE NOUN
A ADJECTIVE NOUN HATES A NOUN
A ADJECTIVE BIRD HATES A BIRD
A ADJECTIVE BIRD HATES A BIRD
FINISH
INPUT START SYMBOL LIST - USE PERIOD TO END LIST AND END TO EXIT
GENERATE MODE.
END
MODE - GRAMMAR, GENERATE, EDIT, READ, WRITE, TERMINATE:
TERMINATE
```

Figure 4.

Edit Mode

Edit mode is used to modify an existing grammar. There are four commands: List, Add, Delete, and Modify. The List command will list the current grammar rules in their storage format. This consists of the original Grammar mode input format, plus two count values giving the length in characters of the initial string and the number of possible replacement strings.

The Add and Modify commands allow new rules to be changed. The Modify command requires the modified rule to be specified completely. The Delete string needs to be given. Figures 3 and 4 illustrate the operation of the Edit mode.

Read/Write Modes

Read/write modes are used to input or store in a grammar. I/O facilities may have to be provided by the user. The subroutines shown are designed for my machine, a TRS-80. These modes will save the trouble of redefining a complex grammar.

Applications

Applications of this program are many and varied. The examples given suggest demonstrating the rules of English grammar. The same rules defined "in reverse" would analyze a given sentence into its component parts. This program can be used to generate or analyze bars of music, statements in languages such as Pascal, graphic commands, or speech synthesizer commands—almost anything defined by a grammar.

Extensions

There is, of course, plenty of room for improvement. In the current program, the length of a generated string is limited to the length of a single string variable. It should be possible to generate into a string array rather than a single variable. Another possibility is special command characters to control output, such as a "#" character causing a skip to a new line. Additional features will suggest themselves with experience. □

A Modification of the Original

Guess My Animal

Mike Orlove

This program plays "Guess My Animal" similar to the program in *Basic Computer Games*. Its internal method of storing questions and answers is somewhat different from the original version. The original stored the questions and possible answers in a linked list simulated with a character string array. This version, by Michael Orlove, stores the information in a tree or program-like form in a string array. The animal-guessing clues are stored as a set of nested IF . . . THEN . . . ELSE structures with the appropriate questions and animal name answers embedded.

At the start of the program run, the computer asks if you are writing the program. If you answer YES, the computer allows you to enter the animal-guessing tree expressed in its simple language. If you answer NO, then it will load a previously saved data tape. Once the game is running, it will recognize a set of commands as given below:

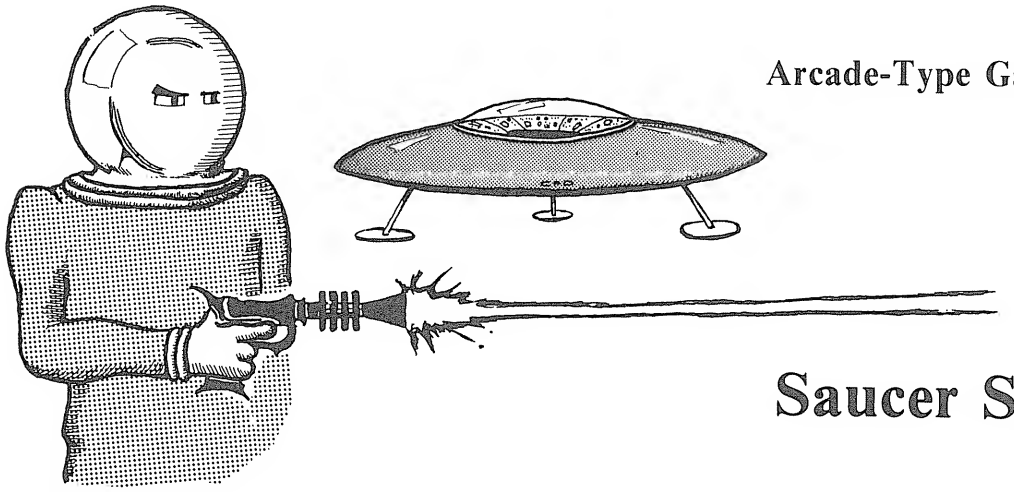
SAVE	Save animal-guessing information on data tape.
READ	Loads in previously saved data tape.
PRINT	Prints out the animal-guessing data tree.
MORE	Continues printing tree after stopping at end of screen.
FINISH	Stops printing tree as an alternate to MORE.
REPLACE	Replaces one string with another throughout the tree.

Note that the tape I/O routines are not implemented in this program but are indicated by remark statements and may be filled in by the user. In entering the program, please enter the words THEN, ELSE, and IF in lines 730, 770, and 810 as *lower case*. On the TRS-80 this must be done by holding down the shift key. The purpose of this subterfuge is to prevent the user from bombing the system by entering an animal named IF, THEN, or ELSE by converting the case. □

Mike Orlove, 3195 Hull Ave., Bronx, NY 10467.

```
9 CLEAR 9000
10 DIM V$(900)
12 X7=200
20 CLS:INPUT"ARE YOU WRITING THIS PROGRAM?";A$
30 IFA$="NO"THEN830
40 V$(1)="THEN":FORI=2TOX7
50 INPUT V$(I)
60 IFV$(I)="OK"THEN80
70 NEXTI
80 CLS:PRINT"LET'S PLAY 20 QUESTIONS, I GUESS AN ANIMAL
90 PRINT:PRINT:PRINT
100 I3=0:E3=0
110 I=1
120 I=I+1
130 IFV$(I)="IF"THEN170
140 IFV$(I-1)="THEN"THEN 320
150 IFV$(I-1)="ELSE"THEN 320
160 GOTO120
170 I=I+1:FORI=LTOX7
180 J=I:IFV$(I) ="THEN"THEN200
190 PRINTV$(I):NEXTI
200 INPUT A$:IFA$="YES"THEN120
201 IF A$="PRINT"THEN 690
202 IFA$="REPLACE"THEN900
204 IFA$="LOAD"THEN830
210 IFA$="SAVE"THEN 850
220 FORI=JTOX7
230 IFV$(I)="THEN"THEN 250
240 GOTO270
250 I3=I3+1
260 GOTO300
270 IFV$(I)="ELSE"THEN290
280 GOTO300
290 E3=E3+1
300 IFI3=E3THEN120
310 NEXTI
320 PRINT"IT MUST BE A "V$(I)" RIGHT?"
335 INPUT A$:IFA$="YES"THEN80
340 PRINT:INPUT"OK I LOSE THIS TIME, WHAT IS THE NAME OF YOUR ANIMAL";A$
350 GOSUB 710
360 PRINT:PRINT"WHAT IS THE DIFFERENCE BETWEEN A ";:PRINT V$(J);:PRINT"
    AND A ";  :PRINT A$;
370 INPUT D$
380 PRINT:PRINT"TYPE IN AN APPROPRIATE QUESTION, WHICH WHEN ANSWERED YES"
390 PRINT"WILL INDICATE YOU ARE THINKING OF THE ";:PRINT A$
400 PRINT"WHEN YOU ARE FINISHED TYPE OK ON A LINE TO ITSELF."
420 I$="IF":I9=I
430 GOSUB620
440 S$=A$
```

```
449 I=I
450 I=L+1
460 INPUT A$
470 GOSUB710
480 I$=A$
490 IFI$="OK"THEN530
500 I9=L
510 GOSUB620
520 GOTO450
530 I=L-1:I9=L+1
540 I$="THEN"
550 GOSUB620
560 A$=S$
570 I9=L+2:I$=A$
580 GOSUB620
590 I9=L+3:I$="ELSE"
600 GOSUB620
610 GOTO80
620 M=I9-1
621 M=M+1
630 IFV$(M)="OK"THEN650
640 GOTO621
650 IF M=X7THEN 680
660 FOR N=M+1TOI9STEP-1:V$(N)=V$(N-1):NEXTN
670 V$(I9)=I$:RETURN
680 REM USE TELETYPE
690 X8=2
691 FORH=X8-1TOX8+14:PRINT V$(H):NEXTH
692 X8=X8+15
693 INPUTX$
694 IFX$<>"FINISH"THEN691
695 GOTO80
700 REM BACK TO SCREEN
710 IFA$="THEN"THEN730
720 GOTO750
730 A$="then"
740 RETURN
750 IFA$="ELSE"THEN 770
760 GOTO790
770 A$="else"
780 RETURN
790 IFA$="IF"THEN810
800 RETURN
810 A$="if"
820 RETURN
830 INPUT#-1,M:M=M-1:FORI=1TOM+1:INPUT#-1,V$(I):NEXTI
840 GOTO80
850 PRINT#-1,M+1:FORI=1TOM+1:PRINT#-1,V$(I):NEXTI
860 REM SAVE V$ ON TAPE
870 GOTO80
900 INPUT A$,B$
910 I=0
920 I=I+1
930 IFV$(I)="OK"THEN 80
940 IFA$<>V$(I)THEN920
950 V$(I)=B$
960 GOTO80
```



Arcade-Type Game Teaches Math

Saucer Shoot

Ralph White

The accompanying *Saucer Shoot* program, written in Radio Shack Level II Basic, demonstrates that those of us not experienced in machine language need not be limited to static displays. We too can provide movement and animation to bring our screens to life. It also dispels the notion that an educational program must be dull.

The program mixes the environment of a TV arcade game with percentage problems. The result, which is used in a junior high classroom, helps make percentage drill problems palatable. Even the most unmotivated child likes to play games. By camouflaging the purpose of the program, perhaps we can convince an otherwise unwilling student to work problems that would normally be untouched.

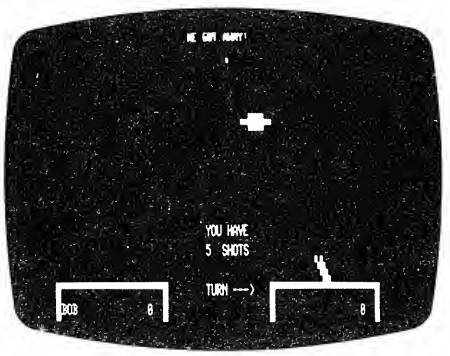
Saucer Shoot is written for two players to compete against each other. The computer is a neutral nonparticipant. It randomly thinks of problems to befuddle the players and does the housekeeping chores required to referee the game.

The program combines competition, animation and CAI to achieve its purpose—which is that the players will become better aquainted with percentage problems without experiencing the drudgery.

The computer alternately presents the players with problems. Each player may choose any one of five levels of difficulty for each turn. If the question is answered correctly, the player gets to shoot at flying saucers. The number of shots is directly related to the level of difficulty chosen. A player gets one shot for choosing a level 1 (easiest) question, two shots for a level 2 question, on up to five shots for successfully answering a level 5 (hardest) question. If the question is not answered correctly, the correct answer will be displayed and then play will pass to the other player. The first player to shoot down ten flying saucers wins.

After a player answers a question, a little head will rise up from the proper gun emplacement, turn, look at the answer, and nod "yes" or "no" in response to whether the answer is right or not. If the answer is correct then the player is allowed to shoot at flying

Ralph White, 529 South Vermont, Columbus, KS 66725.



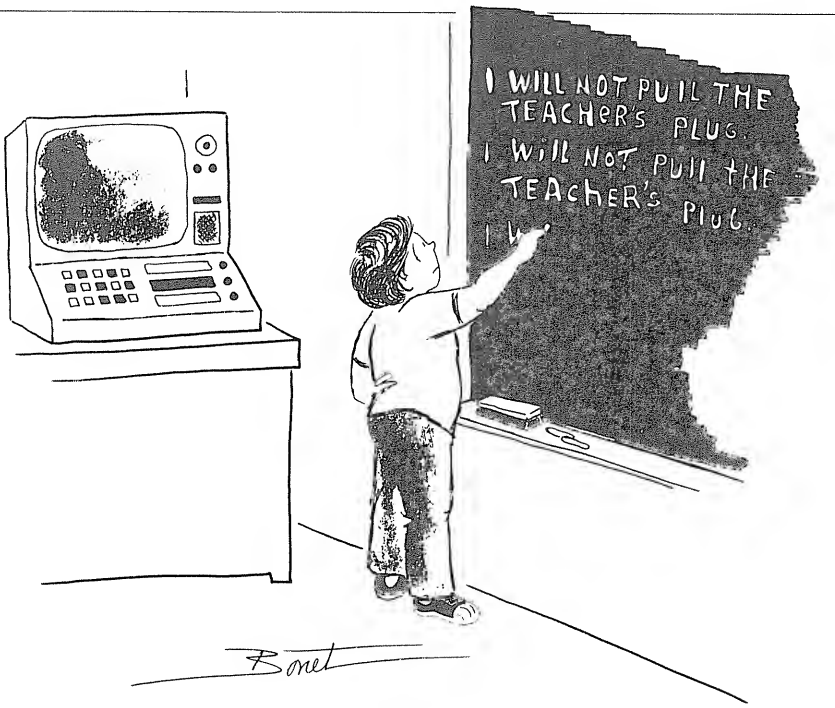
saucers that move across the screen. The saucers fly at various heights, and may appear on either side of the screen. Only one shot at each saucer is allowed. When a saucer is hit, it explodes and shatters into pieces.

Animation is provided by a series of strings that are comprised of graphics characters. There are 28 different strings that are needed to supply movement of the heads and fourteen strings that create the flying saucers and explosions. This procedure of printing strings of graphics characters rather than employing the SET and RESET commands increases the speed of the graphics to the point that smooth animation is possible. □

```
 10
10 CLS:CLEAR(400):DL=20
12 DIMH$(30),S$(15)
15 BL$(1)="    ":BL$(2)="                                        "
20 P(1)=919:P(2)=929:A$(1)="<---":A$(2)="--->"
25 TL$="################"
50 PRINTCHR$(23):PRINT:PRINT:PRINT:PRINTTL$;TL$
52 PRINTTAB(10)"SAUCER SHOOT":PRINT:PRINTTAB(11)"PERCENTAGE":PRINT
54 PRINTTL$;TL$
100 GOSUB10000
200 CLS:PRINTTAB(20)"S A U C E R    S H O O T":PRINTTAB(26)"INSTRUCTIONS":PRINT
210 PRINT"THE PROBLEMS IN THIS PROGRAM ASK YOU TO FIND THE PERCENT OF A"
215 PRINT"NUMBER.  THERE ARE 5 LEVELS OF DIFFICULTY.  YOU CHOOSE THE ONE"
220 PRINT"YOU WISH (1-EASIEST : 5-HARDEST).  IF YOU ANSWER THE QUESTION"
225 PRINT"CORRECTLY, YOU GET TO SHOOT AT FLYING SAUCERS.  THE NUMBER OF"
230 PRINT"SHOTS YOU GET DEPENDS ON WHICH LEVEL OF DIFFICULTY IS CHOSEN."
235 PRINT"LEVEL 1 ALLOWS YOU ONE SHOT, LEVEL 2 GETS TWO SHOTS AND SO ON.
240 PRINT"THE FIRST PLAYER TO SCORE TEN HITS IS THE WINNER."
241 PRINT"TO SHOOT AT THE SPACE SHIPS PRESS THE SPACE BAR.  YOU GET ONE"
242 PRINT"SHOT AT EACH SAUCER.":PRINT
245 INPUT"WHAT IS THE NAME OF THE FIRST PLAYER ";N$(1)
246 IFLEN(N$(1))<10GOTO250
247 PRINT"NAME TOO LONG.  10 LETTERS OR LESS, PLEASE":GOTO245
250 INPUT"WHAT IS THE NAME OF THE SECOND PLAYER ";N$(2)
251 IFLEN(N$(2))<10GOTO260
252 PRINT"NAME TOO LONG.  10 LETTERS OR LESS, PLEASE":GOTO250
260 CLS
500 FORI=0TO41:SET(0+I,42):SET(80+I,42):NEXT
510 FORI=0TO4:SET(0,43+I):SET(1,43+I):SET(40,43+I):SET(41,43+I):SET(80,43+I):SET
(81,43+I):SET(120,43+I):SET(121,43+I):NEXT
700 P=RND(2)
710 PRINT@961,N$(1);:PRINT@1001,N$(2);
800 PRINT@924,"TURN";
805 PRINT@976,S(1);:PRINT@1016,S(2);
810 PRINT@P(P),A$(P);
820 PRINT@349,"CHOOSE":PRINT@411,"A LEVEL OF":PRINT@475,"DIFFICULTY";:PRINT@540,
"(1 -- 5)
900 A$=INKEY$:IFA$=""GOTO900
910 VL=ASC(A$):IFVL<49ORVL>53GOTO900
920 D=VAL(A$)
950 PRINT@349,S$(14);:PRINT@411,S$(14);:PRINT@475,S$(14);:PRINT@540,S$(14);
1000 ONDGOTO1010,1020,1030,1040,1050
1010 PC=RND(3)*25:NU=RND(20)*20:GOTO1500
1020 PC=RND(4)*20:NU=RND(50)*10:GOTO1500
1030 PC=RND(9)*10:NU=RND(50)*10:GOTO1500
1040 PC=RND(19)*5:NU=RND(20)*20:GOTO1500
1050 PC=RND(24)*4:NU=RND(16)*25
1500 AN=NU*PC/100
1510 PRINT@670,"WHAT  IS":PRINT@729," ";:PRINTPC;"% OF ";NU;
1515 PRINT@794," ";:INPUTEX$
1516 PRINT@200," "
1518 IFLEN(EX$)=0GOTO1515
1520 NV=VAL(EX$):FV=0
1530 FORI=1TOLEN(EX$)
1535 EV=ASC(MID$(EX$,I,1))
1540 IFEV<48OREV>57THENFV=1
1560 IFFV=0GOTO1580
1570 PRINT@794,"         ";:GOTO1515
1580 HT=782:HB=846:IFP=1GOTO1600
1590 HT=809:HB=873
1600 FORI=1TO3:PRINT@HB,H$(I);:FORTM=1TODL:NEXT:NEXT
1610 FORI=1TO2:PRINT@HT,H$(I);:PRINT@HB,H$(I+3);:FORTM=1TODL:NEXT:NEXT
1620 PRINT@HT,H$(6);:PRINT@HB,H$(7);:FORTM=1TODL:NEXT
1630 FORTM=1TODL:NEXT
1650 ONPGOTO1800,1900
1800 FORI=9TO15STEP2:PRINT@782,H$(I);:PRINT@846,H$(I+1);:FORTM=1TODL:NEXT:NEXT
1820 FORTM=1TO300:NEXT
1825 FORI=13TO9STEP-2
1830 PRINT@782,H$(I);:PRINT@846,H$(I+1);:FORTM=1TODL:NEXT
1835 NEXT
1840 PRINT@782,H$(6);:PRINT@846,H$(7);:FORTM=1TODL:NEXT
1890 GOTO2000
1900 FORI=17TO23STEP2:PRINT@809,H$(I);:PRINT@873,H$(I+1);:FORTM=1TODL:NEXT:NEXT
1910 FORTM=1TO300:NEXT
1920 FORI=21TO17STEP-2
1925 PRINT@809,H$(I);:PRINT@873,H$(I+1);:FORTM=1TODL:NEXT
1930 NEXT
1940 PRINT@809,H$(6);:PRINT@873,H$(7);:FORTM=1TODL:NEXT
2000 IFNV=ANGOTO2100
2005 FORI=1TO3
2010 PRINT@HT,H$(9);:PRINT@HB,H$(10);:FORTM=1TODL*2:NEXT
2020 PRINT@HT,H$(6);:PRINT@HB,H$(7);:FORTM=1TODL*2:NEXT
```

```
2030 PRINT@HT,H$(17);:PRINT@HB,H$(18);:FORTM=1TO2*DL:NEXT
2040 PRINT@HT,H$(6);:PRINT@HB,H$(7);:FORTM=1TO2*DL:NEXT
2050 NEXT
2060 FORI=2TO1STEP-1:PRINT@HT,H$(I);:PRINT@HB,H$(I+3);:FORTM=1TODL:NEXT:NEXT
2062 PRINT@HT,H$(8);
2065 FORI=3TO1STEP-1:PRINT@HB,H$(I);:FORTM=1TODL:NEXT:NEXT
2070 PRINT@HB,H$(8);:FORTM=1TODL:NEXT
2080 GOTO2200
2100 FORI=1TO3
2110 PRINT@HT,H$(25);:PRINT@HB,H$(26);:FORTM=1TO2*DL:NEXT
2120 PRINT@HT,H$(6);:PRINT@HB,H$(7);:FORTM=1TO2*DL:NEXT
2130 PRINT@HT,H$(27);:PRINT@HB,H$(28);:FORTM=1TO2*DL:NEXT
2140 PRINT@HT,H$(6);:PRINT@HB,H$(7);:FORTM=1TO2*DL:NEXT
2150 NEXT
2160 FORI=2TO1STEP-1:PRINT@HT,H$(I);:PRINT@HB,H$(I+3);:FORTM=1TODL:NEXT
2162 PRINT@HT,H$(8);
2165 FORI=3TO1STEP-1:PRINT@HB,H$(I);:FORTM=1TODL:NEXT:NEXT
2170 PRINT@HB,H$(8);:FORTM=1TODL:NEXT
2180 PRINT@670,S$(14);:PRINT@729,S$(14);:PRINT@786,S$(14);
2190 GOTO2300
2200 PRINT@786,S$(14);:PRINT@787,"THE CORRECT ANSWER IS ";AN;
2210 FORTM=1TO3000:NEXT
2220 PRINT@786,BL$(2);:PRINT@P(P),BL$(1);
2230 PRINT@670,BL$(2);:PRINT@729,BL$(2);
2240 IFP=2GOTO2270
2250 P=P+1:GOTO800
2270 P=P-1:GOTO800
2300 PRINT@724,BL$(2);:PRINT@729,BL$(2);:PRINT@786,BL$(2);
2310 PRINT@732,"YOU HAVE";:PRINT@795,D;" SHOTS";
2390 FORTM=1TO1500:NEXT
2400 H=RND(6):SH=64*H:EN=SH+56
2410 DR=RND(2):IFDR=2GOTO2430
2420 BG=SH:ST=EN:IN=1:GOTO2500
2430 BG=EN:ST=SH:IN=-1
2500 IFP=2GOTO2800
2600 FORI=0TO3:FORJ=0TO2:SET(19+I+J,41-I):NEXT:NEXT
2601 X=26:Y=35
2605 PRINT@BG,S$(1);
2610 A$=INKEY$:IFA$=""GOTO2630
2620 IFASC(A$)=32GOTO2700
2630 BG=BG+IN:IFBG=STGOTO3100
2640 GOTO2600
2700 BG=BG+IN:IFBG=STGOTO3100
2710 PRINT@BG,S$(1);
2720 RESET(X-3,Y+3):SET(X,Y)
2730 CK=POINT(X+3,Y-3):IFCK=-1GOTO3000
2740 X=X+3:Y=Y-3:IFY<5GOTO3100
2750 GOTO2700
2800 FORI=0TO3:FORJ=0TO2:SET(100-I+J,41-I):NEXT:NEXT
2801 X=95:Y=35
2805 PRINT@BG,S$(1);
2810 A$=INKEY$:IFA$=""GOTO2830
2820 IFASC(A$)=32GOTO2900
2830 BG=BG+IN:IFBG=STGOTO3100
2840 GOTO2800
2900 BG=BG+IN:IFBG=STGOTO3100
2910 PRINT@BG,S$(1);
2920 RESET(X+3,Y+3):SET(X,Y)
2930 CK=POINT(X-3,Y-3):IFCK=-1GOTO3000
2940 X=X-3:Y=Y-3:IFY<5GOTO3100
2950 GOTO2900
3000 PRINT@BG-64,S$(2);:PRINT@BG,S$(3);:PRINT@BG+64,S$(4);:FORTM=1TO2*DL:NEXT
3010 PRINT@BG-64,S$(5);:PRINT@BG,S$(6);:PRINT@BG+64,S$(7);:FORTM=1TODL:NEXT
3020 PRINT@BG-1,S$(8);:PRINT@BG+63,S$(9);:FORTM=1TODL+10:NEXT
3030 PRINT@BG-2,S$(10);:PRINT@BG+62,S$(11);:FORTM=1TODL:NEXT
3040 PRINT@BG-3,S$(12);:PRINT@BG+61,S$(13);:FORTM=1TODL:NEXT
3045 PRINT@BG-3,S$(14):PRINT@BG+61,S$(14);
3050 S(P)=S(P)+1:IFS(P)>9GOTO3200
3055 FORTM=1TO800:NEXT
3060 D=D-1:IFD>0GOTO2310
3070 GOTO2220
3100 PRINT@25,"HE GOT AWAY!";
3110 FORTM=1TO1800:NEXT
3115 PRINT@BG,S$(5);:PRINT@25,S$(14);:RESET(X,Y)
3117 PRINT@89,S$(14);:PRINT@151,S$(14);:PRINT@215,S$(14);
3120 GOTO3060
3200 CLS:PRINTCHR$(23):PRINT
3210 PRINTTAB(5);N$(P)
3220 PRINT"IS THE WINNER!":PRINT:PRINT
3230 PRINTN$(P);" SAVED THE WORLD"
3240 PRINT"FROM THE INVADERS FROM MARS!"
3250 END
3330 END
```

```
10000 FORH=1TO28:FORI=1TO6:READCH:H$(H)=H$(H)+CHR$(CH):NEXT:NEXT
10010 FORS=1TO7:FORI=1TO8:READCH:S$(S)=S$(S)+CHR$(CH):NEXT:NEXT
10020 FORS=8TO9:FORI=1TO10:READCH:S$(S)=S$(S)+CHR$(CH):NEXT:NEXT
10030 FORS=10TO11:FORI=1TO12:READCH:S$(S)=S$(S)+CHR$(CH):NEXT:NEXT
10040 FORS=12TO13:FORI=1TO14:READCH:S$(S)=S$(S)+CHR$(CH):NEXT:NEXT
10050 S$(14)=STRING$(14,CHR$(128))
10060 RETURN
11000 DATA128,176,176,176,176,128
11001 DATA128,188,188,188,188,128
11002 DATA160,191,175,159,191,144
11003 DATA136,191,187,183,191,132
11004 DATA130,191,158,173,191,129
11005 DATA160,191,175,159,191,144
11006 DATA128,143,183,187,143,128
11007 DATA128,128,128,128,128,128
11008 DATA160,191,159,191,175,144
11009 DATA128,143,191,179,143,128
11010 DATA128,191,191,175,159,128
11011 DATA128,143,191,183,139,128
11012 DATA128,191,191,159,191,128
11013 DATA128,143,191,191,139,128
11014 DATA128,191,191,191,175,128
11015 DATA128,143,191,191,143,129
11016 DATA160,159,191,175,191,144
11017 DATA128,143,179,191,143,128
11018 DATA128,175,159,191,191,128
11019 DATA128,135,187,191,143,128
11020 DATA128,191,175,191,191,128
11021 DATA128,135,191,191,143,128
11022 DATA128,159,191,191,191,128
11023 DATA130,143,191,191,143,128
11024 DATA160,191,187,183,191,144
11025 DATA128,143,189,190,143,128
11026 DATA160,191,191,191,191,144
11027 DATA128,143,158,173,143,128
11028 DATA128,140,174,191,191,157,140,128
11029 DATA128,160,176,176,176,176,144,128
11030 DATA174,179,145,191,191,162,179,157
11031 DATA128,130,131,131,131,131,129,128
11032 DATA128,128,128,128,128,128,128,128
11033 DATA140,136,133,190,189,138,132,140
11034 DATA128,128,129,129,130,130,128,128
11035 DATA176,160,133,132,162,145,136,138,144,176
11036 DATA128,136,136,129,129,130,130,132,132,128
11037 DATA128,128,152,160,128,132,136,128,144,164,128,128
11038 DATA131,146,160,136,162,191,191,145,132,144,161,131
11039 DATA128,128,160,128,128,128,128,128,128,128,128,144,128,128
11040 DATA130,153,161,130,152,160,129,130,144,164,129,146,166,129
```



Reading Practice With the TRS-80 Voice Synthesizer

The First "R"

John F. Rogers

"Reading, Riting, and Rithmetic" — the "Three R's." Without the first "R," a person can have much difficulty in this Age of Information. If a child is having reading troubles, perhaps a little extra drill would be appropriate. The program "Reading Practice with the TRS-80 Voice Synthesizer" may be suitable for such extra drill.

The Radio Shack TRS-80 Voice Synthesizer gives your microcomputer the ability to speak. (See "Phonetically Speaking," June 1979, *Creative Computing.)* This program utilizes that ability to give a student drill in reading and saying the basic words that should be known at a given age.

The Dolch Basic Sight Word List contains 212 words that the average third grade child should recognize. (You should be able to get a copy of the Dolch List from your nearby primary school.) By listing those words which comprise 50% to 75% of all reading matter in a DATA statement along with their Voice Synthesizer phonetic spellings, you can give a student drill in retaining those words in his reading and speaking vocabularies.

Program REMarks

The program is written in Radio Shack Level II Basic, but it can be adapted to Level I or to other Basics.

Lines 20-100 give the student instructions in how the drill will proceed. Both written and spoken instructions are given throughout the program.

Lines 110-155 present the words that are to be spoken by the student. Subroutine 500 actually writes the word on the screen and gives the correct pronunciation. For emphasis, the word blinks inside a graphics rectangle.

In lines 170-200, the computer asks the student to repeat the word once more along with the computer.

Subroutine 400 outlines the screen — a "dress-up" to set off the written instructions.

Subroutines 1000 and 1100 send the phonetic spellings of words to be spoken to the Voice Synthesizer.

Subroutine 2000 is used by the programmer to check the pronunciations of words to be listed in the DATA statements to ensure clarity and correctness. (Just enter RUN 2000.)

Lines 300 and following contains the DATA statement listings of the words to be read and spoken. The numbers are important for correct execution of the program. □

John F. Rogers, 600 Seventh St., Morgan City, LA 70380.

```
1 REM ***       READING PRACTICE       ***
2 REM ***      WITH THE TRS-80         ***
3 REM ***     VOICE SYNTHESIZER        ***
4 REM ***       PROGRAM BY             ***
5 REM ***     JOHN F.  ROGERS          ***
6 REM ***     600 SEVENTH ST.          ***
7 REM ***     MORGAN CITY,             ***
8 REM ***     LOUISIANA 70380          ***
9 REM ***                              ***
10 CLS:GOSUB 400
20 PRINT@150,"H E L L O -";:VO$="H38L8[U":GOSUB 1000:FOR K=0 TO 600:NEXT
30 PRINT@270,"TODAY YOU WILL PRACTICE";:VO$="T<UD@*&Y'UW!ILLPR99KT!IS":GOSUB 1000
35 FOR K=0 TO 960:NEXT
40 PRINT@404,"SAYING WORDS I SHOW YOU.";:VO$="S@@&E+W/RDZ;5#&>>OOWY'U":GOSUB 1000
50 FOR I=0 TO 1500:NEXT
60 CLS:GOSUB 400
70 PRINT@140,"I WILL FLASH A WORD ON THE SCREEN,";
75 VO$=";5#&W!ILLFL79>>66W/RD":GOSUB 1000:FOR K=0 TO 800:NEXT
80 VO$=";ANN<67SKR.ENN":GOSUB 1000:FOR K=0 TO 600:NEXT
85 PRINT@280,"THEN YOU WILL SAY IT ";:VO$="<<35NY'UW!ILLS@@&!IT":GOSUB 1000:FOR K=0 TO 1200:NEXT
90 PRINT@404,"THEN  I  WILL SAY THE WORD.";:VO$="<<35N0;5#& W!ILLS@@&<67W/RD":GOSUB 1000
95 FOR K=0 TO 1400:NEXT:PRINT@520,"FINALLY, WE'LL SAY THE WORD TOGETHER.";
100 VO$="F;#&N8LE& W &8LS@@&<67W/RD":GOSUB 1000:FOR K=0 TO 1040:NEXT:VO$="T<UG35<</R":GOSUB 1000
105 FOR K=0 TO 1500:NEXT
110 CLS:GOSUB 400:PRINT@217,"R E A D Y ?";:VO$="R345DE&":GOSUB 1000:FOR K=0 TO 1000:NEXT
120 CLS:GOSUB 400
130 PRINT@140,"THE FIRST WORD IS...";:VO$="<.F/RSTW/RD!IZZ":GOSUB 1000:FOR K=0 TO 600:NEXT
135 READ Z,S$,WD$:GOSUB 500:GOTO 160
140 CLS:GOSUB 400:ON ERROR GOTO 900:READ Z
150 PRINT@140,"THE NEXT WORD IS...";:VO$="<.N35KSTW/RD!IZZ":GOSUB 1000
155 READ S$,WD$:GOSUB 500
160 CLS:GOSUB 400
170 PRINT@140,"DID YOU SAY THE WORD CORRECTLY?";
175 VO$="D!IDDY'US@@&<67W/RDKOR45KTLE&":GOSUB 1000:FOR K=0 TO 1200:NEXT
180 PRINT@265,"LET'S SAY IT TOGETHER:";:VO$="L35TS0S@@&!IT0T<UG35<</R":GOSUB 1000
185 FOR K=0 TO 1000:NEXT
190 PRINT@404,"R E A D Y ?";:VO$="R345DE&":GOSUB 1000:FOR K=0 TO 600:NEXT
200 PRINT@530,"THE WORD IS:  ";S$;:VO$="<67W//RD!IZZ":GOSUB 1000:GOSUB 1100
210 FOR I=0 TO 1800:NEXT:GOTO 140
300 DATA 1,CLEAN,KLL.ENN,2,HURT,HH//RT,3,GREEN,GRR.ENN,4,LAUGH,LL99FF
400 FOR I=0 TO 62 STEP 2:PRINT@I,"#";:NEXT
405 FOR I=64 TO 832 STEP 64:PRINT@I,"#";:PRINT@I+62,"#";:NEXT
410 FOR I=896 TO 958 STEP 2:PRINT@I,"#";:NEXT
420 RETURN
500 FOR J=35 TO 94:SET(J,11):SET(J,22):NEXT
505 FOR J=11 TO 22:SET(35,J):SET(94,J):NEXT
510 FOR I=0 TO 7:PRINT@348," ";:FOR J=0 TO 100:NEXT J:PRINT@348,S$;:FOR J=0 TO 600:NEXT
520 PRINT@590,"THE WORD IS PRONOUNCED...";:VO$="<67W/RD!IZZPRON;UNST":GOSUB 1000
525 GOSUB 1100:FOR I=0 TO 1600:NEXT:GOSUB 1100:FOR J=0 TO 1500:NEXT
```

```
530 RETURN
900 CLS:GOSUB 400
910 PRINT@130,"THE WORD LIST HAS ENDED.";:VO$="<77W/RDL!IST0H99Z035ND4D":GOSUB 1000
915 FOR K=0 TO 1000:NEXT
920 PRINT@260,"PLEASE CALL THE INSTRUCTOR.";:VO$="PL.EZK122LL0<.INSTR67KT/":GOSUB 1000
930 RESUME 950
950 FOR I=0 TO 1500:NEXT
960 PRINT@390,"THE DATA LIST OF WORDS";:PRINT@479,"HAS BEEN DEPLETED.";
970 PRINT@576,"PRESS 'BREAK' KEY TO GET CONTROL OF THE COMPUTER:";:GOTO 970
1000 POKE 16383,63:POKE 16383,32
1010 FOR VX=1 TO LEN(VO$)
1020 POKE 16383,ASC(MID$(VO$,VX,1))
1030 NEXT VX
1040 POKE 16383,32:POKE 16383,63:POKE 16383,32
1050 RETURN
1100 POKE 16383,63:POKE 16383,32
1110 FOR VX=1 TO LEN(WD$)
1120 POKE 16383,ASC(MID$(WD$,VX,1))
1130 NEXT VX
1140 POKE 16383,32:POKE 16383,63:POKE 16383,32
1150 RETURN
2000 CLS
2010 PRINT@0,"THIS IS THE PRONUNCIATION TESTING ROUTINE:";
2020 PRINT@128,VO$
2030 PRINT@192,"ENTER PHONEMES..."
2040 INPUT VO$:GOSUB 1000
2050 GOTO 2000
```

Can you recognize a word in a minute?

Mind Exerciser

Jason Woolf
Charles Hemminger

Have you an agile memory? Are you sober? Try this one on your friends. It's a simple game to measure your powers of recall.

Mind Exerciser displays words at random points on the screen. After a short time you are asked to remember as many of the words as possible.

This program should be adaptable to any Basic which allows positioning of the cursor. This is the version we have worked out for a TRS-80.

The first section of the program initializes everything and reads the words to be used from the DATA statements at the bottom of the program. You can choose up to 199 words up to 12 letters long, but be sure to conclude with "-1".

In Lines 410-520 the cursor positions are chosen. The object is to place them below the title message and so they do not overlap each other or wrap around the screen.

The next seven Lines shuffle the order of the words in the data set. The rest of this simple program is quite straight forward. If you can display upper and lower case then Lines 850-900 can be omitted. The typical TRS-80 enters the ASCII value when an upper-case letter is typed and displays it accordingly. Lower-case entries are taken into the buffer as ASCII lower case but displayed as upper case. This can make for confusing errors by whomever is trying to remember the words. Therefore the program assumes that all DATA is entered with no capitals. This routine can match the stored words with any entries. □

Jason Woolf, Charles Hemminger, 20 North Harrison Ave., Northampton, MA 01060.

```
10  REM MIND EXERCISER
20  REM VERSION 1.0
30  REM BY JASON WOLF & CHARLES HEMMINGER
40  CLS
50  CLEAR2000
60  DIM A$(200),S(200),B$(25),B(25)
70  N=0
80  N=N+1
90  READ A$(N)
100  IF A$(N)="-1" THEN 130
110  S(N)=N
120  GOTO 80
130  N=N-1
140  REM INSTRUCTIONS
150  PRINT @20,"MEMORY EXERCISER"
160  PRINT
170  PRINT "DO YOU NEED INSTRUCTIONS  (YES OR NO)  ";
180  INPUT Q$
190  IF LEFT$(Q$,1)<>"Y" THEN 310
200  PRINT
210  PRINT "     THIS IS AN EXERCISE OF YOUR ABILITY TO CONCENTRATE."
220  PRINT "YOU WILL BE REQUESTED TO CHOOSE THE DENSITY OF WORDS TO"
230  PRINT "BE DISPLAYED ON THE SCREEN.  THEY WILL BE SHOWN FOR A SHORT"
240  PRINT "PERIOD OF TIME WHICH YOU HAVE ALSO CHOSEN.  YOU MUST THEN"
250  PRINT "TRY AND REMEMBER AS MANY AS POSSIBLE."
260  PRINT
270  PRINT "YOU SHOULD TRY AT LEAST 3 EXERCISES TO OBTAIN AN AVERAGE."
280  PRINT
290  PRINT "GOOD LUCK !"
300  PRINT
310  REM SET DENSITY
320  PRINT "DENSITY  (1-5) ";
330  INPUT D
340  D=INT(D)
350  IF D<1 OR D>5 THEN 320
360  D=D*5
370  PRINT "SPEED  (1-30) ";
380  INPUT S
390  S=INT(S)
```

```
400  IF S<1 OR S>30 THEN 370
410  REM RANDOMIZE PRINTING
420  C=0
430  FOR I=1 TO D
440    R=128+RND(883) : REM WANT A NUMBER FROM 128 TO 1023 - 12
450    IF R-INT(R/64)*64>51 THEN 440
460    IF I=1 THEN 500
470    FOR J=1 TO C
480      IF R>B(J)-12 AND R<B(J)+12 THEN 440
490      NEXT J
500    B(I)=R
510    C=C+1
520    NEXT I
530  REM RANDOMIZE DATA WORDS
540  FOR I=N TO 2 STEP -1
550    K=INT(RND(0)*N+1)
560    T=S(I)
570    S(I)=S(K)
580    S(K)=T
590    NEXT I
600  REM DISPLAY TEST
610  CLS
620  PRINT @20,"MEMORY EXERCISER"
630  FOR I=1 TO D
640    PRINT @B(I),A$(S(I))
650    NEXT I
660  REM DELAY
670  FOR I=1 TO S
680    FOR J=1 TO 600
690      NEXT J
700    NEXT I
710  REM ANSWER SHEET
720  C=0
730  CLS
740  FOR I=1 TO D
750    IF I/11<>INT(I/11) THEN 790
760    FOR J=1 TO 600
770      NEXT J
780    CLS
790    PRINT @20,"MEMORY EXERCISER"
800    PRINT @128,"HOW MANY OBJECTS CAN YOU RECALL"
810    M=I-INT(I/11)*11
820    PRINT @256+M*64,"OBJECT # ";I;" ";
830    INPUT T$
840    PRINT @306+M*64,"";
850    REM MAKE SURE NO UPPER CASE CHARACTERS
860    T=LEN(T$)
870    FOR J=1 TO T
880      K=ASC(MID$(T$,J,1)
890      IF K>96 THEN T$=LEFT$(T$,J-1)+CHR$(K-32)+RIGHT$(T$,T-J)
900      NEXT J
910    REM DID HE GET IT RIGHT
920    FOR L=1 TO D
930      IF A$(S(L))<>T$ THEN 1010
940      IF L=1 THEN 1040
950      FOR K=1 TO C
960        IF B$(K)<>T$ THEN 990
970        PRINT "REPEAT"
980        GOTO 1070
990        NEXT K
1000      GOTO 1040
1010      NEXT L
1020    PRINT "SORRY"
1030    GOTO 1070
1040    PRINT "CORRECT"
1050    C=C+1
1060    B$(C)=T$
1070    NEXT I
1080 PRINT "YOU HAD ";C;" OUT OF";D;" CORRECT."
1090 U=U+C
1100 V=V+D
1110 PRINT "WOULD YOU LIKE TO TRY AGAIN  (YES OR NO) ";
1120 INPUT Q$
1130 IF LEFT$(Q$,1)="Y" THEN 420
1140 IF LEFT$(Q$,1)<>"N" THEN 1110
1150 PRINT "YOUR FINAL SCORE WAS";INT(U/V*100);"% CORRECT."
1160 END
1170 REM STORE DATA WORDS OF YOUR CHOSING
1180 DATA DOG,CAT,ELEPHANT,ZEBRA,COW,PIG,SHEEP,DUCK,CHICKEN,RAT
1190 DATA HOUSE,TREE,ROSE,DAFFODIL,BIRD,ROBIN,BLUEJAY,CAR,GARAGE
1200 DATA PICTURE,PENCIL,PEN,PAPER,TYPEWRITER,DIPLOMA,WRENCH,TACK
1210 DATA LIGHT,THIMBLE,SPOOL,WIRE,SNAKE,BUSH,FLOWER,CHAIN
1220 REM LAST DATA WORD MUST BE -1
1230 DATA -1
```

How to Keep Score

Track Meet

Clinton Morey

"Track Meet" is a program designed to simplify the task of keeping score at a track meet.

A track meet is an exciting activity. Athletes competing in many different events, brightly colored uniforms, men and women striving to improve their performances to achieve victory.

A track meet is exciting—it is also a highly organized activity (at least it should be). The spectators and athletes concerned only with the event of the moment usually pay little attention to the organizational aspects of a meet. Recording winners of events, distributing ribbons or medals, keeping track of team scores, are just a few of the many details that go into a successful track meet. And that's where the computer can help. With its willingness to aid the inexperienced and its ability to perform calculations quickly and accurately, the computer is a friend and helper of the often overworked (and seldom noticed) people who sit at the official scorer's table.

The Program

The program "Track Meet," was written to meet the needs of a small rural school in Northeastern Montana but modifications will make it suitable for any athletic department. "Track Meet" allows individuals with little experience in track (typing would be helpful) to serve as scorers at a track meet. The program is designed for use on a TRS-80, Level II, 16K machine and a Centronics 730 (Line Printer II). It is easily adapted to other Basics.

At the prompting of the computer, the operator enters identifying information about the track meet (meet title, date, etc.). The only other keyboard entry required is to enter the results from the various events when the information is received (usually in barely legible scribblings by the official at the event).

Clinton Morey, Box 524, Peerless, MT 59253.

With the the data in the computer the operator may:

1) Display or print the results of a particular event. (Five points are awarded for first place, four for second place and so on.) Any scorer who has tried to record results while athletes and coaches are asking about events that took place 30 minutes earlier will appreciate the benefits of this part of the program.
2) Maintain a running total of team points. This allows scores to be posted and maintains a good level of interest in the team aspect of a track meet.
3) Print results of the entire meet. Hardcopy printouts can be provided to the coaches almost immediately after the meet. The results can also be sent to the local newspaper or the PR people.

Modifications

Several modifications could be made to this program to meet specific needs of individual schools. If ribbons are awarded at the meet an option could be added which would print self-adhesive labels to save someone the trouble of doing the work by hand. Someone would have to stick the labels on the backs of the ribbons, but that is a whole lot easier than writing them.

The events in this program are geared for a small rural school with limited track facilities. Events like the pole vault are usually not available to our athletes. To add or delete events merely change the READ-DATA statements. If the number of events is changed lines 40, 740, 1100, 1140, and 1320 will have to be adjusted to show the proper number of events. □

```
10 REM--TRACK MEET BY MOREY
20 CLS:POKE 16553,255:CLEAR 5000
30 DIM E$(5,3,22):DIM N$(3)
40 FOR X=1TO22:READ E$(0,0,X):NEXT X
45 FOR X=1TO3:READ N$(X):NEXTX
50 CLS
55 PRINT@340,"TRACK MEET":FORX=1TO1000:NEXTX:CLS
60 INPUT"WHAT IS THE TITLE OF TODAY'S MEET";MN$
70 INPUT"WHAT IS TODAY'S DATE";MD$
80 INPUT"HOW MANY TEAMS ARE ENTERED";T
90 DIM T$(T)
100 CLS:PRINT"ENTER THE NAME OF EACH TEAM."
110 FOR X=1TOT:PRINT"TEAM NO.";X;:INPUT T$(X):NEXT X
120 CLS:PRINTTAB(20)MN$:PRINTTAB(20)MD$:PRINT:PRINT
130 PRINTTAB(20)"TEAMS PARTICIPATING"
140 FOR X=1TOT:PRINTTAB(25) T$(X):NEXT X:PRINT
150 GOTO 1000
160 CLS
170 PRINTTAB(30)"MENU"
180 PRINTTAB(20)"1. ENTER EVENT RESULTS"
190 PRINTTAB(20)"2. DISPLAY EVENT RESULTS"
200 PRINTTAB(20)"3. CORRECT EVENT RESULTS"
210 PRINT"          ------------------------------------------------"
220 PRINTTAB(20)"4. DISPLAY TEAM SCORES"
230 PRINTTAB(20)"5. PRINT MEET STATS"
240 PRINT:PRINT
250 INPUT"WHICH DO YOU WISH TO USE (ENTER NUMBER)";Z
```

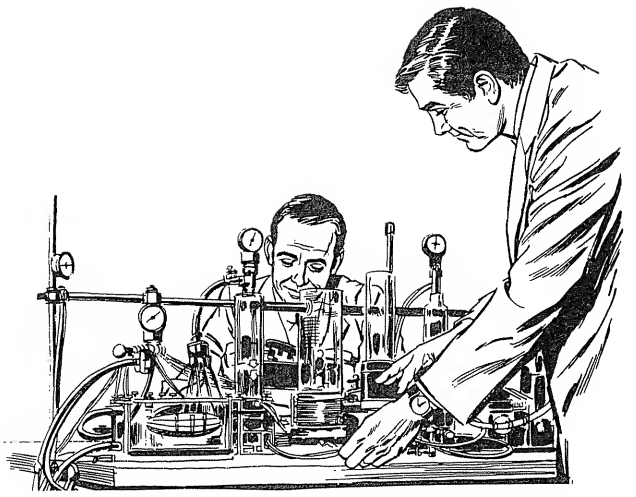
```
260 ON Z GOTO 300, 400, 500, 600, 700
270 IF Z>5 OR Z<1 GOTO 250
300 REM--ENTER EVENT RESULTS
310 GOSUB 1100
320 CLS:PRINTTAB(20) E$(0,0,M)
330 P=5
340 FOR A=1TO5
342    PRINT"PLACE:";A;
344    FOR B=1TO3
346       PRINT N$(B);":      ";
348       INPUT E$(A,B,M)
350    NEXT B
352    FOR Y=1TOT
354       IF T$(Y)=E$(A,3,M) THEN TP(Y)=TP(Y)+P:GOTO 358
356    NEXT Y
358    P=P-1
360 NEXT A
370 GOSUB 1200
380 GOTO 1000
400 REM--DISPLAY EVENT RESULTS
410 GOSUB 1100
420 GOSUB 1200
430 INPUT"DO YOU WANT A PRINT OUT (YES,NO)";Y$
440 IF LEFT$(Y$,1)="Y"THEN GOTO 460
450 GOSUB 1000
460 PRINT"PRESS ENTER WHEN THE PRINTER IS READY.":INPUTZ
465 LPRINTTAB(20) E$(0,0,M):LPRINT"PLACE";:LPRINTTAB(9) N$(1);:LPRINTTAB(30) N$(
2);:LPRINTTAB(50) N$(3):FOR A=1TO5:LPRINTA;:LPRINTTAB(9)E$(A,1,M);:LPRINTTAB(30)
E$(A,2,M);:LPRINTTAB(50)E$(A,3,M):NEXTA
470 GOTO 1000
500 REM--CORRECT EVENT RESULTS
510 GOSUB 1100
540 PRINT"TO MAKE CORRECTIONS FOR THIS EVENT YOU WILL BE ASKED EACH ENTRY QUESTI
ON AGAIN.  IF THE INFORMATION IS CORRECT, MERELY PRESS <ENTER>."
545 PRINT"   IF YOU WISH TO CHANGE ANY INFORMATION, TYPE IN THE NEW DATA WHEN RE
QUESTED THEN PRESS <ENTER>."
547 INPUT"PRESS <ENTER> TO CONTINUE";X:GOSUB 1200
549 PRINT"ENTER NEW DATA...."
550 P=5:FORA=1TO5:FORY=1TOT:IFT$(Y)=E$(A,3,M)THENTP(Y)=TP(Y)-P:GOTO554
552 NEXTY
554 PRINT"PLACE:";A;:FORB=1TO3:PRINTN$(B);":      ";:INPUTE$(A,B,M):NEXTB
556 FORY=1TOT:IFT$(Y)=E$(A,3,M)THENTP(Y)=TP(Y)+P:GOTO560
558 NEXT Y
560 P=P-1
565 NEXTA
570 GOSUB1200
580 GOTO1000
600 REM--DISPLAY TEAM SCORES
630 CLS:PRINTTAB(20) MN$:PRINTTAB(20) MD$:PRINT:PRINT:PRINTTAB(5)"TEAM";:PRINTTA
B(40)"POINTS"
640 FOR X=1TOT:PRINT T$(X);:PRINTTAB(42) TP(X):NEXT X
650 GOTO 1000
700 CLS:PRINT"PREPARE PRINTER.  PRESS <ENTER> WHEN READY.";:INPUTX
705 INPUT"HOW MANY COPIES DO YOU WANT PRINTED";K
707 FOR V=1 TO K
710 LPRINTTAB(20) MN$
720 LPRINTTAB(20) MD$
730 FOR X=1TO4:LPRINT" ":NEXTX
732 GOSUB 1300
735 LPRINTTAB(5)"TEAM";:LPRINTTAB(40)"POINTS"
737 FOR X=1 TO T:LPRINT T$(X);:LPRINTTAB(42) TP(X):NEXTX
739 FORX=1TO4:LPRINT" ":NEXTX
740 FOR M=1 TO 22
750   LPRINTTAB(20) E$(0,0,M)
760   LPRINT"PLACE";:LPRINTTAB(9) N$(1);:LPRINTTAB(30) N$(2);:LPRINTTAB(50) N$(3)
770   FOR A=1 TO 5:LPRINT A;:LPRINTTAB(9) E$(A,1,M);:LPRINTTAB(30) E$(A,2,M);:LP
RINTTAB(50) E$(A,3,M):NEXT A
780   LPRINT" "
790 NEXT M
800 NEXT V
1000 PRINT"PRESS <ENTER> TO SEE THE MENU.";
1010 INPUT Z
1020 GOTO 160
1100 CLS:FORX=1TO22:PRINTX;"    ";E$(0,0,X);:X=X+1:PRINTTAB(30)X;"    ";E$(0,0,X
):NEXT X
1110 FOR X=1TO60:PRINT"-";:NEXTX
1120 PRINT:PRINT
1130 INPUT"WHICH EVENT (ENTER NUMBER)";M
1140 IF M>22 GOTO 1130
1150 RETURN
1200 CLS:PRINTTAB(20) E$(0,0,M)
1210 PRINT"PLACE";:PRINTTAB(9) N$(1);:PRINTTAB(30) N$(2);:PRINTTAB(50) N$(3)
1220 FOR A=1TO5:PRINT A;:PRINTTAB(9) E$(A,1,M);:PRINTTAB(30) E$(A,2,M);:PRINTTAB
(50) E$(A,3,M):NEXT A
1230 RETURN
```

```
1300 CLS
1310 FORX=1TOT:F(X)=0:NEXTX:L=0
1320 FORM=1TO22
1340    FORA=1TO5:FORH=5TO1 STEP-1:FORX=1TOT:IF E$(A,3,M)=T$(X) THEN F(X)=F(X)+H:
NEXTX:NEXTH:NEXTA:NEXTM
1350 RETURN
2000 DATA 100 METERS--BOYS,100 METERS--GIRLS
2010 DATA 200 METERS--BOYS,200 METERS--GIRLS
2020 DATA 800 METERS--BOYS,800 METERS--GIRLS
2030 DATAMILE RUN--BOYS,MILE RUN--GIRLS
2040 DATA MILE RELAY--BOYS,MILE RELAY--GIRLS
2050 DATA JAVELIN--BOYS,JAVELIN--GIRLS
2060 DATA LONG JUMP--BOYS,LONG JUMP--GIRLS
2070 DATA HIGH JUMP--BOYS,HIGH JUMP--GIRLS
2080 DATA SHOT PUT--BOYS,SHOT PUT--GIRLS
2090 DATA DISCUS--BOYS,DISCUS--GIRLS
2100 DATA3000 METER RUN--BOYS,3000 METER RUN--GIRLS
2110 DATA NAME,TIME/DISTANCE,SCHOOL
3000 END
```

```
                    MOREY'S MARVELOUS MEET
                    APRIL  23  1981

     TEAM                                   POINTS
PEERLESS                                      19
SCOBEY                                        15
OUTLOOK                                        8

                      100 METERS--BOYS
PLACE     NAME                 TIME/DISTANCE        SCHOOL
  1       BILL BAILEY          10.0                 PEERLESS
  2       RON RUNNER           10.2                 OUTLOOK
  3       FRED FAST            10.5                 SCOBEY
  4       SAM SLOW             12.8                 PEERLESS
  5       TOM TANK             15.1                 OUTLOOK
                      100 METERS--GIRLS
PLACE     NAME                 TIME/DISTANCE        SCHOOL
  1       RONDA RUNNER         10.8                 SCOBEY
  2       SALLY SWIFT          11.3                 PEERLESS
  3       JOAN JALOPY          12.2                 PEERLESS
  4
  5
                      200 METERS--BOYS
PLACE     NAME                 TIME/DISTANCE        SCHOOL
  1       TOM TERRIFIC         24.8                 SCOBEY
  2       BILL BAILEY          25.4                 PEERLESS
  3       TOM TANK             26.0                 OUTLOOK
  4       FRED FAST            28.1                 SCOBEY
  5       SAM SLOW             34.9                 PEERLESS
                      200 METERS--GIRLS
PLACE     NAME                 TIME/DISTANCE        SCHOOL
  1
  2
  3
  4
  5
                      800 METERS--BOYS
PLACE     NAME                 TIME/DISTANCE        SCHOOL
  1
  2
  3
  4
  5
                      800 METERS--GIRLS
PLACE     NAME                 TIME/DISTANCE        SCHOOL
  1
  2
  3
  4
  5
                    MILE RUN--BOYS
```

Genetic Engineering

Dick Straw



Have you ever thought you would like to be a genetic engineer? You know, someone with the skills and techniques to make plants and animals to your own designs? Or to eliminate some wasteful genetic disease by skillfully manipulating the hereditary material?

The means for accomplishing real genetic engineering are still not available, although biologists working in that field get closer and closer each year. With the CODON program that follows, you can simulate some of the hoped-for techniques, however, and get some ideas about how the genes do their work. If you happen to be a biology teacher, you can use the program to help students learn some of the basics of molecular biology as well.

Even if you are just an ordinary programming buff without much concern for genetic engineering or molecular biology, the CODON program might still be of interest to you because of the string-handling routines involved. They show off some important ways that the Basic language is, I feel, superior to Fortran, which handles strings only with difficulty. (Calm the hackles, friend — I use Fortran a lot too, when its advantages are needed!)

In addition, there is a segment that will direct all input and output to the printer without modifying the program to use LPRINT statements. (I owe this idea to Charles Butler, from an article in *Personal Computing.)*

This is not the place for an extended lesson in molecular biology. Almost all recent high school and college general biology books will give a good and more complete description of the details. A book called *Invitation to Biology* by Helena Curtis, used widely for college biology courses taken by non-specialists, is a good example. For a more detailed and superbly written account, try James Watson's (of double helix fame) book, *The Molecular Biology of the Gene.* Still, a very brief account might make the program more interesting.

Practically all the work of the cell is accomplished by proteins. These proteins are large and complex molecules made up of twenty elementary building blocks called amino acids. The numbers, kinds and order of the amino acids determine the shape and thus the function of the protein. Think of them, if you will, as factory "jigs" that hold things together so a needed chemical operation can be carried out. What molecules the proteins hold and what they do with them is determined entirely by the arrangement of their amino acids. Whether the cell or individual survives depends on what they do and how well, however.

What kinds of jigs the cellular factory makes is determined by the kinds of genes in the nucleus of the cell. You can think of genes as the specifications for building jigs. Every time a new factory (cell) is built, the jig-specifying genes are copied and passed on to it from the previous cell. Every time a new individual (a new person or flower, full of cell-factories) is produced, the cells in it get a portion of the genes from each parent and can make new combinations of jigs. This determines the individual's unique characteristics. But the genes are not used directly — they are first copied into "blueprints" to be used in the factory itself, outside the nucleus, where the genes are never allowed to go.

The genes are really coded structures made up of four kinds of molecules called "bases," put together into a complex structure called a nucleic acid. A certain length of the kind of nucleic acid found in chromosomes, called DNA, is a single gene, and it carries the specifications for a single protein to be built. The blueprints are copies of this genetic information in a second kind of nucleic acid, RNA, which also carries its message in the arrangements of four sub-units. These sub-unit bases are the chemical molecules called adenine, cytosine, guanine, and uracil, and are referred to by their initials. Remember that their order is a code that specifies the organization of a particular protein with a specific function.

When the blueprint-RNA is shipped out into the cell's factory, it is operated on by a cellular structure that reads the message in a very special manner, much as a film-strip projector shows the film one frame at a time. The frames in this case contain three of the bases at a time, and each set of three bases in a row is called a codon — the smallest piece of information in the RNA message. Recall that there are twenty kinds of amino acids but only four kinds of bases. This means that the smallest quantity of coded information that can translate a code of four symbols into another of twenty symbols must have a minimum of three symbols per unit. (There is an exercise in combinatorics for you.) Each codon of the RNA is thus equivalent to one amino acid (except for three that are terminators and give a STOP message that cuts off the protein building at that point). With four symbols taken three at a time in specific order, there are 64 kinds of codons. Since there are only twenty amino acids, you get a lot of synonyms (called redundancy), but that has its advantages too.

Mutations are changes in the structure of the genes that, by way of the messenger RNA, are translated into incorrectly built proteins. These

Richard M. Straw, 2100 Maiden Lane, Altadena, CA 91001.

"bad" proteins usually either do something wrong or do not do whatever they were supposed to do. Either of these can upset the functioning of the cell and the individual. Thus things that cause mutations, like radiation and many kinds of chemicals, are bad for the health also.

Enough of that. Let's look at what the program does.

Briefly, CODON invites you to enter a sequence of base initials, A, C, G or U, in any order you desire up to a total of 150 in up to three lines. Short examples do as well as long ones, usually. This message is then echoed back to you in codon-groups, three letters at a time, as shown in Figure 1.

```
RNA MESSAGE NOW READS (BY CODONS):

AUC AUC AUC AUC AUC AUC

YOUR INPUT STRING TRANSLATES TO:

ILE-ILE-ILE-ILE-ILE-ILE-
```

Figure 1. The original input string of RNA bases is echoed in codon groups of three bases and translated into the corresponding amino acid string. Only one codon type is used to make the changes more evident.

Here, a simple sequence repeating the codon AUC was used. The program then translates that RNA message into the corresponding sequence of amino acids it would produce in the cell. In the example, AUC translates into a sequence of the amino acid, isoleucine, abbreviated by the letters ILE. If you look at lines 7160-7250 you will see the abbreviations used, which will be printed for you if you ask for information at the beginning of the program. See the full sample run, too.

You then have three options (other than stopping): you can change one

```
RNA MESSAGE NOW READS (BY CODONS):

AUC AUC ACC AUC AUC AUC

PROTEIN HAS NOW BECOME:

ILE-ILE-THR-ILE-ILE-ILE-
```

Figure 2. The base at position 8 (indicated by an arrow) was changed from U to C, and the resulting amino acid string has one different acid because of this mutation.

base in the message to another, or insert some more bases into the list, or take out some bases. Figure 2 shows what happens when you change base number 8 from U to C. One amino acid in the structure is changed this time, from ILE to THR, or threonine. This simple mutation, representative of the smallest genetic change possible, can be extremely important. The only difference between the normal hemoglobin that most of us have in our cells and the sickle-cell hemoglobin that can kill its carriers through severe anemia is just such a single amino acid substitution, putting in valine instead of the glutamic acid that is usually present. The RNA message change was probably putting a U where an A belonged.

Figure 3 shows what happens when you add a single base. If you

```
RNA MESSAGE NOW READS (BY CODONS):

AUC AUC CAC CAU CAU CAU C

CURRENT PROTEIN READS:

ILE-ILE-HIS-HIS-HIS-HIS-
```

Figure 3. A new base, C, was inserted into the string after position 6, with the result that the whole message has changed beyond the position of the insertion. Note also that two different codons have been translated into one amino acid — an example of code redundancy.

examine the sequence, you will find that a new C has been stuck into the sequence at position 7. The list is now one base longer, but the program does not translate the stray end. Two things will be apparent. First, you can see that the whole protein after the insertion has been changed — to a sequence of histidines (HIS). Secondly, the third and fourth codons, CAC and CAU, both translate into HIS, an example of the code redundancy.

If you now take out a base, number 13 in the example, you get the results shown in Figure 4. Here you can see

```
RNA MESSAGE NOW READS (BY CODONS):

AUC AUC AUC AUC AUC AUC

YOUR INPUT STRING TRANSLATES TO:

ILE-ILE-ILE-ILE-ILE-ILE-
```

Figure 1. The original input string of RNA bases is echoed in codon groups of three bases and translated into the corresponding amino acid string. Only one codon type is used to make the changes more evident.

that there is a place in the sequence between the insertion and the deletion that is different from the original, but both ends are back to where you started. It was by using techniques like this that Nobel prizewinner Francis Crick demonstrated that codons really were three bases long — adding three or taking out three bases leaves the protein between modified but the ends the same. If the modified length is not too great, the protein may still function partly — depending on a lot of things. In general, however, any mutations cause problems that may vary from insignificant to great.

Each of the parts of the program is marked by REM lines so that it should be easy to follow. The codon dictionary (found in the data lines 6000 to 6080) is read into array C by lines 150 — 170. It is used to translate the RNA message. If your version of Basic does not accept three-dimensional arrays some rewriting will be needed here. Notice also that I used the DEFSTR (define string) and DEFINT (define integer) functions of the TRS-80 Basic to declare the types of several variables ahead of time. At least the C's, A's and S's in the program would need to be changed to C$'s, etc., if you do not have this capacity. Also permitted here, but perhaps not by your interpreter, is the reading of string data without quotation marks from lines 6000 - 6080.

The program segment in lines 211 to 216 allows you to divert all input and output to the printer if you wish. If you answer "yes" to the first question, the two pairs of PEEK statements pick up the addresses of the video driver (VL% and VH% — low and high bytes) and printer driver (PL% and PH%), and then two POKEs put the printer driver address where the video driver address usually resides. When you stop the run (at lines 510-530, by selecting option 4 from the menu), the video addresses are restored to their proper places. The complete run example was produced by this option. It is best not to stop the program with the <BREAK> key.

The routine at lines 340-390 reads in your input string and puts together a long one, C1. The echo routine at subroutine 8700 puts the input string C1 into string C5 and lists the input string by codons. String C5 has spaces inserted between the groups of three bases, which are not wanted by the translating routines.

Translating is done by the subroutine beginning at 8000, which in turn calls another at 8200. The second translates the letters of each successive codon in your input string into numbers by which to access the array C, and the prime subroutine builds another string, A, which is then printed out by the subroutine at 8500. In both printing subroutines line lengths are controlled so a uniform and neat output is produced. The whole program is written for video display; but, as noted above, output can be sent to the printer if you wish.

The segments that insert or delete strings of bases into the original (or any subsequent) string start at lines

2000 and 3000. All they do is find out what location is desired, break the original string into the two ends needed, and put in or take out what you request. If you ask to delete beyond the end of the string, the end is cut off and the extra "deletion" is ignored. You can't delete more than the whole string, though.

The program is written in Disk Basic, but the only unique feature is used in line 1060 of the change-a-base routine. This is the MID$ function on the left side of an assignment statement. To convert to Level II Basic this line would need to be rewritten to use the techniques of the insert or delete methods. The following substitution for line 1060 will work:

```
1060 C2=LEFT$(C1,K1-1):
C3=RIGHT$(C1,L-K1):
C1=C2+A5+C3
```

It is the only change that should be needed, however, as everything else is standard usage.

With that amount of description, everything else should be easily followed. Most common mistakes (like entering an incorrect letter) are trapped, and lots of prompts and information are provided to make this an essentially independent program once its basic intentions are understood. □

Figure 5. A full run of the program using the printer option, including the information that is optionally provided.

```
DO YOU WANT INFORMATION AND INSTRUCTIONS? Y

MESSENGER RNA IS A COPY OF THE GENETIC INFORMATION IN
THEN DNA OF THE GENE IN THE CHROMOSOME. THE ORDER OF THE
FOUR BASES (A = ADENINE, C = CYTOSINE, G = GUANINE, AND
U = URACIL) DETERMINES THE ORDER OF THE AMINO ACIDS IN
THE PROTEIN.  THAT ORDER REGULATES THE FUNCTION OF THE
PROTEIN IN THE CELL.

YOU MAY ENTER A STRING OF BASES TO SIMULATE THE
STRUCTURE OF THE RNA MOLECULE.  USE ONLY THE LETTERS,
A, C, G, AND U, AND LEAVE NO SPACES. IF YOU WANT MORE
THAN ABOUT 50, USE 2 OR 3 LINES BUT DO NOT ENTER MORE
THAN 150 BASES IN ALL.

    DO YOU WANT MORE INFORMATION? Y

EACH GROUP OF THREE BASES IN THE MESSAGE IS CALLED A CODON.
EACH CODON TRANSLATES INTO ONE AMINO ACID IN THE PROTEIN.
THE AMINO ACID ABBREVIATIONS ARE:
ALA = ALANINE           LEU = LEUCINE
ARG = ARGENINE          LYS = LYSINE
ASN = ASPARAGINE        MET = METHIONINE
ASP = ASPARTIC ACID     PHE = PHENYLALANINE
CYS = CYSTEINE          PRO = PROLINE
GLN = GLUTAMINE         SER = SERINE
GLU = GLUTAMIC ACID     THR = THREONINE
GLY = GLYCINE           TRP = TRYPTOPHANE
HIS = HISTIDINE         TYR = TYROSINE
ILE = ISOLEUCINE        VAL = VALINE
    READY? Y

YOU MAY NOW ENTER AN RNA MESSAGE.
ENTER NO MORE THAN THREE LINES OF DATA NOR MORE THAN 150
BASES IN ALL, USING ONLY LETTERS A, C, G, AND U
RESPOND TO '?' WITH <ENTER> IF DONE WITH INPUT.
? ACCGUGCACUAGGUCAAUCG
?
LENGTH IS 20

RNA MESSAGE NOW READS (BY CODONS):

ACC GUG CAC UAG GUC AAU CG

YOUR INPUT STRING TRANSLATES TO:

THR-VAL-HIS-STOP

YOU CAN MAKE ANY OF THESE CHANGES:
   1.   CHANGE ONE BASE IN THE LIST.
   2.   INSERT ONE OR MORE ADJACENT BASES.
   3.   DELETE ONE OR MORE ADJACENT BASES.
  OR STOP (TYPE 4)
WHICH DO YOU CHOOSE? 1

CHANGE A SINGLE BASE IN RNA MESSAGE


CHANGE WHICH BASE (NUMBER)? 10
BASE NUMBER 10 IS NOW U

WHAT BASE DO YOU WISH TO CHANGE IT TO? C

RNA MESSAGE NOW READS (BY CODONS):

ACC GUG CAC CAG GUC AAU CG

PROTEIN HAS NOW BECOME:

THR-VAL-HIS-GLN-VAL-ASN-

YOU CAN MAKE ANY OF THESE CHANGES:
   1.   CHANGE ONE BASE IN THE LIST.
   2.   INSERT ONE OR MORE ADJACENT BASES.
   3.   DELETE ONE OR MORE ADJACENT BASES.
  OR STOP (TYPE 4)
WHICH DO YOU CHOOSE? 2

INSERT BASES INTO RNA MESSAGE

ENTER STRING OF BASES TO INSERT:
? CCAAGGUUCAC
INSERT AFTER WHICH BASE (NUMBER)? 18
RNA NOW HAS 31  BASES

RNA MESSAGE NOW READS (BY CODONS):

ACC GUG CAC CAG GUC AAU CCA AGG UUC ACC G

CURRENT PROTEIN READS:

THR-VAL-HIS-GLN-VAL-ASN-PRO-ARG-PHE-THR-

YOU CAN MAKE ANY OF THESE CHANGES:
   1.   CHANGE ONE BASE IN THE LIST.
   2.   INSERT ONE OR MORE ADJACENT BASES.
   3.   DELETE ONE OR MORE ADJACENT BASES.
  OR STOP (TYPE 4)
WHICH DO YOU CHOOSE? 3

DELETE BASES FROM RNA MESSAGE

HOW MANY BASES DO YOU WISH TO REMOVE? 4

DELETE BEGINNING WITH WHICH BASE (NUMBER)? 5
RNA NOW HAS 27  BASES

RNA MESSAGE NOW READS (BY CODONS):

ACC GCC AGG UCA AUC CAA GGU UCA CCG

PROTEIN NOW HAS THIS STRUCTURE:

THR-ALA-ARG-SER-ILE-GLN-GLY-SER-PRO-

YOU CAN MAKE ANY OF THESE CHANGES:
   1.   CHANGE ONE BASE IN THE LIST.
   2.   INSERT ONE OR MORE ADJACENT BASES.
   3.   DELETE ONE OR MORE ADJACENT BASES.
  OR STOP (TYPE 4)
WHICH DO YOU CHOOSE? 4
```

```
10 REM      CODON PROGRAM
20 REM      (C) DICK STRAW 1980
30 REM      TRS-80 DISK BASIC
40 REM
100 CLEAR 2000
110 DEFSTR C,A,S: DEFINT I-N
120 DIM C(4,4,4)
150 FOR I=1 TO 4:FOR J=1 TO 4: FOR K=1 TO 4
160 READ C(I,J,K)
170 NEXT K,J,I
200 CLS:PRINTTAB(20);"CODON PROGRAM"
210 PRINT:PRINT"SIMULATES TRANSLATION OF MESSENGER-RNA INTO PROTEIN"
211 PR%=0: INPUT"DO YOU WANT OUTPUT TO PRINTER";S1
212 IF ASC(S1) <> 89 GOTO 220
213 PR%=1: INPUT"WHEN PRINTER IS READY, PRESS ENTER";S1
214 PL%= PEEK(16422): PH%= PEEK(16423)
215 VL%= PEEK(16414): VH%= PEEK(16415)
216 POKE 16414,PL%: POKE 16415,PH%
220 PRINT:INPUT"DO YOU WANT INFORMATION AND INSTRUCTIONS";S1
230 IF ASC(S1)=89 THEN GOSUB 7000
299 REM    START MAIN PROGRAM
300 PRINT:PRINT"YOU MAY NOW ENTER AN RNA MESSAGE."
310 PRINT"ENTER NO MORE THAN THREE LINES OF DATA NOR MORE THAN 150"
320 PRINT"BASES IN ALL, USING ONLY LETTERS A, C, G, AND U"
330 PRINT"RESPOND TO '?' WITH <ENTER> IF DONE WITH INPUT."
340 C1=""
350 FOR I=1 TO 3
360 C2="": INPUT C2
370 IF C2="" GOTO 400
380 C1=C1+C2
390 NEXT I
400 L=LEN(C1): PRINT"LENGTH IS";L
405 GOSUB 8700
410 PRINT:PRINT"YOUR INPUT STRING TRANSLATES TO:"
420 GOSUB 8000          'TRANSLATING ROUTINE
430 GOSUB 8500          'PRINT PROTEIN
440 PRINT:PRINT"YOU CAN MAKE ANY OF THESE CHANGES:"
450 PRINT"   1.   CHANGE ONE BASE IN THE LIST."
460 PRINT"   2.   INSERT ONE OR MORE ADJACENT BASES."
470 PRINT"   3.   DELETE ONE OR MORE ADJACENT BASES."
480 PRINT"  OR STOP (TYPE 4)"
490 INPUT"WHICH DO YOU CHOOSE";N
500 ON N GOTO 1000, 2000, 3000, 510
510 IF PR%=0 GOTO 530
520 POKE 16414,VL%: POKE 16415,VH%
530 STOP
999 REM  CHANGE ONE BASE
1000 PRINT:PRINT"CHANGE A SINGLE BASE IN RNA MESSAGE"
1010 PRINT:INPUT"CHANGE WHICH BASE (NUMBER)";K1
1020 IF K1>L OR K1<0 PRINT"INVALID NUMBER":GOTO 1010
```

```
1030 A5= MID$(C1,K1,1)
1040 PRINT"BASE NUMBER";K1;"IS NOW ";A5
1050 INPUT"WHAT BASE DO YOU WISH TO CHANGE IT TO";A5
1060 MID$(C1,K1,1) = A5
1070 GOSUB 8700              'PRINT RNA
1080 PRINT:PRINT"PROTEIN HAS NOW BECOME:"
1090 GOSUB 8000
1100 GOSUB 8500
1110 GOTO 440
1999 REM  INSERT BASES
2000 PRINT:PRINT"INSERT BASES INTO RNA MESSAGE"
2010 PRINT: A5="": PRINT"ENTER STRING OF BASES TO INSERT:"
2020 INPUT A5
2060 L2=LEN(A5): IF L2+L > 150 PRINT"TOO LONG":GOTO 2010
2070 INPUT"INSERT AFTER WHICH BASE (NUMBER)";K1
2080 IF K1>L PRINT"NOT A VALID LOCATION - PAST END":GOTO 2070
2090 C2=LEFT$(C1,K1): L2=L-K1
2100 C3=RIGHT$(C1,L2)
2120 C1=C2+A5+C3: L=LEN(C1)
2130 PRINT"RNA NOW HAS";L;" BASES"
2140 GOSUB 8700
2150 PRINT:PRINT"CURRENT PROTEIN READS:"
2160 GOSUB 8000
2170 GOSUB 8500
2180 GOTO 440
2999 REM   REMOVE BASES
3000 PRINT:PRINT"DELETE BASES FROM RNA MESSAGE"
3010 PRINT:INPUT"HOW MANY BASES DO YOU WISH TO REMOVE";L2
3020 IF L-L2 < 0 PRINT"RNA IS NOT THAT LONG":GOTO 3010
3030 PRINT:INPUT"DELETE BEGINNING WITH WHICH BASE (NUMBER)";K1
3040 L3=L-K1-L2: IF L3<0 THEN L2=L-K1: GOTO 3040
3050 C2=LEFT$(C1,K1-1): C3=RIGHT$(C1,L3+1)
3060 C1=C2+C3: L=LEN(C1)
3070 PRINT"RNA NOW HAS";L;" BASES"
3080 GOSUB 8700
3090 PRINT:PRINT"PROTEIN NOW HAS THIS STRUCTURE:"
3100 GOSUB 8000
3110 GOSUB 8500
3120 GOTO 440
4000 END
4001 REM

6000 REM   CODON DICTIONARY
6010 DATA PHE-,PHE-,LEU-,LEU-,SER-,SER-,SER-,SER-
6020 DATA TYR-,TYR-,STOP,STOP,CYS-,CYS-,STOP,TRP-
6030 DATA LEU-,LEU-,LEU-,LEU-,PRO-,PRO-,PRO-,PRO-
6040 DATA HIS-,HIS-,GLN-,GLN-,ARG-,ARG-,ARG-,ARG-
6050 DATA ILE-,ILE-,ILE-,MET-,THR-,THR-,THR-,THR-
6060 DATA ASN-,ASN-,LYS-,LYS-,SER-,SER-,ARG-,ARG-
6070 DATA VAL-,VAL-,VAL-,VAL-,ALA-,ALA-,ALA-,ALA-
6080 DATA ASP-,ASP-,GLU-,GLU-,GLY-,GLY-,GLY-,GLY-
6999 REM
7000 REM  INFORMATION TEXT
7010 PRINT:PRINT"MESSENGER RNA IS A COPY OF THE GENETIC INFORMATION IN"
7020 PRINT"THEN DNA OF THE GENE IN THE CHROMOSOME. THE ORDER OF THE"
7030 PRINT"FOUR BASES (A = ADENINE, C = CYTOSINE, G = GUANINE, AND"
7040 PRINT"U = URACIL) DETERMINES THE ORDER OF THE AMINO ACIDS IN"
7050 PRINT"THE PROTEIN.  THAT ORDER REGULATES THE FUNCTION OF THE"
7060 PRINT"PROTEIN IN THE CELL."
7070 PRINT:PRINT"YOU MAY ENTER A STRING OF BASES TO SIMULATE THE"
7080 PRINT"STRUCTURE OF THE RNA MOLECULE.  USE ONLY THE LETTERS,"
7090 PRINT"A, C, G, AND U, AND LEAVE NO SPACES. IF YOU WANT MORE"
7100 PRINT"THAN ABOUT 50, USE 2 OR 3 LINES BUT DO NOT ENTER MORE"
7110 PRINT"THAN 150 BASES IN ALL."
7120 PRINT:INPUT"    DO YOU WANT MORE INFORMATION";S1
7130 IF ASC(S1) <> 89 THEN RETURN
7140 PRINT:PRINT"EACH GROUP OF THREE BASES IN THE MESSAGE IS CALLED A CODON."
7150 PRINT"EACH CODON TRANSLATES INTO ONE AMINO ACID IN THE PROTEIN."
7155 PRINT"THE AMINO ACID ABBREVIATIONS ARE:"
7160 PRINT"ALA = ALANINE            LEU = LEUCINE"
7170 PRINT"ARG = ARGENINE           LYS = LYSINE"
7180 PRINT"ASN = ASPARAGINE         MET = METHIONINE"
7190 PRINT"ASP = ASPARTIC ACID      PHE = PHENYLALANINE"
7200 PRINT"CYS = CYSTEINE           PRO = PROLINE"
7210 PRINT"GLN = GLUTAMINE          SER = SERINE"
7220 PRINT"GLU = GLUTAMIC ACID      THR = THREONINE"
7230 PRINT"GLY = GLYCINE            TRP = TRYPTOPHANE"
7240 PRINT"HIS = HISTIDINE          TYR = TYROSINE"
7250 PRINT"ILE = ISOLEUCINE         VAL = VALINE"
7260 INPUT"    READY";S1
7270 RETURN
7999 REM
8000 REM  TRANSLATING ROUTINE
8010 A="": FOR I=1 TO L STEP 3
8020 C3="": C3= MID$(C1,I,3)
8025 IF LEN(C3) < 3 GOTO 8130
8030 A2 = LEFT$(C3,1): GOSUB 8200
8040 J1=J: IF J=0 GOTO 8120
8050 A2 = MID$(C3,2,1): GOSUB 8200
8060 J2=J: IF J=0 GOTO 8120
8070 A2 = RIGHT$(C3,1): GOSUB 8200
8080 J3=J: IF J=0 GOTO 8120
8090 A1= C(J1,J2,J3): A=A+A1: IF A1="STOP" GOTO 8130
8100 NEXT I
8120 IF J=0 PRINT"BAD BASE IN CODON AT POSITION";I;" - ";C3
8130 RETURN
8200 REM DECODING BASES TO DICTIONARY INDICES
8205 J=0
8210 IF A2="U" THEN J=1: GOTO 8250
8220 IF A2="C" THEN J=2: GOTO 8250
8230 IF A2="A" THEN J=3: GOTO 8250
8240 IF A2="G" THEN J=4
8250 RETURN
8500 REM    PRINT PROTEIN
8510 IF LEN(A) < 57 PRINT :PRINT A: RETURN
8520 A1 = LEFT$(A,56)
8530 A2 = MID$(A,57,56): A3="": A4=""
8540 IF LEN(A) > 112 THEN A3=MID$(A,113,56)
8550 IF LEN(A) > 168 THEN A4=MID$(A,169,56)
8560 PRINT:PRINT A1: PRINT: PRINT A2
8570 IF LEN(A3) > 0 THEN PRINT: PRINT A3
8580 IF LEN(A4) > 0 THEN PRINT: PRINT A4
8590 RETURN
8700 REM    PRINT RNA MESSAGE
8705 PRINT:PRINT"RNA MESSAGE NOW READS (BY CODONS):"
8710 C5="": FOR I=1 TO LEN(C1) STEP 3
8720 C5=C5+MID$(C1,I,3)+" "
8725 NEXT I
8730 IF LEN(C5)<57 PRINT: PRINT C5: RETURN
8740 C2=LEFT$(C5,56): C3=MID$(C5,57,56): C7="":C8=""
8750 IF LEN(C5) > 112 THEN C7=MID$(C5,113,56)
8760 IF LEN(C5) > 168 THEN C8=MID$(C5,169,56)
8770 PRINT:PRINT C2: PRINT: PRINT C3
8780 IF LEN(C7)>0 THEN PRINT:PRINT C7
8790 IF LEN(C8)>0 THEN PRINT:PRINT C8
8800 RETURN
```

Calculus on a TRS-80

Numerical Integration on a TRS-80

Alan Scheidler

Are you intimidated by seemingly difficult integration formulas? Do you have to search through a table of integrals every time you need to calculate the area under the graph of a curve? Are you a student of college calculus? If so, please read on. I think I can show you how to perform even the most difficult integration—and even save time. If you don't know calculus, I believe I can introduce you to an interesting and valuable tool which can be used to solve many practical problems.

Alan Scheidler, 3739 53rd St., Apt. 211, Moline, Ill. 61625.

Calculus teaches basically two types of operations which can be performed on mathematical expressions. One such operation is called differentiation which can be used to find the slope of a curve at any prescribed point. This article deals with another operation called integration which can be used to find the exact area under a curve between two specified points. Some people spend years studying and perfecting the techniques needed to perform both operations. More often than not, these techniques involve more art than science and require a great deal of memorization.

A Simple Integration Example

Suppose you are asked to determine

the area under the curve $y = x^2$ between the x-values of 0 and 3 as shown in Figure 1. If you use the integration techniques prescribed by calculus you can calculate the exact area. But before you old calculus buffs blow the dust off your textbooks, let's analyze the problem a little more closely. Let's apply an old integration principle called the "Approximation Principle" to the problem. The Approximation Principle states that a difficult integration problem may be reduced to one or more very simple problems if allowance is made for inaccuracy. In other words, if you allow yourself a bit of inaccuracy, you can get an approximation of the area under the curve $y = x^2$. For example, you can obtain a very rough estimate of the area by connecting points (0,0) and (3,9) with a straight line (Figure 2). Then by calculating the area of the triangle formed by the three points (0,0), (3,9), and (3,0) you find that the area under the curve is 1/2 x 3 x 9 = 13.5.

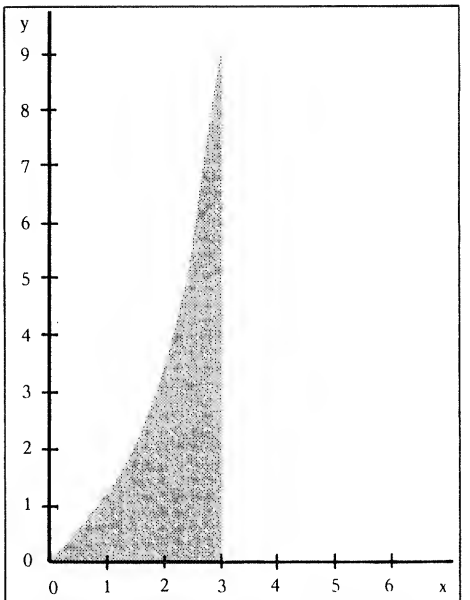


Figure 1. Area under $y = x^2$ between 0 and 3.

Normally this crude estimate is not good enough. The Approximation Principle can be carried a step further to get a better approximation. You can divide the area under the curve into three equally spaced sections as in Figure 3. Each segment is one unit wide. Again connect the points by straight lines. Now calculate the area under each trapezoid. The formula to calculate the area under a trapezoid is:

$$\text{Area} = \frac{(h1 + h2)}{2} \times b$$

where h1 = height of the left side of trapezoid
h2 = height of the right side of trapezoid
b = base of the trapezoid

In Figure 3, the first area starting on the left is actually a triangle which is merely a special case of a trapezoid with the height of the left side equal to zero. If you proceed to calculate and sum up the individual areas using the above trapezoid formula you will find the area under $y = x^2$ to be:

$$\text{Area} = \frac{(0 + 1)}{2} \times 1 + \frac{(1 + 4)}{2} \times 1 + \frac{(4 + 9)}{2} \times 1$$

Area = 9.5

This is a much better approximation—but probably still not good enough for you budding mathematicians. No doubt you've already guessed the exact area is somewhat less than 9.5. This is true because the curve $y = x^2$ always lies on or below the straight lines which connect the top corner points of the trapezoids. The error in our area approximations is equal to the small crescent shaped areas which lie between the oblique trapezoid sides and the curve.

This error can never be totally eliminated, but it can be minimized. You've already seen how it can be reduced by calculating three areas instead of one. Well now, let's divide the area into 10 segments and again sum the areas enclosed by the trapezoids. If this sounds like too much work, well, you're right—but this is just the sort of thing that computers were intended for. The Basic program listed in Figure 4 is designed to run on a TRS-80 Level II microcomputer to perform just such a task. This program can also be run on a Level I machine by eliminating line 20 and changing all variable names to just the first letter of each variable name as listed.

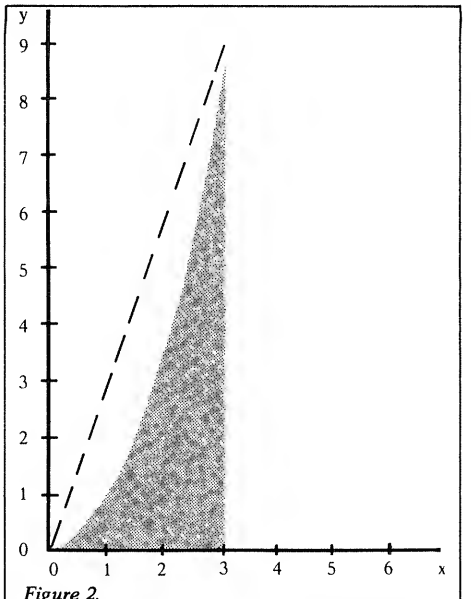


Figure 2.
A very rough estimate of the area under $y = x^2$.

If you run this program, you will find that with 10 segments, the estimated area is 9.045. As long as you have the computer handy, try 100 segments. Here the calculated area will be 9.00044. If you are a calculus expert you've no doubt already calculated the exact area to be 9. So you can see the computer result comes very close to the correct value.

This technique for calculating the area under a curve is called Numerical Integration using the Trapezoidal Rule. As we have seen, the Trapezoidal Rule approximates the area under a curve by summing the areas of a specified number of trapezoids. The height of a vertical side of a trapezoid is equal to the y-value which corresponds to particular x-value. Each x-value is found by dividing the integration interval into the desired number of equally spaced segments. The y-value which corresponds to a particular x-value is found by substituting the x-value into the equation of the curve. Thus, for the curve $y = x^2$, the y-value which corresponds to an x-value of 2 is 4. For an x-value of 2.5, the corresponding y-value is 2.5^2 or 6.25, and so on.

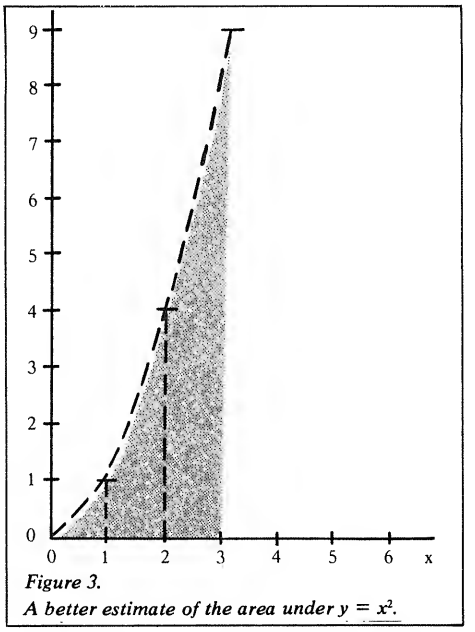


Figure 3.
A better estimate of the area under $y = x^2$.

The program listed in Figure 4 functions in just this way. First the program asks you to input the upper and lower limits (U and L respectively) which mark the interval over which the integration will be performed. Then it will ask for the number of calculation increments, N, which is actually the number of trapezoids which will be used to approximate the area. With this information, the computer calculates a DX value which is the width of each trapezoid.

Then with X starting at the lower limit, N + 1 y-values are computed, incrementing

X by DX each time until the upper limit for X has been reached. A(I) is an array used to represent the y-values. Once all of the y-values have been computed, the accumulated area under the curve is calculated by summing the areas of each individual trapezoid using the trapezoid area formula. Then this accumulated area is displayed by the computer. The accumulated area is the result of the numerical integration.

A Practical Example Problem

The trapezoidal numerical integration program can be used to solve difficult practical problems. Here is an example of the type of problem that can be solved using this method.

You are the engineer in charge of emergency power systems at a nuclear power plant. Your responsibility is to make sure that two 5000 horsepower emergency diesel generators start up to provide standby electrical power upon the loss of both normal and reserve power sources.

Large quantities of diesel fuel must be stored at the plant at all times to ensure that there is enough fuel for even the longest power outages. For maximum safety and reliability, each diesel generator is supplied by its own diesel fuel storage tank. Part of your responsibility is to monitor the oil level in these tanks. Each tank measures 12 feet in diameter by 50 feet in length and is buried lengthwise underground to reduce fire-hazard. The level indicator gauge on one of the tanks indicates an oil level of nine feet. How many gallons of diesel fuel does this represent?

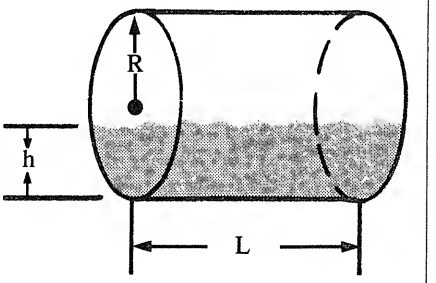


Figure 5. Diesel fuel tank.

The essentials of this problem are diagrammed in Figure 5. To calculate the maximum volume of a tank, you must multiply the cross sectional area of the tank by its length. Thus, the maximum volume of a cylindrical tank is the area of a circle whose diameter is equal to the diameter of the tank multiplied by the length of the tank. This is simple enough, but what about a partially filled tank? Basically the problem is still the same—determine the area of the diesel fuel cross section and multiply by the length. In this case, the diesel fuel cross section will appear to be a circle with its top cut off. This sounds very simple until you actually try to calculate this area.

This problem may appear to be a very difficult problem—and it definitely is difficult if you want absolute accuracy, but let's not forget about the time-honored Approximation Principle. If we can tolerate a small degree of inaccuracy, we can make use of the Trapezoidal rule integration program.

To make use of this program, we must determine the equation of the curve to be integrated. You've probably already guessed that the curve to be integrated will be the equation of a circle whose diameter is equal to the diameter of the tank. To simplify the derivation of this equation, I suggest that you place the origin (0,0) at the bottom of the tank with the x-axis pointing straight up as shown in Figure 6. By using this orientation, you can integrate under the circle from a lower limit of 0 to an upper limit equal to the depth of diesel fuel in the tank. This will give you the area of the diesel fuel cross section in the tank. The general equation of the circle which describes the tank is:

$$(x - a)^2 + (y - b)^2 = R^2$$

where R = the radius of the tank
a = R
b = 0

(the point (a,b) is the coordinates of the center of the circle or tank in this case)

Substituting the values for a and b, the equation becomes:

$$(x - R)^2 + y^2 = R^2$$

Before we use this equation in the integration program, it must be rearranged so that y is isolated on the left side of the equation:

$$y = \sqrt{R^2 - (x - R)^2}$$

You can now use the integration program in the same way you did when you calculated the area under $y = x^2$. Just substitute this new equation in its place. The lower limit of integration will be 0 and the upper limit will be 9, which is the depth of diesel fuel in the tank. This result will be the area under the curve which is described by the above equation.

You have probably guessed that if you perform this integration, you actually get only one-half of the area of the desired cross section. This is because the Trapezoid integration program only calculates the area above the x-axis. We can easily correct this discrepancy by multiplying by 2. Multiplying this result by L, the length of the tank, will give you the volume of diesel fuel in the tank in cubic feet:

$$y = 2L\sqrt{R^2 - (x - R)^2}$$

Now if you want to get real fancy, you can multiply this equation by a factor of 7.4805195 to convert cubic feet to U.S. gallons. The final equation to integrate is then:

```
10 CLS
20 DIM A(101)
30 PRINT "NUMERICAL INTEGRATION USING THE TRAPEZOIDAL RULE"
40 PRINT "ENTER THE FORMULA TO BE INTEGRATED IN LINE 150"
50 INPUT "ENTER THE LOWER LIMIT "; L
60 INPUT "ENTER THE UPPER LIMIT "; U
70 INPUT "ENTER THE NUMBER OF CALCULATION INCREMENTS "; N
80 AREA = 0
90 DX = (U - L)/N
100 X = L
110 FOR I = 1 TO N + 1
120 '
130 'REPLACE THE NEXT LINE WITH THE FORMULA TO BE INTEGRATED
140 '
150 A(I) = X * X
160 X = X + DX
170 NEXT I
180 FOR I = 1 TO N
190 AREA = AREA + DX * (A(I) + A(I + 1))/2
200 NEXT I
210 PRINT "THE AREA UNDER THE CURVE FROM "; L ;" TO "; U
220 PRINT "IS "; AREA
230 PRINT "DO YOU WISH TO CHANGE THE INPUT PARAMETERS (Y / N) "
240 INPUT A$
250 IF A$ = "Y" THEN 50
260 IF A$ = "N" THEN END
270 GOTO 210
```

Figure 4.

$$y = 2L\sqrt{R^2 - (x - R)^2} \times 7.4805195$$

where L = 50
R = 6
x varies from lower limit to upper limit (0 to 9)

Simply substitute this equation for $y = x^2$ in the integration program, and you're all set. Using a lower limit of 0 and an upper limit of 9 with 20 increments, the calculated volume is 33862 gallons. The exact volume which can be derived very painstakingly by the methods of calculus is 34031 gallons. This amounts to an error of only 0.50%, which is good enough accuracy for most situations. When you actually substitute this equation into the the integration program I suggest that you avoid using the raising-to-a-power function which is available in TRS-80 Basic. This is just a precaution against negative numbers. In other words you cannot raise a negative number to a power, but you can always multiply a negative number by itself. Thus instead of using Y = X ↑ 2, use Y = X * X.

By the way, for all you die-hard calculus addicts, here is the closed form solution for the volume of the diesel fuel in the tank:

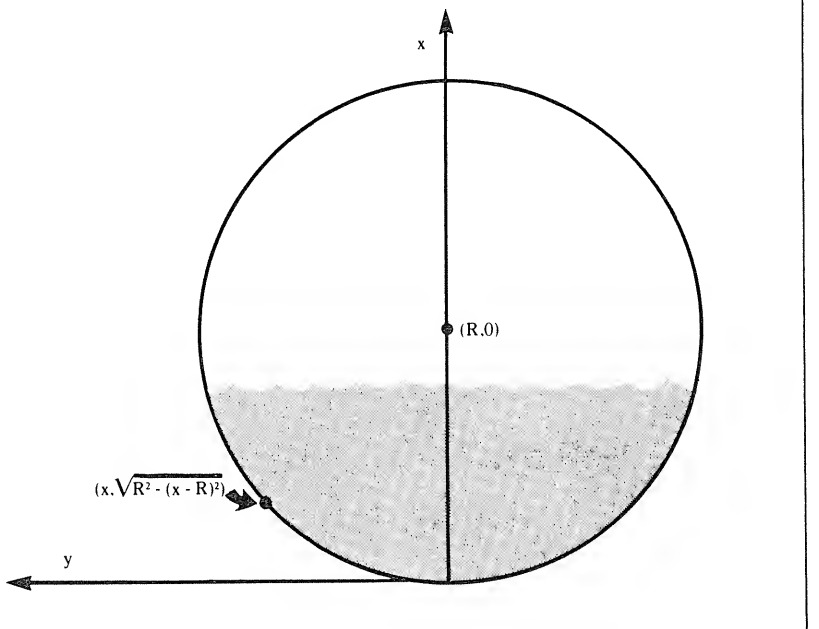


Figure 6. The coordinate system for the tank.

$$\text{Volume} = L\left[(h - R)\sqrt{2Rh - h^2} + R^2 \text{Arctan}\left(\frac{Q}{\sqrt{1 - Q^2}}\right) + \frac{\pi}{2}R^2\right] \times 7.4805195 \quad \text{(gallons)}$$

where
$Q = (h - R)/R$
$h =$ diesel fuel depth (ft)
$R =$ radius of tank (ft)
$L =$ length of tank (ft)
all angles are in radians

As you can see, you don't need to be a mathematician to perform complicated integrations, and thanks to the computer, numerical integration can be performed to practically any desired degree of accuracy. You can probably think of many more examples where this technique can be used, but here are a few more:

1. Estimation of the weights of irregularly shaped objects.
2. Estimation of the energy consumed by an electric motor operating under a known but varying load cycle.
3. Estimation of the speed of a space capsule which free-falls from a very high altitude to the surface of the earth. □

How readable is your writing?

Fog Index

R. B. Nottingham

On page 17 of the November 1980 issue, I mentioned an article in **The Wall Street Journal** *on readability. Mr. Nottingham has now taken the "Fog Index" described in the article and computerized it. He computed the index for several pieces in* **Creative** *and other publications but excluded his own article. Why not use his program to compute it and compare it to your own writing. Hint: Mr. Nottingham will be tough to beat.*

Recently, *The Wall Street Journal* had an article on clarity of writing and its importance to industry. It cited a "Fog Index," developed by the Gunning-Mueller Clear Writing Institute. The Fog Index is roughly equal to the years of schooling required to read a piece of writing.

It is simply the average sentence length plus the percentage of words having three or more syllables, all multiplied by .4.

Now I am not a game player. I like to solve problems but only if they have some practical application. Programming the Fog Index seemed like a trivial but useful piece of work. Now how do you count the number of syllables in a word? If you would like to make a game of this, solve the problem before reading further or looking at the program.

Let us say, that a syllable always has a vowel. The number of syllables equals the number of vowels. Uhuh, how about the word "pear"? Two vowels but only one syllable. All right, we will not count double vowels; that is, two vowels in succession. Fine. How about the word "piece"? O.K. so we don't count final "e"s. Then consider "syllable." If we don't count the final "e" it would have only two syllables. Sooo—we make an exception for "le," if it is final "le."

After some hours of head scratching I finally arrived at the following theorem: Any word having three or more vowels (a,e,i,o,u,y) has three or more syllables if:

Double vowels are counted as one.

Final "e" and final "ed" are not counted except in the case of "ded," "ted," and "le."

Now I know this theorem is not correct. It doesn't cope with double "e" as in Whoopee! However, I feel that it is close enough for practical purposes. If a reader would like to take a few thousand words of text, mark the words of three or more syllables, then apply the theorem and determine the percentage error, I would be grateful. And amazed. Seriously, any suggestions for significant improvement will be gratefully received.

Now, unless you have a computer with good string handling abilities, including RIGHT$, skip the next few paragraphs and go to the results at the end. If you have a TRS-80, the program is in its language, Microsoft Level II Basic. The short program shown, is liberally sprinkled with remarks, but at the risk of redundancy, we will explain it.

To begin, please note (and applaud) that the program is uniformly numbered in increments of 10 with no missing lines. The peculiar appearance of lines 50-60 is due to the screen format of the TRS-80 plus the addition of returns to accommodate my printer. Line 70 begins the input, line 110 puts it on the screen. Line 100 builds the characters into a string.

Line 120 detects the ASCII for "space" to count words. Line 180 similarly counts sentences. Now the fun of counting syllables begins. Line 140 detects vowels and tentatively counts syllables. V is incremented to detect double vowels. Line 140 decrements the syllable count for two consecutive vowels. Line 160 detects "ded" or "ted" and by jumping to line 180, bypasses 170 which rejects "ed." Note that the sub-strings must include a space since we are only concerned here with word endings. Line 180 similarly bypasses 190 to keep "le" from being rejected under the final "e" rule.

Now things get easier. Line 200 detects word ends via the ASCII for "space," tests for three or more syllables in the word and if found, increments the "long word" count. Line 210 resets the syllable count for the next word. Line 220 tests for 100 or more words (minimum size for an adequate sample) and stops at the end of the sentence.

Line 230 detects the ASCII for "enter" and makes it possible to break off the testing before 100 words so that you may test the program or satisfy your curiosity. Line 250 goes back for the next letter. You will note that many things are happening between one character and the next. If you are a speed typist, you may have to slow your pace. Pretend you are running an old-fashioned Teletype. One note, which you may wish to add to the instructions, is that where a sentence has a clearly independent clause, it should be treated as a separate sentence. Use periods to break it into two sentences.

Now for results. A logical source for samples was the August issue of *Creative Computing*. First, the expert, E.H. Weiss, Ph.D., on page 138. Not bad. Based chiefly on the last section of the column the Fog Index is 14.3. About right for a junior college graduate which is probably a conservative estimate of the average reader. How about the Big Boss himself, on p. 134? Very good, 11.2, perhaps he deserves a raise. Doug Green on p. 26? Very good, 14.8. George Blank, on p. 154? Fine, 11.5. You knew it was a good magazine didn't you?

How does this compare with the rest of the world? The Fort Lauderdale Sunday paper: a wire service writer, 21.3! A local reporter, 11.5; the editorial page editor, 14.6, and Max Rafferty's column, 10.4. A British authority writing to jet pilots,

R. B. Nottingham, The Book Rack, 1619 SE 3rd Court, Deerfield Beach, FL 33441.

16.4, and a random sample from one of our government's Aircraft Accident Reports 22.8! What else would you expect from a bureaucrat?

Since I am not a Junior Citizen it gives me pleasure to note that mature writers seem to write more simply than younger ones. Probably for several reasons; simplicity requires self-assurance. If you are not sure that you really have anything to say, make it impressively difficult to read. Older writers on the average received a better education, including the study of Latin and the classics. Well, "Sic transit gloria Tuesday." Have fun. □

```
10 '        ***   FOG INDEX   ***
20 '    BY  R. B. NOTTINGHAM, LIGHTHOUSE POINT, FLORIDA
30 REM A=LAST LETTER. B= LETTERS. S=SYLLABLES. L=WORD LENGTH
   NS=NUMBER OF SENTENCES. W=NUMBER OF WORDS. LW=NUMBER OF
   LONG WORDS. V=VOWEL COUNTER.
40 CLS
50 PRINT"THIS PROGRAM CALCULATES THE FOG INDEX OF TEXT.  THIS
 IS EQUAL TOTHE GRADE LEVEL OF READER FOR WHICH IT IS SUITABLE.
 IN TYPING TEXT, USE ONLY ONE SPACE AFTER A PERIOD AND IGNORE
 ALL OTHER PUNCTUATION. THE PROGRAM WILL STOP WHEN YOU HAVE";
60 PRINT" A 100 WORD    SAMPLE. IF YOU MAQKE AN ERROR, DO NOT
 BACKSPACE TO CORRECT IT. ITWILL MAKE LITTLE DIFFERENCE IN THE
 SCORE.":PRINT
70 INPUT"PRESS ENTER TO BEGIN";QZ$
80 CLS: DEFSTR A-B : DEFINT C-Z
90 A=INKEY$: IF A="" THEN GOTO 90
100 PRINT A;
110 B=B+A : IF LEN(B) >4 THEN B=RIGHT$(B,4)
120 IF A=CHR$(32) THEN W=W+1  : REM SPACE W=NUMBER OF WORDS
130 IF A=CHR$(46) THEN NS=NS+1:REM 46=PERIOD S=NO OF
 SENTENCES
140 IF A="A"OR A="E" OR A="I" OR A="O" OR A="U" OR A="Y"
 THEN S=S+1 : V=V+1 :ELSE V=0 : GOTO 160  : REM COUNTS VOWELS
 CONSIDERED EQUAL TO SYLLABLES
150 IF V=2 THEN S=S-1 :V=0 :REM ELIMINATES DOUBLE VOWEL
160 IF RIGHT$(B,4)="DED " OR RIGHT$(B,4)="TED " THEN GOTO 180
17- IF RIGHT$(B,3)="ED " THEN S=S-1
180 IF RIGHT$(B,3)="LE "THEN GOTO 200
190 IF RIGHT$(B,2)="E "THEN S=S-1
200 IF A=CHR$(32) AND S>2 THEN LW=LW+1 :REM COUNTS LONG WORDS
 3 OR MORE SYLABLES
210 IF A=CHR$(32) THEN S=0
220 IF W=>100 AND A=CHR$(46) THEN GOTO 260  : REM STOPS
AT END OF SENTENCE WITH 100 OR MORE WORDS.
230 IF A=CHR$(13) THEN GOTO 260  : REM STOPS IF ENTER IS PRESSED
240 A=""
250 GOTO 90
260 PRINT:PRINT"NUMBER OF SENTENCES =";NS
270 PRINT"NUMBER 0F WORDS";W
280 PRINT"WORDS PER SENTENCE =";W/NS
290 PRINT"NUMBER OF LONG WORDS =";LW
300 F=(W/NS +100*LW/W)*.4:REM CALCULATES THE "FOG INDEX"
310 PRINT"FOG INDEX =";F
```

Teaching Concepts in Computer Class

Computer-Assisted Instruction is Not Always a Drill

James W. Hutcheson

James W. Hutcheson, Muscogee County School District, Columbus, GA 31901.

Reprinted from *The Mathematics Teacher*, December 1980 (Volume 73, pages 689-691, 715).

If you say you have computer-assisted instruction in your school, many educators will think of drill work delivered by a computer. Some will think of simulation activities in which students "run city government" or "mix dangerous chemicals." Others who have had computer-programming experiences will think of problem solving. While teaching a statistics class, I discovered still another use of the computer to aid instruction.

Calculators have been used in my introductory course in statistics to facilitate a "number crunching" aspect of the course. Recently, a microcomputer was added to the list of materials supporting the course. Prior to the time the computer assisted in the instruction, I had some

difficulty in teaching a particular concept in sampling. Most students understood that generalizations could be made from the sample to the population when the size of the sample almost equaled the size of the population. However, they were skeptical when the sample represented a smaller portion of population.

Initially, the students examined the information on Table 1 and Table 2.

TABLE 1
Ten Samples of Size 10 Drawn from a Population of Size 1000

Samples	Means	Standard Deviations
10	16.4	11.2
10	19.9	12.0
10	20.1	15.4
10	20.4	8.8
10	23.7	16.4
10	25.5	10.7
10	26.5	17.6
10	27.1	14.0
10	27.3	14.6
10	29.0	15.6
Population	25.5	14.4

Table 1 indicates the means of ten samples of size 10 drawn at random from a population of size 1000. Table 2 indicates the means of ten samples of size 50 drawn from the same population. The students compared the differences in the population mean and standard deviation and the means and standard deviations in the two different size samples. The students found a smaller range of means on Table 2 than on Table 1. They concluded that the trend of a smaller range of sample means with larger and larger samples would continue. My current statistics class was treated to a similar activity, but with more dramatic results.

Instead of distributing the two tables to the students, the class was allowed to choose its own small sample size and large sample size. Summary sheets similar to the tables were generated on the microcomputer and displayed on the screen (see Table 1 in the Appendix for the program). This live demonstration sparked the students to begin speculating on the effects of selecting even more sample sizes. A discussion of random selection and sample size followed that seemed to involve more students than had ever been involved before. The students selected their own sample sizes and speculated on the range of sample means. Occasionally, a large sample size resulted in in an unusually large range of sample means. Initially, students were puzzled at the "error." Then one student remarked, "But, that's just the nature of random numbers!!" This insight helped others to internalize what had happened. I felt most of the benefits of the lesson were over. I was wrong.

Later in the course, we were discussing control groups and experimental groups. Several students remarked that it was possible to select two samples at random from the same population and have the samples be significantly different. They each referred to the computer activity that was previously cited, where samples were selected from the same population at random. The comment was made, "How do you know we didn't accidentally select a large sample and a small sample?" No student had recalled this idea when I used the summary sheets as handouts.

Perhaps computers can assist instruction by developing mathematical concepts that are remembered better through a live demonstration. Other live demonstrations have been used where students predicted what would happen and then let the computer verify (or refute) their predictions.

Usually the development of linear regression has included a brief comment about other predictive procedures. The students understood that the line that best fit a scattergram might not be straight. However, I had no success in developing an intuitive feeling for multiple linear regression. A computer program was used that was successful in developing this intuitive feeling. The program generated three tests A, B, and C and produced correlation coefficients between Test A and C and Test B and C. The user was prompted to pool Test A

TABLE 2
Ten Samples of Size 50 Drawn from a Population of Size 1000

Samples	Means	Standard Deviations
50	21.8	14.8
50	22.5	15.0
50	22.8	13.5
50	26.1	14.7
50	26.3	15.5
50	26.8	13.5
50	27.4	14.9
50	27.5	13.0
50	27.5	14.2
50	28.2	15.3
Population	25.5	14.4

TABLE 1, APPENDIX

```
1 REM .... JIM HUTCHESON, TRS-80, LEVEL II
2 REM .... PROGRAM GENERATES POPULATION OF 1000 RANDOM #S
3 REM .... EACH 1-50. USER SELECTS SAMPLE SIZES
4 REM .... PROGRAM RETURNS MEAN & SD OF 10 SAMPLES ORDERED
5 REM .... ON MEANS ... PROGRAM RETURNS POPULATION MEAN & SD
10 CLS
20 PRINT"PROGRAM IS LOADING 1000 NUMBERS FOR THE POPULATION"
25 PRINT:PRINT
30 PRINT"A SHORT PAUSE ...(ABOUT A MINUTE)..."
35 S = 0:S1 = 0
40 DIM A%(1000),M(10),S(10)
50 FOR J = 1 TO 1000
55 A%(J) = RND(50)
60 PRINT@475,J;A%(J)
70 S = S + A%(J):S1 = S1 + A%(J)↑2
80 NEXT J
90 PRINT@475,"POPULATION LOADED"
100 FOR J = 1 TO 300:NEXT:CLS
110 PRINT"NOW SELECT A SAMPLE SIZE...10 SAMPLES WILL BE SELECTED"
120 PRINT"FROM THE POPULATION AT RANDOM"
130 PRINT:PRINT"THE MEANS AND STANDARD DEVIATIONS WILL BE GIVEN"
140 PRINT"THE COMPUTER WILL ORDER THE SAMPLES ON THEIR MEANS"
150 PRINT:PRINT"HOW LARGE DO YOU WANT EACH SAMPLE TO BE";
160 INPUT N
170 CLS
180 FOR J = 1 TO 10
190 A = 0:B=0
200 FOR K = 1 TO N
210 X = A%(RND(1000))
220 PRINT@470,"SAMPLE";J;"SELECTED";K
230 A=A+X:B=B+X*X
240 NEXT K
250 M(J) = A/N
260 S(J) = SQR((B-A↑2/N)/(N-1))
270 NEXT J
280 CLS:PRINT"NOW ORDERING ON SAMPLE MEANS"
290 FOR J = 1 TO 9
300 FOR K = J+1 TO 10
310 IF M(J)<=M(K) THEN 350
320 H = M(J):M(J) = M(K):M(K)=H
330 H = S(J):S(J) = S(K):S(K) = H
350 NEXT K
360 NEXT J
370 PRINT:PRINT"SAMPLE SIZE      MEAN      STANDARD DEVIATION"
380 FOR J = 1 TO 10:PRINTN,M(J),S(J):NEXTJ
390 PRINT:PRINT"POPULATION MEAN "; S/1000,SQR((S1-S↑2/1000)/1000)
400 PRINT"PRESS <ENTER> TO GENERATE OTHER SAMPLES";
405 INPUT Y
410 CLS:GOTO 150
```

and Test B to form a new test, Test X. The computer indicated a correlation between Test X and Test C. The scores in Test X were formed by the equation

$$X = (\ )A + (\ )B.$$

The user supplied the weightings for Test A and Test B. A copy of the computer program appears in Table 2 in the Appendix. A sample run with coefficients appears in Table 3.

After a few trials, all students seemed to know which test to give more weight in the pooling process. I asked if the weighting had to be positive, and immediately several suggestions for negative weightings were tried by the students.

Following the computer activity, the students reported that the test with the higher correlation should be given more weighting. I asked how the process could be extended to Test A, B, C, and D. A student suggested a formula for generating scores for Test X,

$$X = (\ )A + (\ )B + (\ )C,$$

along with the proper order for weightings on Tests A, B, and C.

I was pleased with the results of the introductory activities and quite pleased that the total time for the computer a tivity and discussions was less than twenty minutes.

TABLE 3

YOU TRIED $X = (3)A + (4)B$

TEST *A*	TEST *B*	TEST *X*	TEST *C*
1	2	11	5
2	1	10	8
3	9	45	15
3	6	33	8
6	16	82	12
0.615	0.703	0.692	

IS THERE A BETTER CORRELATION WITH A DIFFERENT WEIGHTING?

Computer programs such as these have helped me to emphasize statistical thinking and de-emphasize the arithmetic of statistics. Perhaps this shift in emphasis has been the most important aspect of using computers in the classroom to aid instruction. □

TABLE 2. APPENDIX

```
10 REM ... JIM HUTCHESON, TRS-80, LEVEL II
20 REM ... MULTIPLE LINEAR REGRESSION
30 REM ... USER ENTERS WEIGHTINGS - COMPUTER SHOWS PEARSON'S R
40 REM ... USER TRIES TO GET A BETTER R WITH BETTER WEIGHTINGS
50 REM ... PROMPT OF NEGATIVE WEIGHTINGS IS NEEDED
60 REM
70 REM
80 DIM A(5),B(5),C(5),T(5)
90 CLS:PRINT"POOLING SCORES IN A MULTIPLE LINEAR REGRESSION TEST"
100 PRINT:PRINT"TESTS ARE A,B,AND C"
110 PRINT"SAMPLE NUMBERS ARE ALREADY LOADED"
120 S1=0:S2=0:S4=0:S5=0:S6=0:S7=0:S8=0
130 FOR J = 1 TO 5
140 READ A,B,C
150 A(J)=A:B(J)=B:C(J)=C
160 S1=S1+A:S2=S2+A*A
170 S3=S3+B:S4=S4+B*B
180 S5=S5+C:S6=S6+C*C
190 S7=S7+A*C:S8=S8+B*C
200 NEXT J
210 DATA 1,2,5,2,1,8,3,9,15,3,6,8,6,16,12
220 PRINT"PEARSON'S R FOR TEST A AND TEST C"
230 R1 = (S7-S1*S5/5)/SQR((S2-S1↑2/5)*(S6-S5↑2/5))
240 R1=INT(1000*R1+.5)/1000
250 PRINT" R1 = ";R1
260 PRINT:PRINT"PEARSON'S R FOR TEST B AND TEST C"
270 R2 =(S8-S3*S5/5)/SQR((S4-S3↑2/5)*(S6-S5↑2/5))
280 R2=INT(R2*1000+.5)/1000
290 PRINT" R2 = ";R2
300 PRINT:PRINT"SUPPOSE YOU FORMED A NEW SCORE, SAY X, BY"
310 PRINT"POOLING THE SCORES IN TEST A WITH THE SCORES IN TEST B"
320 PRINT"PRESS <ENTER> TO CONTINUE":INPUT J
330 CLS:PRINT"TEST A AND TEST C      TEST B AND TEST C"
340 PRINTTAB(1);R1;TAB(24);R2
350 PRINT:PRINT"TEST X WILL BE FORMED LIKE:  X = (3)A + (4)B
360 PRINT"WHAT WEIGHTING SHOULD BE        ↑        ↑"
370 PRINTTAB(33);"HERE    HERE"
380 PRINT:PRINT"LOOK AT THE TWO CORRELATION COEFFICIENTS AND"
390 PRINT"SUGGEST A WEIGHTING. THE COMPUTER WILL FORM TEST X"
400 PRINT"AND COMPUTE PEARSON'S R FOR TEST X AND TEST C."
410 PRINT:PRINT"PRESS <ENTER> TO CONTINUE":INPUT J
420 CLS:PRINT"R1                    R2"
430 PRINT R1,R2
440 PRINT:INPUT"ENTER WEIGHING FOR TEST A";W1
450 INPUT"ENTER WEIGHING FOR TEST B";W2
460 C1=0:C2=0:C3=0
470 FOR J=1TO5
480 T=W1*A(J)+W2*B(J)
490 T(J)=T
500 C1=C1+T:C2=C2+T*T:C3=C3+T*C(J):NEXTJ
510 CLS
520 R3=(C3-S5*C1/5)/SQR((C2-C1↑2/5)*(S6-S5↑2/5))
530 PRINT"YOU TRIED X = (";W1;")A + (";W2;")B":PRINT
540 PRINT"TEST A";TAB(16);"TESTB";TAB(32);"TEST X";TAB(48);"TEST C"
550 FOR J=1TO5:PRINTA(J),B(J),T(J),C(J):NEXTJ
560 R3=INT(R3*1000+.5)/1000
570 PRINT:PRINTR1,R2,R3
580 PRINT:PRINT"IS THERE A BETTER CORRELATION WITH A DIFFERENT WEIGHTING?"
590 GOTO 440
```

APPENDIX

Note: The TRS-80, Level II Basic uses several functions that are different from other versions of Basic.

1. CLS—clears the screen
2. RND(X)—returns a random number 1 to X if X is greater than 1
3. PRINT@—prints at a position on the screen. The positions are 0-1023

The programs are written in an "endless loop" style. The user is required to depress the BREAK key. This style was chosen over the more conventional END statement or the escape to END question for two reasons:

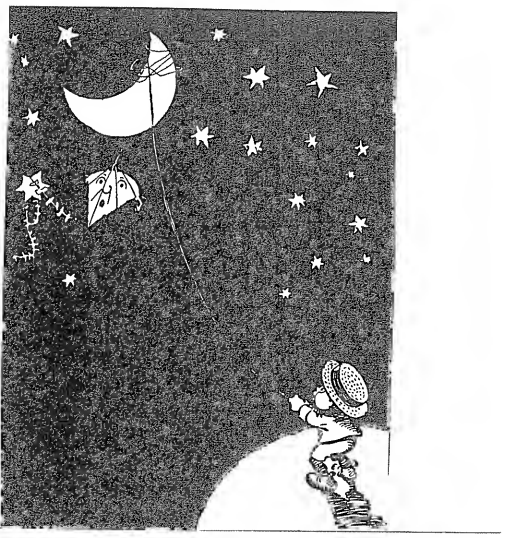
1. The END statement required the author to anticipate the number of attempts a particular user would need. This varied too often to be practical.

2. The escape question would normally appear at the end of the user's sample selection or weighting. The students in the statistics class made too many typing errors to risk an additional user input. Further, the escape question tends to break the train of thought when the students are concentrating on the output of the computer.

A Simulation of Motion

The Vibrating String

William E. Bennett



The realistic simulation of harmonic motion by computer is difficult to accomplish using the Basic language. Although Basic is a powerful tool, the price paid for convenience is slow computation. The only real alternative, if Basic is too slow, is to write the time-critical parts of the program in assembler (machine) language.

The vibration of a string is a kind of motion that can be simulated by the use of a simple algorithm. This can be of value in teaching, if it is realistic, since the concept of standing waves and harmonics is difficult to illustrate by other means. We will first present a Basic program for this simulation and then give the equivalent assembler language program. The latter program is short enough so that one should be able to compose it with T-Bug on the TRS-80 if an assembler is not available.

A simulated string is suspended vertically on the TRS-80 screen. It consists of 48 segments. The top and bottom segments are held in place while the remaining 46 are free to move. Any middle segment is constrained by a force which is proportional to its distance from its nearest neighbors.

The string is "plucked" by moving the top segment to either side a brief time to set the string in motion. The condition of each segment is given by the two variables, X and V. The variable X gives the horizontal position of each segment and the variable V gives the velocity. The Basic program for the vibrating string is as follows:

```
  5 'THE VIBRATING STRING
 10 DEFINT I
 20 DIM X(47),V(47)
 30 CLS:FOR I=0 TO 47
 40 X(I)=63:V(I)=0
 50 SET(X(I),I):NEXT I
 60 A$=INKEY$
 70 IF A$="" GOTO 140
 80 RESET(X(0),0)
 90 IF A$="X" GOTO 30
100 IF A$="R" THEN X(0)=83
110 IF A$="L" THEN X(0)=43
120 IF A$="M" THEN X(0)=63
130 SET(X(0),0)
140 FOR I=1 TO 46
150 AC=X(I-1)+X(I+1)-2*X(I)
160 V(I)=V(I)+AC/2:NEXT I
170 FOR I=1 TO 46:RESET(X(I),I)
180 X(I)=X(I)+V(I)
190 SET(X(I),I):NEXT I
200 GOTO 60
```

William E. Bennett, 708 Diana Ct., Iowa City, IA 52240.

When this program is run, things happen very slowly, too slowly in fact, to give a feeling for what is going on. Plucking is accomplished by typing "R" to move the top segment to the right, "L" to move it to the left, and "M" to move it back to the center. The program is re-started by typing "X." The purpose of listing this program is to illustrate the simple calculations involved and to serve as a comparison to the assembler language program. The assembler program is nearly a direct translation of this Basic program but is more than 100 times faster. It was designed to fit into the upper 1K of a 4K TRS-80 computer. This leaves ample room for T-Bug or for the Basic program we will give later.

To key the assembler program with T-Bug, start with M4C00 and then enter 3E 00 32 AF 4C F3 . . . etc., to the very end (line 1400). Then save it on tape with the PUNCH command (P 4C00 4D1E 4C00 STRG enter).

The program starts at hexadecimal 4C00 (decimal 19456). Those having the use of disk systems and an assembler should move it to, say, 7C00. Line 100 should be changed to "ORG 7C00H," and line 140 to "LD SP,7FFFH."

When the string is "plucked" by pressing the left or right arrow keys, a disturbance is created. This disturbance travels along the string and reflects at each end. The shape of the wave soon changes, giving a motion to the string which is a combination of its various modes of vibration. One can simulate particular modes of vibration by periodic plucking. The string has no friction and will continue to move until the up-arrow is pressed. Also, the way the arithmetic is handled gives roll-around. This is, if the wave motion goes off the screen, it re-appears on the opposite side.

This program gives quite a realistic effect. It will soon be found that plucking always gives a mixture of harmonics. Pure harmonics will be illustrated later. Stringed instruments are designed to damp unwanted harmonics while reinforcing desired ones.

A step by step documentation of the program is given below. We should explain the form of the variables X and V for clarification. Each of these is a two byte number of the form "N.M." One byte, N, is an integer and ranges from 0 to 127, the normal TRS-80 horizontal numbering. The second byte, M, is a fraction. A value of X near the center of the screen would be, in decimal form, 63.0; one halfway between positions 63 and 64 would be 63.5. For each of the 48 values of X and V, two bytes are used. The one at the higher address is N.

LINE	
100	The starting address.
110-120	Set the flag "MOB" to indicate that this is a run-alone program.
130	Disable interrupts. This is needed for disk systems to turn off the clock interrupt. It is good housekeeping anyway.
140	Establish the stack pointer.
150-190	The equivalent of Basic CLS. Clear the screen.
200-240	Set all V values to zero.
250-270	If this is a Basic subroutine skip the next step.

280-350 Set all values of X to 63. This is equivalent to:
FOR I=0 TO 47:X(I)=63:NEXT I

360-370 Call the subroutine SET which will SET all string segments on the CRT screen.

380-480 Read the keyboard to see if the specific keys ←, →, or ↑ are pressed.
↑ = Restart the program or return to the Basic program.
← = Change the value of X(0) to 53.
→ = Change the value of X(0) to 73.

490-500 Initialize the FOR loop. This loop terminates at line 710 and is equivalent to:
```
10 FOR I=1 TO 46
20 V(I)=V(I)+(X(I-1)+X(I+1)
   -2*X(I))/2
30 NEXT I
```
The index register IX initially points to X(0) while IX+96 points to V(0).

510-520 Load X(I-1) into register pair DE.

530-540 Load X(I+1) into register pair HL.

550 Add X(I-1)+X(I+1) giving the sum in register HL.

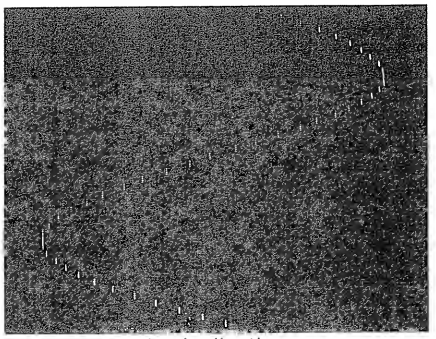
560-570 Load X(I) into register pair DE.

580-590 Multiply X(I) by two.

600 Clear the carry flag since SBC will subtract it.

610 Subtract HL - DE giving the result in HL.

620-630 Divide HL by two.



simple vibration

640-650 Load V(I) into register pair DE.

660 Add HL and DE giving the sum in HL.

670-680 Store the new value of V(I).

690-700 Bump the pointer for X(I) and V(I).

710 This corresponds to the NEXT I.

720-730 Change X(0) to 63 in case it was changed by ← or →.

740-750 Call subroutine SET. If register C is zero, the subroutine will do the following sequence for each string segment:
RESET a string segment
X(I)=X(I)+V(I)
SET the string segment

760 Go back, read the keyboard, and do the next time cycle.

770-790 Check to see if this program is run-alone or is a Basic subroutine. If it is run-alone, start over.

800-810 If this is a Basic program subroutine, then recover the Basic stack pointer and return to that program.

820 The point for a Basic program to enter this subroutine. Save the Basic stack pointer. Good housekeeping again.

830-840 Set the flag "MOB" to indicate that this is now a subroutine of a Basic program.

850 MOB is the flag. If zero, it means that this is a run-alone program. If one, it is a Basic subroutine.

860 BSP is the saved Basic stack pointer.

870-920 These are masks for each of the six pixies in a graphics word.

930 Initialize the register pair DE to contain the CRT line pointer. It now points to horizontal line zero.

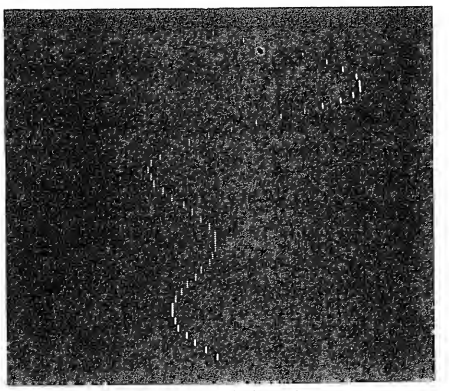
940 Initialize the FOR loop. This loop terminates at line 1200. This loop is equivalent to:
```
10 FOR I=0 TO 47
20 HL=INT(X(I)) (well, sort of)
30 IF C=0 GOTO 70
40 SET(HL,I)
50 IF C=3 GOTO 110
60 C=0:GOTO 110
70 RESET(HL,I)
80 C=1
90 X(I)=X(I)+V(I)
100 GOTO 20
110 NEXT I
```

950 Initialize register IY to point to the first pixie mask.

960 Initialize register IX to point to X(0), IX+96 points to V(0).

970-1000 Load the integer portion of X(I) into the register pair HL. Mask off the sign bit. This will give a number between zero and 127, the normal horizontal graphics range.

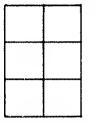
1010 Divide L by two. This gives the number of the graphics word (0 - 63). The remainder, which is now the carry flag, tells which of the



compound vibration

two pixies to SET or RESET.

1 2



1020-1050 Place the correct mask for the particular pixie in register A.

1060 Add the CRT line pointer to HL. The register pair HL now will point to the screen graphics word to be altered.

1070-1080 Are we supposed to SET or are we to RESET? If RESET, go to line 1270.

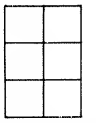
1090-1100 SET the pixie.

1110-1120 If we are to SET only, skip the next line.

1130 Prepare C for the next SET, add, RESET the sequence.

1140-1150 Where are we on the graphics character?

1
4
16



If 16, go to line 1220 to fix things up for the next CRT line.

1160-1170 Increment the mask pointer.

1180-1190 Bump to the next X(I) and V(I).

1200 This corresponds to the NEXT I.

1210 Exit subroutine SET.

1220 Reset the mask pointer.

1230-1250 Bump the CRT line pointer to the next line.

1260 Go to the loop end at line 1180.

1270-1290 RESET the pixie.

1300 Set C to flag the next part of the sequence.

1310 Save the CRT line pointer.

1320-1330 Load V(I) into register DE.

1340-1350	Load X(I) into register HL.
1360	Add X(I)+V(I) giving the result in HL.
1370-1380	Save the new value of X(I).
1390	Recover the saved CRT line pointer.
1400	Go back and SET the new point.
1410	The storage for X values. (4D1F-4D7E)
1420	The storage for V values. (4D7F-4DDE)

The Z-80 instruction set makes programming of this type quite straightforward. The index registers IX and IY are particularly useful as pointers to data, while the other registers are used for arithmetic and logic.

The Best of Both Worlds

The assembler program was written so that it can act as a standalone program or as a subroutine for a Basic program. We can then use a Basic program for the things which it handles best and the Assembler program for things we want to do in a hurry or that are difficult to do with Basic.

The following Basic program generates pure harmonics or mixtures of harmonics which are then transferred to the storage area for the X values of the assembler program. The basic program then calls the assembler program as a subroutine which creates the display.

```
 10 'HARMONICS GENERATOR
 20 DEFINT I-N:PI=3.141593/47:CLS
 30 DIM H(3)
 40 'LOAD STARTING ADDRESS OF
    SUBROUTINE 4CA6
 50 POKE 16526,166:POKE 16527,76
 60 INPUT"WHAT THREE HARMON-
    ICS";H(1),H(2),H(3)
 70 'SET N TO POINT TO X(0)
 80 CLS:N=19743  'THIS IS 4D1F
 90 FOR I=0 TO 47
100 X=63
110 FOR K=1 TO 3
120 X=X+20*SIN(I*PI*H(K))
130 NEXT K
140 'CONVERT X TO THE TWO
    BYTE FORM
150 IA=INT(X)
160 IB=INT((X-IA)*256)
170 IA=IA AND 255
180 'SHOW THE STRING
190 IC=IA AND 127:SET (IC,I)
200 'NOW STORE THE X VALUE
210 POKE N,IB
220 N=N+1
230 POKE N, IA
240 N=N+1:NEXT I
250 'JUMP TO THE SUBROUTINE
260 J=USR(J)  'J IS A DUMMY
270 GOTO 60
```

The program was designed to accept integral values of the harmonics, but one can try others. A single harmonic can be generated by any combination of it with zeroes. For example, 2,2,2 or 2,0,0 gives the second harmonic. Due to the limited resolution, 48 segments, and the speed of the display, the upper limit of the harmonics which can be readily recognized, is about 12. Above 12, some interesting patterns are formed. An important consideration is the Nyquist frequency, harmonic 47. At this frequency there are two segments per sine wave cycle. Above this frequency, no higher harmonics can be represented and the net result is a harmonic of frequency less than the Nyquist frequency.

Combinations of two harmonics can be introduced by combining any of the two numbers with zero. For example, 2,3,0 or 2,2,3. Some of the very high harmonics, such as 45,6,0, give very interesting patterns, but are unstable causing the string to "explode." This can be prevented by reducing the amplitude. To do this, change statement 120 to a lower amplitude. For example:

```
120 X=X+8*SIN(I*PI*H(K))
```

This "explosion" is the result of an overflow of the velocity of segments at high frequency. The assembler program could be revised to avoid this, but

Assembler Programs

```
4C00           00100          ORG    4C00H
4C00 3E00      00110          LD     A,0
4C02 32AF4C    00120 STRA     LD     (MOB),A
4C05 F3        00130          DI
4C06 31FF4F    00140          LD     SP,4FFFH
4C09 21003C    00150 STRB     LD     HL,3C00H
4C0C 11013C    00160          LD     DE,3C01H
4C0F 01FF03    00170          LD     BC,3FFH
4C12 3680      00180          LD     (HL),80H
4C14 EDB0      00190          LDIR
4C16 217F4D    00200          LD     HL,CV
4C19 11B04D    00210          LD     DE,CV+1
4C1C 015F00    00220          LD     BC,95D
4C1F 3600      00230          LD     (HL),0
4C21 EDB0      00240          LDIR
4C23 3AAF4C    00250          LD     A,(MOB)
4C26 CB47      00260          BIT    0,A
4C28 200E      00270          JR     NZ,STRD
4C2A 211F4D    00280          LD     HL,CX
4C2D 0630      00290          LD     B,4BD
4C2F 11003F    00300          LD     DE,3F00H
4C32 73        00310 STRC     LD     (HL),E
4C33 23        00320          INC    HL
4C34 72        00330          LD     (HL),D
4C35 23        00340          INC    HL
4C36 10FA      00350          DJNZ   STRC
4C3B 0E03      00360 STRD     LD     C,3
4C3A CDB84C    00370          CALL   SET
4C3D 3A4038    00380 STRE     LD     A,(3840H)
4C40 E668      00390          AND    104D
4C42 2811      00400          JR     Z,STRH
4C44 CB5F      00410          BIT    3,A
4C46 2051      00420          JR     NZ,EXIT
4C4B CB6F      00430          BIT    5,A
4C4A 2004      00440          JR     NZ,STRF
4C4C 3E49      00450          LD     A,73D
4C4E 1802      00460          JR     STRG
4C50 3E35      00470 STRF     LD     A,53D
4C52 32204D    00480 STRG     LD     (CX+1),A
4C55 DD211F4D  00490 STRH     LD     IX,CX
4C59 062E      00500          LD     B,46D
4C5B DD5E00    00510 STRI     LD     E,(IX)
4C5E DD5601    00520          LD     D,(IX+1)
4C61 DD6E04    00530          LD     L,(IX+4)
4C64 DD6605    00540          LD     H,(IX+5)
4C67 19        00550          ADD    HL,DE
4C68 DD5E02    00560          LD     E,(IX+2)
4C6B DD5603    00570          LD     D,(IX+3)
4C6E CB23      00580          SLA    E
4C70 CB12      00590          RL     D
4C72 A7        00600          AND    A
4C73 ED52      00610          SBC    HL,DE
4C75 CB2C      00620          SRA    H
4C77 CB1D      00630          RR     L
4C79 DD5E62    00640          LD     E,(IX+98D)
4C7C DD5663    00650          LD     D,(IX+99D)
4C7F 19        00660          ADD    HL,DE
4C80 DD7562    00670          LD     (IX+9BD),L
4C83 DD7463    00680          LD     (IX+99D),H
4CB6 DD23      00690          INC    IX
4C88 DD23      00700          INC    IX
4C8A 10CF      00710          DJNZ   STRI
4C8C 3E3F      00720          LD     A,63D
4C8E 32204D    00730          LD     (CX+1),A
4C91 0E00      00740          LD     C,0
4C93 CDB84C    00750          CALL   SET
4C96 C33D4C    00760          JP     STRE
4C99 3AAF4C    00770 EXIT     LD     A,(MOB)
4C9C CB47      007B0          BIT    0,A
4C9E CA094C    00790          JP     Z,STRB
4CA1 ED7BB04C  00800          LD     SP,(BSP)
4CA5 C9        00810          RET
4CA6 ED73B04C  00820 ENTRY    LD     (BSP),SP
4CAA 3E01      00830          LD     A,1
4CAC C3024C    00840          JP     STRA
4CAF 00        00850 MOB      DEFB   0
4CB0 0000      00860 BSP      DEFW   00
4CB2 01        00B70 VPX      DEFB   1
4CB3 02        00B80          DEFB   2
4CB4 04        00890          DEFB   4
4CB5 08        00900          DEFB   B
4CB6 10        00910          DEFB   16D
4CB7 20        00920          DEFB   32D
4CBB 11003C    00930 SET      LD     DE,3C00H
4CBB 0630      00940          LD     B,4BD
4CBD FD21B24C  00950          LD     IY,VPX
4CC1 DD211F4D  00960          LD     IX,CX
4CC5 DD7E01    00970 SETA     LD     A,(IX+1)
4CC8 E67F      00980          AND    7FH
4CCA 6F        00990          LD     L,A
4CCB 2600      01000          LD     H,0
4CCD CB3D      01010          SRL    L
4CCF 3805      01020          JR     C,SETB
4CD1 FD7E00    01030          LD     A,(IY)
4CD4 1B03      01040          JR     SETC
4CD6 FD7E01    01050 SETB     LD     A,(IY+1)
4CD9 19        01060 SETC     ADD    HL,DE
4CDA CB41      01070          BIT    0,C
4CDC 2B24      01080          JR     Z,SETG
4CDE B6        01090          OR     (HL)
4CDF 77        01100          LD     (HL),A
4CE0 CB49      01110          BIT    1,C
4CE2 2002      01120          JR     NZ,SETD
4CE4 0E00      01130          LD     C,0
4CE6 FDCB0066  01140 SETD     BIT    4,(IY)
4CEA 200B      01150          JR     NZ,SETF
4CEC FD23      01160          INC    IY
4CEE FD23      01170          INC    IY
4CF0 DD23      011B0 SETE     INC    IX
4CF2 DD23      01190          INC    IX
4CF4 10CF      01200          DJNZ   SETA
4CF6 C9        01210          RET
4CF7 FD21B24C  01220 SETF     LD     IY,VPX
4CFB 214000    01230          LD     HL,64D
4CFE 19        01240          ADD    HL,DE
4CFF EB        01250          EX     DE,HL
4D00 18EE      01260          JR     SETE
4D02 2F        01270 SETG     CPL
4D03 A6        012B0          AND    (HL)
4D04 77        01290          LD     (HL),A
4D05 0E01      01300          LD     C,1
4D07 D5        01310          PUSH   DE
4D08 DD5E60    01320          LD     E,(IX+96D)
4D0B DD5661    01330          LD     D,(IX+97D)
4D0E DD6E00    01340          LD     L,(IX)
4D11 DD6601    01350          LD     H,(IX+1)
4D14 19        01360          ADD    HL,DE
4D15 DD7500    01370          LD     (IX),L
4D18 DD7401    01380          LD     (IX+1),H
4D1B D1        01390          POP    DE
4D1C C3C54C    01400          JP     SETA
0060           01410 CX       DEFS   96D
0060           01420 CV       DEFS   96D
0000           01430          END
00000 TOTAL ERRORS
```

we wished to keep it as short as possible.

Three harmonics can be combined in any order. For example: 3,4,5 or 6,5,4. Again, for high harmonics such as 44,45,46, reduce the amplitude to about 8. To give a straight string, enter 0,0,0.

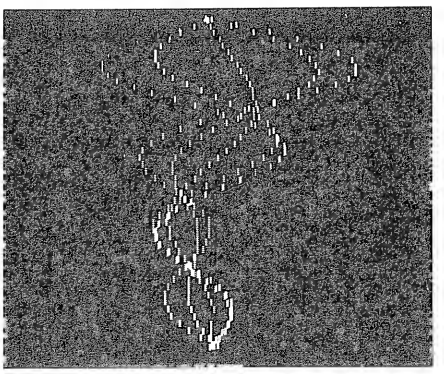
Some persons who have run this program have tried to "break" the string by stimulating large oscillations of the first harmonic. They did not succeed. Try the following:

```
120 X=X+100*SIN(I*PI*H(K))
```

and then enter 1,1,1.

It is up to the programmer's imagination as to what functions to try. For example:

```
120 X=X+20*(SIN(I*PI*H(K))↑2)
                or
115 EX=(1-I)/10
120 X=X+40*(SIN(I*PI*H(K))*
  (2.71828↑EX)
```



cosmic twang

Try any function where X(0) and X(47) are nearly equal to 63. You can devise a kind of mobile screen-graphics art.

To prepare a tape, first produce the assembler program with T-Bug or an assembler. Store it on tape with a name such as STRG. Then copy the above Basic program and save it following the Assembler program, using a name such as "S." On power-up of a Level II system do the following:

```
MEMORY? 19455                    enter
READY
>system                          enter
*? strg                          enter
*?                               enter
?SN ERROR
READY
>cload "S"                       enter
READY
>run                             enter
```

To run the program as a stand-alone program do the following:

```
MEMORY?                          enter
>system                          enter
*? strg                          enter
*?/19456                         enter
(If saved by T-Bug  *?/  enter)
```

□

A New Approach to Teaching Music

Computer Aided Sight Reading

David B. Clark Collette J. Wilkins D. T. Tuma

Introduction

Computer-based instruction (CBI) for training in music has been to a large extent implemented on large computer systems,[1] or on sophisticated minicomputer systems.[2] It is now well established that certain musical skills can be effectively strengthened through drill and practice exercise via CBI.[3] However, not all educational centers have access to such powerful computer systems; and if they do, the cost is too high for the typical class. In addition, computer centers usually do not allow the connection of user devices — such as a "music box" — to the computer.

D. Clark, C. Wilkins, D. Tuma, Carnegie-Mellon University, Pittsburg, PA 15213.

Recent technological developments in electronics have brought to the consumer computational capability at low cost in the form of the small computer. In the same manner, the cost of a dedicated, standalone computer system of acceptable power has come within the financial reach of most educational institutions. It is expected that this trend of availability of more computational power per unit cost will continue for many years. Such circumstances should make the use of small computers in education an irresistable force.

This article describes an implementation of a fixed-do sight reading exercise for music students on a low cost, stand alone small computer system. Although the application is specific, the approach has general features of value to people interested in applying CBI on small computers.

When considering the use of small computers for CBI, certain general objectives need to be attained to ensure the success of the implementation:

- The system should be easy to use and not require any computer knowledge on the part of the student.
- The system should be inexpensive and portable.
- The system should provide interactive operation with the user.
- The system should have some graphical capabilities.
- The system should provide the user with immediate feedback as to the user's level of performance.
- The overall experience to the student should be positive so as to in-

duce the student to use the system voluntarily.

- The system should be flexible enough to allow future implementations of other advanced features as the need arises or as the technology advances.

Specific features that were sought to ensure a successful implementation of the sight reading exercise on the computer consist of the following:

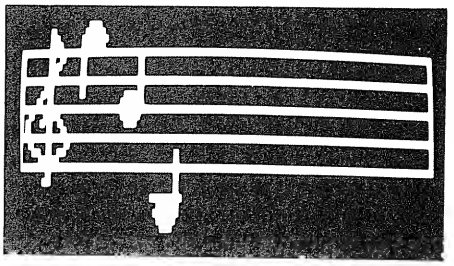


Photo 1

A graphical display of a musical score in progress. The staff, treble clef and three out of ten notes have been displayed. A leger line appears on the third note.

- Graphical display of the musical notes
- Generation of different successive exercises
- Exercises in many clefs
- Inclusion of leger lines
- Timing of student performance
- Clear indication of the student's mistakes
- Possibilities for expansion into other areas of musical training, such as ear training.

Capabilities

The set objectives are found to be satisfied by a Radio Shack TRS-80 computer system equipped with a Level II Basic interpreter and 16K RAM memory. This system, selling below $1,000, consists of a typewriter keyboard, a cathode ray tube (CRT) display and a cassette recorder for storage and retrieval of programs. It is expected that in the near future even more appropriate systems may become available at the same price.

The exercise on the TRS-80 is initiated by the user typing RUN on the keyboard. From then on the interactive features of the program provide the user with all the information needed and options available to proceed in the exercise.

The events of a typical exercise proceed in the following order:

1. An introductory message appears on the CRT. This message describes to the user the lesson to follow and outlines the tasks the user will be expected to perform. The user is asked to press the ENTER key on the keyboard when ready to continue.
2. The list of clefs (treble, bass, soprano, tenor, alto, mezzo-soprano and baritone) is displayed. The user is asked to make a choice by pressing a number corresponding to the desired clef. If the user presses an inappropriate button, an explanatory message is printed and another opportunity given.
3. Following the selection of a clef, the user is asked to indicate by pressing Y or N whether notes on leger lines are to be included in the drill. Again, an appropriate response causes a helpful prompt to be shown on the screen before the question is repeated.
4. Following an acceptable response to the leger line query, the video display is cleared. A five-line staff is drawn, and the clef selected by the student is then superimposed. Ten randomly generated notes are drawn on the staff. Photo 1 shows this in progress for a case in which the user selected treble clef and chose to include notes on leger lines.

All graphics functions are designed as subroutines and are therefore easily usable for other lessons.

5. When all ten notes have been drawn, the message PRESS ENTER WHEN YOU'RE READY TO GO! is shown beneath the score. The user is then given an opportunity to study the score before attempting to sight-read it against a clock. This point in the lesson is illustrated in Photo 2 for a case in which the user selected bass clef and chose not to include leger lines.
6. As soon as the user presses the ENTER key, the previous message is replaced by THE NOTE IS? A timer which records the total time used by the student to identify the ten notes is now initiated. The user presses the key corresponding to the leftmost note on the screen (the keyboard's numeric keys have an overlay marked DO, RE, etc.). If the student's response is correct, the syllable is momentarily shown after the question mark, and the note is erased. The student then proceeds to the next note. Photo 3 shows this for a lesson involving the soprano clef. If the student's answer is incorrect, the message NO, <syllable student gave> IS WHAT'S FLASHING NOW is shown on the screen. A small block on the staff which corresponds to the syllable given by the student is then flashed for a brief time. Following this, the original message THE NOTE IS? is reprinted and the student is given another opportunity to identify the note. In Photo 4, this feature is demonstrated for a lesson on the tenor clef in which the user incorrectly identified a "re" as "sol."

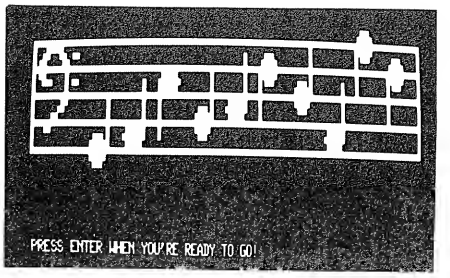


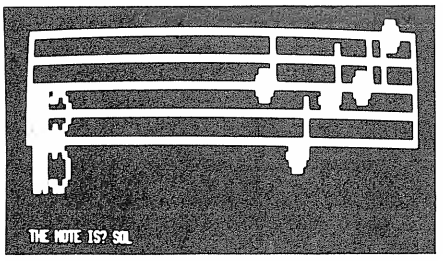
Photo 2

A complete graphical display of a musical score in bass clef without leger lines. The user is being prompted to start the exercise.

7. The lesson proceeds in this fashion until the student has correctly identified all 10 notes. Beneath the empty staff, the screen then displays the student's performance results. Numeric results for accuracy and speed are provided, and a message describing tempo is printed. The message YOUR TEMPO WAS GOOD! indicates that the user did not require more than two seconds to identify any single note. For example, a student who made one error in identifying the ten notes, identified all 10 in 15 seconds, and needed three seconds to identify a particular note, would see the following message: YOU GOT ALL 10 NOTES in 11 TRIES FOR A SCORE OF 90. YOUR AVERAGE RESPONSE TIME FOR EACH NOTE WAS 1.5 SECONDS. NEXT TIME, TRY TO MAINTAIN A BETTER TEMPO. After this message is printed, the user is asked to indicate whether another drill is desired. Typing Y will repeat the drill with ten new randomly generated notes, while typing N ends the lesson.

Implementation

The software for the sight-reading lesson is written in TRS-80 Level II Basic, which allows the use of machine language subroutines when faster execution of particular tasks (such as graphics) is desirable. The program occupies 9.2K of RAM space, requiring slightly more at execution. For versatility in future programming, all graphics functions are designed as subroutines and are therefore easily usable for other lessons. These subroutines are machine-dependent and would have to be completely rewritten if another microcomputer were to be used. The subroutines include subroutine STAFF, which draws a five-line staff, CLEF(C), which draws a specified clef, NOTE (H, V), which draws a note at a staff location specified by its horizontal and vertical coordinates, and ERASE (H, V), which erases a specified note. All but ERASE are written using TRS-80 Basic graphics commands, which offer considerable flexibility but execute slowly. Drawing the full score takes approximately 12 seconds. Once the drill has begun, however, notes must be erased quickly, or a moderately adept student could find the system unpleasantly slow. Subroutine ERASE is therefore written to directly access video memory, thus providing a worst-case execution speed of 165ms per note erased.



A graphical display of a musical score in the soprano clef. The first five notes have been correctly identified and duly erased. The user has just correctly identified the sixth note as a sol, and the note is about to be erased.

The low resolution of the TRS-80 graphics (128H by 48V) combined with the need for relatively round notes severely limits the number of notes which can be displayed at once. Given the presence of a clef and reasonable spacing between notes, only ten can be displayed simultaneously (Photo 2). The vertical display range extends to two leger lines above and below the staff (17 notes in all). This leaves several lines at the bottom of the display for text.

An important design constraint of the software is the ease of operation for the untrained user. One example of this is response prompting coupled with complete error trapping. When a query is made of the user, an explanation is always provided of the type of answer expected (Y or N, a number from 1-7, etc.). An inappropriate or mistyped response from the user does not crash the system, but generates additional helpful prompting and subsequent repetition of the question. Even if the program were erased from memory, it could be reloaded from cassette with a minimum of effort. Another implementation of this design constraint is the writing of special keyboard strobing routines which eliminate the need for the user to type a carriage return following each input. As soon as a key is depressed, appropriate action begins. Since it is desirable to minimize the technical details users are forced to remember, these techniques should be of value to CBI applications in general.

The software can be broken down into the following major routine blocks:

1. Initialization
   a. internal housekeeping
   b. seeding of random number generator
   c. introduction printed on screen
2. Establishing Drill Parameters
   a. query user for desired clef
   b. query user concerning leger lines
3. Drawing the score
   a. draw staff
   b. draw specified clef
   c. draw ten randomly generated notes
   d. wait until user is ready to continue
4. Conducting the drill
   a. accept user inputs
   b. if correct, erase note and proceed
   c. if incorrect, provide feedback and repeat
   d. monitor elapsed time
5. Feedback to user upon completion.
   a. accuracy
   b. speed
   c. tempo
   d. provide option to repeat drill with new notes.

Evaluation

As of this writing, the sight-reading lesson has been in use for several months. Students deficient in sight-reading skills who have used the system are uniformly enthusiastic about the help it provides. The lesson's immediate feedback feature offers a clear advantage over traditional solitary practice, where a student could unknowingly make endless incorrect readings. Students have in general been receptive to the idea of computer drill work. Their suggestions for program improvement have been incorporated whenever possible.

Classroom results have verified that beginning students are progressing much faster than usual in learning this basic skill. Furthermore, students using the machines are often intrigued with the reading of the less common clefs, with the more advanced students using the system to sharpen their skills in these areas. Class time has been freed from the need for drill on clef reading allowing the instructor to cover more stimulating topics. Finally, this simple introduction to computers in music education has stimulated both student and faculty interest in more sophisticated applications within the field.

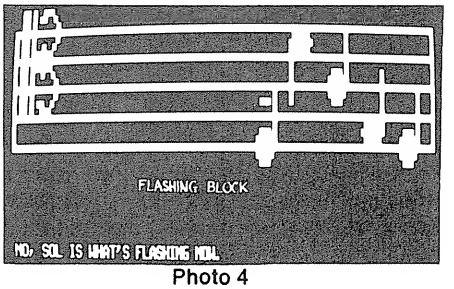


Photo 4

A graphical display of a musical score in the tenor clef. The first five notes have been correctly identified and duly erased. The user has just incorrectly identifed the sixth note. The user is being told of the error and a flashing pointer indicates where the wrong choice falls on the staff.

Future Plans

Current work with the TRS-80 system is centered around the development of a music box which can provide pitches under computer control. At the fundamental level, the concept of a music box emitting pitches synchronized to a visual output is one that truly improves upon the classroom situation, since a teacher cannot simultaneously play a note and write it on the blackboard. Of particular interest is drill in dictations, in which

the student is required to identify the pitches just heard. A music box with several voices could provide drill in dictations of several parts.

A critical weakness of the TRS-80 system is its limited graphics capability. For more sophisticated applications, at least two staffs, each containing considerably more than ten notes, must be shown along with text. Also, with better resolution, rhythm notation could be incorporated. Since harmony and counterpoint follow certain well-defined rules of composition, with improved graphics a whole library of musical drills could conceivably be developed and stored on tape. This is not possible on the TRS-80.

Conclusion

We have been successful in implementing a very useful sight-reading exercise for music students on a small computer system, the Radio Shack TRS-80, that is affordable by most schools. The response of the students to the use of the system has been enthusiastic. A qualitative evaluation of student learning on the system shows very positive results. The system has now been in use for several months without any breakdown or major malfunction.

This experience encourages us to proceed with the implementation of other exercises, such as music dictation, on the small computer. However, we perceive a need for a small computer system that has finer resolution graphics than the TRS-80. All indications are that the market should in the near future have such a system widely available at a cost of $1,000 or less.

ACKNOWLEDGEMENT

The authors acknowledge the financial support of the Research and Development Fund for Education of Carnegie-Mellon University and Akram Midani, Dean of the College of Fine Arts. □

References

1. S. G. Smith and B. A. Sherwood, "The Educational Uses of the PLATO System," **Science,** Vol. 192 (1976) p. 344.

2. J. Margarrell and C. V. Bunderson, "Computerized Education: Time to Sink or Swim," Chronicle of Higher Education, Vol. 12 (1976) p. 1.

3. F. T. Hofstetetter, "Music Dream Machines: New Realities for Computer-Based Musical Instruction," Creative Computing, (March/April 1977) p. 50.


```
10 REM***INITIALIZATION***
20 CLEAR 50:DEFINT A-Z
30 DIM N(10),T(10)
40 RANDOM
45 CLS:PRINT CHR$(23):PRINT STRING$(32,"*"):PRINT"CARNEGIE-"; "MELLON UNIVE
RSITY":PRINT:PRINT"MUSIC EDUCATION SYSTEM": PRINT:PRINT STRING$(32,"*"):F
OR J=1TO2000:NEXT J
48 DIM B$(7,2)
49 B$(1,1)="DO":B$(2,1)="RE":B$(3,1)="MI":B$(4,1)="FA"
50 B$(5,1)="SOL":B$(6,1)="LA":B$(7,1)="SI":B$(1,2)="C"
51 B$(2,2)="D":B$(3,2)="E":B$(4,2)="F":B$(5,2)="G":B$(6,2)="A"
52 B$(7,2)="B"
55 DEFUSR1=&HBB63
60 REM  INSTRUCTIONS TO STUDENT
70 REM*************************************************************
80 CLS
90 PRINT"WELCOME TO SOLFEGE LESSON #1!  IN THIS LESSON,"
100 PRINT"YOU'LL BE IDENTIFYING NOTES ON A CLEF OF YOUR"
110 PRINT"CHOICE.  AS YOU IDENTIFY EACH NOTE FROM LEFT TO"
120 PRINT"RIGHT, IT WILL DISAPPEAR FROM THE SCREEN.  TO ENTER"
130 PRINT"YOUR RESPONSE, PRESS THE KEY WITH THE SYLLABLE FOR"
140 PRINT"THAT NOTE.  TRY TO ANSWER BOTH QUICKLY AND"
150 PRINT"ACCURATELY, AND STILL MAINTAIN AN EVEN TEMPO!"
160 PRINT:PRINT"WHEN YOU'RE READY TO PROCEED, PRESS ENTER"
170 A$=INKEY$:IF A$="" GOTO 170
173 CLS:PRINT"WOULD YOU LIKE TO USE SYLLABLES OR LETTERS?"
174 PRINT"(TYPE S FOR SYLLABLES, L FOR LETTERS.)"
175 A$=INKEY$:IF A$="" THEN 175
176 IF A$="S" THEN X=1: GOTO180
177 IF A$="L" THEN X=2: GOTO 180
178 PRINT"USE S OR L, PLEASE." :GOTO 175
180 CLS:PRINT"YOU MAY CHOOSE FROM ANY OF THE FOLLOWING CLEFS:"
190 PRINT"1. TREBLE":PRINT"2. BASS":PRINT"3. SOPRANO"
200 PRINT"4. TENOR":PRINT"5. ALTO":PRINT"6. MEZZO-SOPRANO"
210 PRINT"7. BARITONE"
230 PRINT@576,"WHICH CLEF (1-7) WOULD YOU LIKE TO WORK WITH?";
240 A$=INKEY$:IF A$="" GOTO 240 ELSE IF ASC(A$)<32PRINT@621,A$;
250 C=VAL(A$):IF C<0 AND C>8 GOTO 290 ELSE PRINT
260 PRINT"I CAN'T UNDERSTAND THAT!  USE A 1 FOR TREBLE,"
270 PRINT"A 2 FOR BASS, AND SO ON."
280 PRINT@621," ";:GOTO 230
290 PRINT@768,"WOULD YOU LIKE TO INCLUDE LEGER LINES ";
300 PRINT"(TYPE Y OR N)?";
310 A$=INKEY$:IF A$="" GOTO 310 ELSE IF ASC(A$)>32 PRINT@820,A$
320 IF A$="Y" OR A$="N" GOTO 340
330 PRINT@832,"USE Y OR N, PLEASE.":PRINT@820," ";:GOTO 290
340 IF A$="Y" THEN L=1 ELSE L=0
350 REM*************************************************************
360 REM  GENERATION AND DRAWING OF A MOTIF
370 REM*************************************************************
380 CLS:GOSUB 5000:GOSUB 5100
390 FOR J=1 TO 10
400 N(J)=30+6*L-2*RND(11+6*L)
405 IF J=1 GOTO 410 ELSE FOR K=1 TO J-1:IF N(K)=N(J) GOTO 400 ELSE NEXT K
410 NEXT J
420 FOR J=1 TO 10:H=9+J*11:V=N(J):GOSUB 5200:NEXT J
430 REM***CALCULATE CLEF DISPLACEMENT FACTOR***
440 ON C GOTO 450,460,470,480,490,500,510
450 D=-4:GOTO 545
460 D=-6:GOTO 545
470 D= 2:GOTO 545
480 D=-3:GOTO 545
490 D=-5:GOTO 545
500 D=-7:GOTO 545
510 D=-8
520 REM*************************************************************
530 REM  BEGIN QUIZ OF STUDENT
540 REM*************************************************************
545 PRINT@932,"PRESS ENTER WHEN YOU'RE READY TO GO!"
546 A$=INKEY$:IF A$="" GOTO 546 ELSE PRINT@832,CHR$(30)
550 PRINT@832,CHR$(30):H=20:T(0)=0
560 V=N(INT(H/11)):T(INT(H/11))=0
565 T(0)=T(0)+1
570 REM***FETCH STUDENT'S ANSWER FROM KEYBOARD***
580 PRINT@832,"THE NOTE IS?"
590 A$=INKEY$:IF A$="" GOTO 610
600 T(INT(H/11))=T(INT(H/11))+1:GOTO 590
610 G=VAL(A$):IF G<1 OR G>7 GOTO 590
700 PRINT@845,B$(G,X)
710 REM*************************************************************
720 REM  CHECK STUDENT'S ANSWER
730 REM*************************************************************
740 BF=50
750 FOR OC=0 TO 42 STEP 14
760 PF=38-2*D-2*G-OC
```

```
770 IF ABS(V-PF)<ABS(V-BF) AND PF>=0 AND PF<=38 THEN BF=PF
780 NEXT OC
790 IF BF<>V GOTO 850
800 REM***ANSWER WAS CORRECT***
810 GOSUB 5200:PRINT@844,CHR$(30)
820 H=H+11
830 IF H<127 GOTO 560 ELSE GOTO 1010
840 REM***ANSWER WAS INCORRECT***
850 PRINT@832,"NO, ";B$(G,X);" IS WHAT'S FLASHING NOW."
  : F=POINT(H,BF)
    FOR J=1 TO 7
    SET(H-1,BF):SET(H,BF):SET(H+1,BF)
    FOR K=1 TO 50:NEXT K
900 RESET(H-1,BF):RESET(H,BF):RESET(H+1,BF)
910 FOR K=1 TO 50:NEXT K
920 NEXT J:A$=INKEY$
930 A$=CHR$(31):PRINT@832,A$
940 IF F=0 GOTO 565
950 SET(H-1,BF):SET(H,BF):SET(H+1,BF)
960 GOTO 565
970 REM*************************************************************
980 REM  MOTIF COMPLETED - EVALUATE SCORES
990 REM*************************************************************
1000 REM***ACCURACY GRADE***
1010 PRINT@704,"YOU GOT ALL 10 NOTES IN";T(0);"TRIES FOR ";
1020 PRINT"A SCORE OF";INT(1000/T(0))
1030 REM***RESPONSE TIME CALCULATION***
1040 T(0)=0
1050 FOR J=1 TO 10
1060 T(0)=T(0)+T(J)
1070 NEXT J
1080 RT!=INT(T(0)/2.57)/100
1090 PRINT"YOUR AVERAGE RESPONSE TIME FOR EACH NOTE WAS";
1100 PRINT RT!;"SECONDS"
1110 REM***TEMPO EVALUATION***
1120 FOR J=2 TO 10
1130 IF T(J)>T(1) THEN T(1)=T(J)
1140 NEXT J
1150 IF (T(1)/25)>=2 PRINT"NEXT TIME, TRY TO MAINTAIN A ";                "BETTER T
EMPO." ELSE PRINT"YOUR TEMPO WAS GOOD!"
1160 PRINT@896,"LIKE ANOTHER TRY (TYPE Y OR N)?";
1170 A$=INKEY$:IF A$="" GOTO 1170 ELSE PRINT@928,A$;
1180 IF A$="Y" GOTO 1250 ELSE IF A$="N" END
1190 PRINT@960,"USE Y OR N, PLEASE";
1200 PRINT@928," ";
1210 GOTO 1160
1250 PRINT@704,CHR$(31)
1253 PRINT@896,"WOULD YOU LIKE TO USE THE SAME CLEF(TYPE Y OR ";      "N)?";
1254 A$=INKEY$:IF A$="" GOTO 1254
1255 IF A$="N" THEN 100
1256 IF A$<>"Y" THEN PRINT@960,"PLEASE USE Y OR N";:GOTO 1253
1257 PRINT@704,CHR$(31)
1260 IF C=3 GOSUB 5100
1270 GOTO 390
5000  REM SUBROUTINE *STAFF* DRAWS A STAFF
5010 FORB=26TO10STEP-4:FORA=0TO127:SET(A,B):NEXTA:NEXTB
5020 FORB=11TO25:SET(0,B):SET(127,B):NEXTB:RETURN
5100  REM SUBROUTINE *CLEF(C)* DRAWS A CLEF
5105  B=18:IF C=3 B=26
5110  IF C=4 B=14
5115  IF (C=6)+(C=7) B=22
5120  ON C GOTO 5125,5150,5180,5180,5180,5180,5150
5125  SET(7,28):FORB=28TO7STEP-1:SET(8,B):NEXTB:SET(9,8):SET(10,9)
5130  FORA=0TO9:SET(11-A,11+A):NEXTA:SET(2,21):SET(3,23):SET(4,24)
5135  SET(5,24):SET(6,25):SET(7,25):SET(9,25):SET(10,25):SET(11,24)
5140  SET(12,24):SET(13,23):SET(12,21):SET(11,20):SET(10,19)
5145  SET(9,19):SET(7,19):SET(6,20):RETURN
5150  SET(5,B-2):SET(4,B-2):SET(3,B-2):SET(3,B-3):SET(2,B-3)
5155  SET(2,B-5):SET(3,B-5):SET(3,B-6):SET(4,B-6):SET(4,B-7)
5160  SET(5,B-7):SET(6,B-7):SET(7,B-7):SET(8,B-7):SET(8,B-6)
5165  SET(8,B-6):SET(9,B-6):SET(9,B-5):SET(10,B-5):SET(10,B-3)
5170  SET(10,B-2):SET(10,B-1):SET(10,B+1):SET(9,B+1):SET(9,B+2)
5175  SET(8,B+2):SET(7,B+3):SET(6,B+3):SET(5,B+5):SET(4,B+5)
5177  SET(12,B-6):SET(13,B-6):SET(12,B-2):SET(13,B-2):RETURN
5180  FORA=-7TO7:SET(2,B+A):SET(3,B+A):SET(5,B+A):NEXTA
5182  SET(7,B-6):SET(8,B-6):SET(8,B-7):SET(9,B-7):SET(10,B-7)
5184  SET(11,B-6):SET(12,B-5):SET(12,B-3):SET(11,B-2):SET(10,B-1)
5186  SET(9,B-1):SET(8,B-1):SET(7,B-1):SET(6,B-1):SET(6,B+1)
5188  SET(7,B+1):SET(8,B+1):SET(9,B+1):SET(10,B+1):SET(11,B+2)
5190  SET(12,B+3):SET(12,B+4):SET(12,B+5):SET(11,B+6):SET(10,B+7)
5192  SET(9,B+7):SET(8,B+7):SET(8,B+6):SET(7,B+6):SET(8,B-2)
5194  SET(8,B+2):RETURN
5200 '***********************************************************
5210 'SUBROUTINE DRAWER - DRAWS OR ERASES NOTES
5220 '
5230 'REQUIRED INPUTS: H AND V, WHERE:
5240 ' 19 < H < 120 , H-20 A MULTIPLE OF 11
5290 ' 1 < V < 35 , V A MULTIPLE OF 2
5300 '************************************************************
5310 POKE &HBB70, 5*((H-20)/11) 'HORIZONTAL DISPLACEMENT
5320 CALL=USR1(&HBBF3+22*(V/2-1)) : RETURN
```

Chapter X
TRS-80 Strings

The following chapter provides you with a glimpse of some of the latest material presented in the *Creative Computing* monthly column *TRS-80 Strings* by Steve Gray.

Splitting Strings • Pocket Computer Peripherals

May, 1982

Splitting Strings

There is a way of figuring out the location of an address on a Manhattan avenue by performing a simple mathematical operation on the number. You simply "cancel the last figure of the number, divide the remainder by 2, and add or subtract the given key number," according to the instructions found on the back of some New York business card.

To locate the Chemical Bank at 2260 Broadway, you divide 226 by 2, which gives 113, and then subtract key-number 31, which gives 82. The bank is actually at the corner of Broadway and 81 St., but 82 is close enough.

How would you write a program to provide cross-street numbers for the avenues of Manhattan? For instance, find the cross-streets for:

123 Broadway
234 Amsterdam
345 Central Park West

There are 30 avenues in Manhattan with key numbers, but these three will do for examples. One involves a simple calculation; the other two are a little more complicated.

According to the Radio Shack manual, various operations can be performed on the number part of the street address. The chapter on strings shows how to use VAL in determining if a street number is odd or even, which is a fairly simple operation. But is there a way to peel off the number and leave the street name to be operated on?

I called Radio Shack's Customer Service number. The woman suggested ways that would work, but only if the number part were always the same length.

But the number can vary from one to four digits, I reminded her. She consulted with someone else, and came back to say, "Well, sir, you'll just have to work around that."

Suppose you take a VAL for the number part, and try to subtract it from the string, with

```
100 READ A$
110 B=VAL(A$)
120 PRINT A$-B
300 DATA 123 BROADWAY
```

which brings up

```
TM ERROR IN 120
```

because you're mixing a string with a non-string which is a type mismatch.

Why not turn the non-string number back into a string, with STR$, like this

```
120 C=STR$(B)
130 PRINT C
```

However, this causes

```
TM ERROR IN 120
```

because, in turning B back into a string, we forgot to use C$ instead of C. So we use C$

```
120 C$=STR$(B)
130 PRINT C$
```

and we get a printout of 123, so we add

```
140 D$=A$-C$
150 PRINT D$
```

which brings up

```
TM ERROR IN 140
```

not because of a type mismatch, but because, as the Model III Disk Basic manual puts it, "string values cannot be used in arithmetic expressions." The Model I, Level-II manual says, "there is only one string operation--concatenation, represented by the plus symbol+." To prove this, change the minus to a plus in line 140

```
140 D$=A$+C$
```

and the printout will be, of course

```
        123

123 BROADWAY 123
```

(Why is the first line indented?)

If you try something like

```
140 C$=-C$
```

you'll also get a TM error, because there's no such thing as a negative string (but there is a catchy rhyme).

So now what?

The Power of INSTR

The answer lies in the use of three string functions. We use them to read the length of the address string, find the location of the space in the address string, subtract the position-number of the space from the length of the string, and use the difference to read characters from the right end of the address string:

```
100 READ A$
110 L=LEN(A$)
120 I=INSTR(A$," ")
130 PRINT L,I
140 B$=RIGHT$(A$,L-I)
150 PRINT B$
300 DATA 123 BROADWAY
```

In line 110, LEN counts the character length of string A$, which is 12. The INSTR in line 120 provides the location of the space in string A$, which is the fourth character. Or in the words of the manual, INSTR searches string A$ to see if it contains the substring that consists of a space, and if so, returns the starting position of the substring in the target string.

Line 130 prints out the 12 and 4. Line 140 subtracts the 4 from the 12, and then returns the last 8 characters from the RIGHT end of string A$. When run, this program produces

```
12  4
BROADWAY
```

But INSTR is a Disk Basic function. How do you peel the number off the address in Level-II Basic?

In the Level-II manual's chapter on strings, a paragraph starts, "Using the intrinsic string functions MID$ and LEN$, it's easy to create a very handy string-handling subroutine, INSTRING. This function takes two string arguments and tests to see whether one is contained in the other."

The subroutine can be simplified because we know that no address will have over four digits in it, so the space won't be past the fifth character, thus eliminating the need for LEN$. Also, we know the space is only a single space. So the subroutine in the Level-II manual can be turned into a couple of much simpler lines, without LEN$. (INSTRING was of course the inspiration for Disk Basic's INSTR.)

For Level-II use, replace line 120 in the Disk Basic version with lines 114-126:

```
114 Y$="  "
118 FOR I=1 TO 5
122 IF Y$=MID$($,I,1) 130
126 NEXT
```

Note first that line 122 does not need a THEN before the 130. The Level-II manual says, "It is optional except when it is used to specify a branch to another line number." The Model III Basic manual is more specific, "THEN is optional except when it is required to eliminate an ambiguity, as in IF A<0 100. THEN should be used in IF-THEN-ELSE statements."

In line 122, MID uses the FOR/NEXT I-loop to look for a match between one-space string Y$ and one of the first five characters in address-string A$. When it finds the space at position 4 in string A$, the program branches to line 130.

The Address Routine

Let's finish up the address-finding program. First replace line 150 with

```
145 C=VAL(A$)
147 D=INT(C/20)
```

Line 145 supplies the number part of the address. Line 147 does two things: it "cancels the last figure of the number" and also "divides the remainder by 2." What is the INT for?

Now we have to write lines that will cause a branch from a particular avenue name to a routine that operates on the number preceding that name:

```
150 IF B$="BROADWAY" THEN 172
160 IF B$="AMSTERDAM" THEN 180
170 E=INT(C/10)+60: GOTO 200
```

If the avenue is neither Broadway nor Amsterdam, then it must be Central Park West, and so the program will "fall through" to line 170. For Central Park West, we do not "divide the remainder by 2," but instead "drop last figure, add 60 to remainder."

The rules for Broadway are a little more complicated. According to the instructions, "Up to 754, below East 8th Street. Above 754, deduct following key number: From 754 to 856 deduct 29. From 857 to 999 deduct 25. Above 1000 deduct 31."

So we take care of the first part with

```
172 IF C<754 THEN 210
```

which branches to a PRINT line we'll look at later. As for writing a program line that will take care of a number between 754 and 856, we could write a line starting with

```
IF C>=754 OR C<=856
      THEN E=D-29
```

which may at first glance seem as though it would work. But on closer examination, or if you try to use it in the program, you'll find that if C is greater than (or equal to) 754, the program will never get to the OR, but skip over it to the THEN E=D-29. No matter what the Broadway address over 754 is, it will be treated as though it's between 754 and 856, and the program will go right from this line to a PRINT line, never reaching any of the lines in between.

But by simply using AND instead of OR in that line, and in lines like it, we solve the problem. The AND in line 174 means that any address between 754 AND 856 will be taken care of by that line alone.

```
174 IF C>=754 AND C<=856
    THEN E=D-29: GOTO 220
176 IF C>=857 AND C<=999
    THEN E=D-25: GOTO 220
178 IF C>=1000 THEN E=D-31
179 GOTO 220
```

Amsterdam is an easy one, requiring only adding 59 to the key number for any address on that avenue:

```
180 E=D+59: GOTO 230
```

Now we get into the printing lines:

```
200 PRINT C;"C.P.W. IS AT":E:"ST."
205 GOTO 100
210 PRINT C;"BWAY IS BELOW EAST
      8 ST.
215 GOTO 100
220 PRINT C; "BWAY IS AT";E;"ST."
230 PRINT C;"AMST IS AT";E;"ST."
235 GOTO 100
```

Note that line 220 takes care of all Broadway addresses except those below 754 because we used E in all three lines 174, 176 and 178. Otherwise you'd need a different PRINT line for each of the three situations.

Without the GOTO 100 after each PRINT line, you'd print out all the subsequent PRINT lines.

All that remains now is to add DATA lines to test the address-finding program, including some new ones to test the lines written for Broadway addresses:

```
300 DATA 123 BROADWAY
310 DATA 234 AMSTERDAM
320 DATA 345 CENTRAL PARK WEST
330 DATA 800 BROADWAY
340 DATA 900 BROADWAY
350 DATA 1100 BROADWAY
```

If you wanted to use this program "for real," you'd have to add lines to take care of some or all of the other 27 Manhattan avenues, nearly all of which would require adding only three lines each: one line like 160, another like line 180, and a third like line 230. Of course, many of the PRINT lines can be combined, without having to use a different one for each avenue. How would you do that?

Once you've got lines for all the avenues added to the program, then you can simply replace line 100 with

```
100 INPUT A$
```

eliminate lines 300-350, and you've got a program that will find the cross-street for any address on Manhattan's 30 avenues.

The point of all this, as you probably realized at least a page ago, is not to build an address-finder, but to look at some simple techniques involving strings. If you knew them all, please consider that many readers of this column are just getting started with strings.

Boolean For Short

Here's something you may not know: how to reduce number comparisons to a minimum line-count. For instance, the number-comparing parts of four lines in the address-finding program can be reduced to a single line, by using Boolean algebra.

Boolean algebra involves what the Level-II and Model III manuals call logical operators. Both manuals are pitifully deficient in discussing this important subject, and as a result many TRS-80 users have little or no idea what can be done (and in a minimum of space) with AND, OR, NOT, etc.

Replace lines 172-179 with these:

```
172 DEFFNG(C)=-((C>=754)+(C>
    =857)+(C>=999))
174 ON FNG(C)+1 GOTO 210, 175,
    176, 178
175 E=D-29: GOTO 220
176 E=D-25: GOTO 220
178 E=D-31: GOTO 220
```

The right side of the DEFFN statement is a Boolean expression containing three terms connected by OR logical operators. If, in the first term, the Broadway address is less than 754, the expression is false, and equal to 0. If the address is greater than or equal to 754, the expression is true, and equal to -1.

If the address is below 754, all three terms are 0, and because they are ORed together, the three-term expression is equal to 0. Adding 1 to it allows the ON/GO statement to branch to line 210, for addresses below East 8 St.

If the first term is true, then the expression has a value of -1, which is made +1 by the minus sign at the beginning, and by adding a 1 in line 174, causes a branch to line 175.

If the Broadway address is 900, then the first two terms in the expression are -1 because 900 is greater than 754 and 857, and the total of -1 terms is -2. If the address is 1100, the three terms are all -1, for a total of -3. These become 2 and 3 via the minus sign, then 3 and 4 via line 174, and cause branching to lines 176 or 178.

Thus, through the use of some fairly simple Boolean algebra, we have an expression whose value is either zero or from one to three minus-ones, which are easily

converted for use in an ON/GOTO statement for branching.

The Boolean lines can be made more elegant, as can the entire address-finding program; both were simplified for tutorial use. The DEFFN statement, like INSTR, is found only in Disk Basic. To conserve memory and to make the program run faster, use integer arithmetic by adding the DEFINT statement or the % sign.

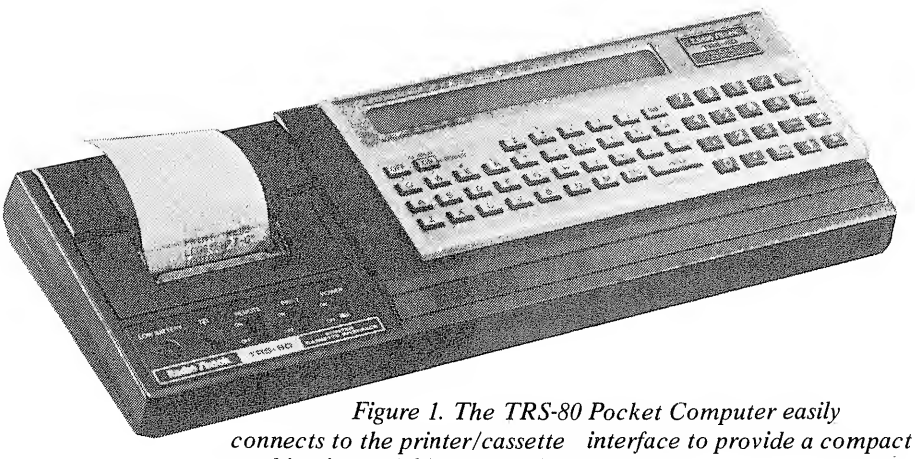


Figure 1. The TRS-80 Pocket Computer easily connects to the printer/cassette interface to provide a compact combination weighing only 1.25 pounds.

Pocket-Computer Printer

If you're going to buy a cassette interface for your TRS-80 Pocket Computer, why not spend $98 more and get a combination printer and cassette interface?

Not only is the tiny printer one of Radio Shack's most fascinating gadgets, but it prints out 5 x 7 dot-matrix characters on regular paper, dark and clear, one line a second, giving you highly legible hard copy of your programs or your results. Because it has a built-in rechargeable Ni-Cad battery, you can carry it along just as easily as the Pocket Computer.

You can also use it as a portable electronic memo pad, but there are only 16 characters to a line on the 1½-inch-wide paper, so you'd have to key in a lot of PRINT lines to record much of a memo.

The printer mechanism looks very much like what Sharp (which makes the PC for Radio Shack) built into their EL-7001 Memowriter, a calculator looking somewhat like the PC but with a "memo" mode for using it like a typewriter for brief messages.

The small 16-page manual is one of Radio Shack's best, covering just about all the points and using two dozen illustrations. You remove the pin-cover from the left end of the PC, and slide the PC in the printer's grooves until the pins engage. The paper, which costs only $1.75 for six rolls (three come with the printer), is easily inserted; a Paper Advance button moves it through the mechanism. The printer ribbon comes in an easily installed cartridge; just push it into place.

A power switch turns the printer/interface on or off; a print switch activates the printer, so you print only what you need. A remote switch controls a cassette recorder when it's connected; when on, the recorder is started and stopped by the Pocket Computer; when off, by the recorder's own keys.

Also supplied are a cable for connecting the recorder to the printer/interface, and an AC adapter for recharging the built-in battery. One charge is said to be good for about 8,000 print lines.

Ordinarily, the first four columns are reserved for line numbers and the separator colon. But when PRINT items are separated by semicolons, all 16 columns are used for printout, as they are if you write memos in quotes in PRINT lines.

```
100:INPUT A,B,C
110:D=(A*A+B*B+C*C)
    *C) (.5)
120:PRINT D
130:END
```

The Pocket Computer is now available at $149.95, the printer/interface is available for $127.95, and the cassette interface is selling for a price of $29.95.

For $14.95, you can get a padded vinyl case that holds the PC and P/I and extra paper, or the "recommended" Minisette-9 cassette recorder, the price of which hasn't changed from its original $79.95.

Games II for PC

For $14.95, you get two cassettes with eight games for the Pocket Computer that "let you test your deductive reasoning, marksmanship, and gambling abilities," according to the 24-page manual.

In *Missile Marksman* you try to hit the target by adjusting the launch angle and the amount of propellant. There are no graphics, of course, to show whether you shot short or long, so you get a readout telling how far, in miles and feet, your shot was from the target.

In *Baccarat* you are the croupier, against two experienced players, in a game similar to *Blackjack*, which is the next game (you play against the dealer), followed by *Aceyducey*; three card games in a row.

One-Armed Bandit simply displays three words instead of the pictures of cherries, plums, etc., displayed by slot-machines; different combinations pay different odds. *Pokerslot* is a similar game, displaying five cards that compose your hand, with five of a kind winning $5,000, etc.

Numguess is similar to the Mastermind game, in which the computer gives you clues, here called direct and indirect hits, as you try to guess the hidden number. *Craps* is the standard game of dice.

These eight games can all be fun if you don't mind a minimum display, of a few words or numbers, on a 24-digit dot-matrix LCD display. The eight have been made about as interesting as possible within the limitations of the Pocket Computer, which aren't all that great once you get right down to playing. In fact, you may get an added kick out of playing games on a computer smaller than two cassette boxes.

Civil Engineering on the PC

Described as an "on-site assistant" in the catalog, this $24.95 package of two cassettes for the Pocket Computer consists of seven programs including Simple-Beam, Cantilever-Beam, and Fixed-Beam calculations (shear, moment, deflection), Column Characteristics (buckling, loading), Cylinders (stress) and Bolts (torque), Section Computations, and Vector Operations.

The 32-page manual provides full details on using these programs, describing all parameters, showing formulas, and giving an example of using each one.

Short Program : Color-Joystick Sound

George Trepal of Augusta, GA, sent "a short program to play on a TRS-80 Color Computer with a joystick:

```
10 X=JOYSTK(0)
20 Y=JOYSTK(1)
30 IF X=0 THEN X=1: IF Y=0
   THEN Y=1
40 SOUND X,Y
50 GOTO 10
```

"The a/d conversion turns out a value from 0 to 63 for the joystick. A more interesting program comes from adding the following lines:

```
15 X=INT(X*3.9)
25 Y=INT(Y/8): IF Y<1 THEN Y=1
```

"This program should be kept away from children if a non-deaf adult has to stay in the same house as the computer."

According to George's program notes, line 10 is for "horizontal left joystick," line 20 for "vertical left joystick," line 30 "gets rid of zero," and line 40 makes a sound of tone X for duration Y.

The horizontal coordinate controls the sound frequency (low at left, high at right) and the vertical controls its duration (short at top, long at bottom). You'll need a very delicate handling of the joystick to play a recognizable tune; you'll go crazy trying to play the first two solo bars of "The Flight of the Bumblebee."

The shorter program gives the sound a range of only a couple of notes, so use additional lines 15 and 25, which provide a range of over four octaves, although with very little control over the top half.

On my Color Computer, the highest, shortest note is at about 1:30 (considering the face of the joystick case as a clock dial), and the lowest, longest note is at about 7:30.

If this program doesn't work with your joystick plugged into LEFT JOYSTK, try plugging it into RIGHT JOYSTK. □

New Computers • Software

June, 1982

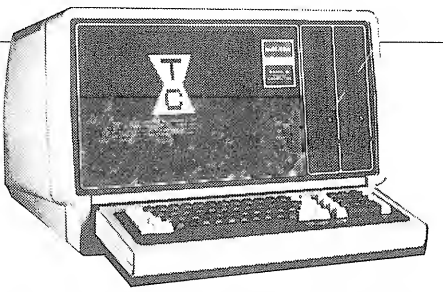


Figure 1. Radio Shack's new TRS-80 Model 16 uses a Motorola 16-bit microprocessor and is compatible with the Model II.

TRS-80 Model 16

Take a TRS-80 Model II, replace the 8-bit Z-80A with a 16-bit microprocessor, replace the single 8" single-sided disk drive with one or two "thin-line" 8" double-sided disk drives, raise the maximum internal RAM storage space from 64K to 512K, add multi-user capability to the operating system, change the black keytops to white and vice-versa, and you've pretty much got a TRS-80 Model 16. In fact, you can upgrade a Model II to a Model 16.

The Model II uses one Z-80A for the keyboard, the other for the video. The Model 16 uses a Z-80A for input/output, but for a main processor uses a Motorola MC68000, which allows the 16 to access more memory than the Z-80A does. Also, because the data path of the 16 is 16 bits wide, and because the MC68000 can perform 32-bit operations internally, the new computer can process more complex data and at higher speeds. The MC68000, by the way, is a generation newer than the Motorola 6809E used in the Color Computer.

The single built-in disk drive in the Model II stores 416,000 characters. The TRS-80 Model 16 comes in two basic versions, both with 128K of RAM: single drive, 1¼ megabytes of storage, at $4999; two drives for a 2 1/2-megabyte capacity, $5798. That's three (or six) times the internal-disk capacity of the Model II, and four times as much RAM as the basic one-drive 64K Model II that sells for $3499.

The multi-user feature permits adding one or two local or remote terminals (such as the new DT-1 described later) so that up to three people can use the computer at the same time.

Upgrading a Model II to a 16

The Model II is upward-compatible with the Model 16, meaning Model II programs will run on the 16, but not always vice-versa. Like the Model II, the 16, through one of its serial ports, is capable of bisynchronous communications with IBM and other mainframes. Adding a hard-disk port permits using up to four of Radio Shack's 8.4-megabyte Winchester-type hard disk drives.

A Model 16 Enhancement Option consists of two plug-in boards that provide the Model II with the 16-bit, dual-processor, multi-user features of the Model 16, for $1499 (plus installation). The Model II's disk drive and display remain unchanged in this upgrade.

The TRS-80 Model 16 is aimed at sophisticated corporate users. It is compatible with Radio Shack's ARCNET local network (based on Datapoint Corp.'s ARC network), which supports up to 255 Model II and 16 computers in any combination, and which is due in mid-1982.

TRS-80 DT-1 Video Terminal

Replace the dual disk drives of a TRS-80 Model III with a single cover, change the display from 16 lines of 64 characters to 24 lines of 80 characters, change the microprocessor, and you've pretty much got Radio Shack's DT-1 Video Data Terminal. The price is $699, which coincidentally is the price of a 4K Model III without disk drives.

The DT-1 can emulate the protocols of four standard terminals—Televideo 910, Lear-Siegler ADM-5, ADDS 25, Hazeltine

1410—with configurations selected from the keyboard, and will maintain a protocol despite power failures through the use of an EEPROM (Electrically Erasable Programmable Read-Only Memory).

The display provides normal, reverse, invisible, blinking, underlined and half-intensity video, along with four types of keyboard-selected cursors: steady or blinking, block or underline.

The DT-1 has both parallel and RS-232C serial interfaces enabled as printer ports by a local-monitor mode that also can position the video cursor using local-control and escape modes. Ten keyboard-selectable baud rates, from 75 to 19,200, are available. An electronic bell is standard.

PC-2 Pocket Computer

At first glance, the new PC-2 Pocket Computer looks very much like the older model that Radio Shack may eventually call the PC-1. But look closer and you'll see several differences in the PC-2, which is list-priced at $279.95.

The new display is 26 characters long instead of 24, and has both uppercase and lowercase characters. The typewriter-style keyboard still uses the QWERTY layout, but the 13 shift-mode characters (dollar-sign, up-arrow, etc.) have all been relocated and three more have been added: square-root, ampersand and @. Also added are six function keys; in addition, the PC-2 has 18 "soft keys" and 10 pre-programmed command keys.

However, the *big* differences are that the PC-2's memory can be expanded internally, it can be expanded externally through a 60-pin I/O connector, it uses Extended Basic, and it will be usable as a terminal.

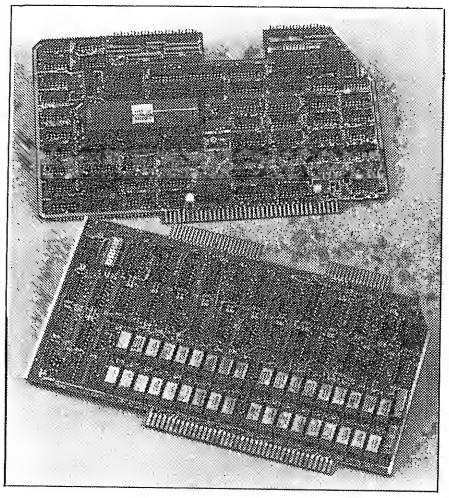


Figure 2. These two plug-in boards upgrade a TRS-80 Model II to a Model 16, giving the user 16-bit, dual-processor, multi-user operation.

A $69.95 4K RAM module plugs into a recessed port on the back of the PC-2; up to 16K of RAM and/or ROM may be plugged in. (Plug-in ROM-based programs may thus be in the offing.) The standard PC-2 memory consists of 16K bytes of system ROM and 2640 bytes of user memory (1850 bytes of Basic program and data RAM, 600 bytes of fixed data memory, and 190 bytes of reserve memory).

The PC-2's Extended Basic has 42 statements, 34 functions and 6 commands, and includes two-dimensional arrays, variable-length strings up to 80 characters long, and string-handling commands such as LEFT$, LEN, VAL and CHR$. A built-in real-time quartz clock is accessible from the keyboard or from Basic.

A fully-addressable LCD dot-matrix display of 7 by 156 dots permits generating special user-definable characters.

Late in 1982, Radio Shack plans to introduce an RS-232C interface and software that will let the PC-2 be used as a terminal with many RS-232C-equipped computers and peripherals, as well as providing connection to Radio Shack's modems for data communications over ordinary telephone lines.

The PC-2 Pocket Computer, according to Radio Shack, has "advanced capabilities" that "open up a great many application areas, including engineering, scientific and medical calculations; banking, credit, insurance, investment and financial applications...its capabilities are closer to a desktop computer than any previous pocket computer, yet it costs much less than typical desk top personal computers."

The PC-2's Extended Basic is upward-compatible with TRS-80 Pocket Basic, so programs prepared for the "PC-1" will run on the PC-2.

PC-2 Four-Color Printer/Plotter

A four-color printer that can plot graphics, plus a dual-cassette interface, comprise the PC-2 Pocket Computer Printer/Cassette Interface, available in mid-1982 at a list price of $239.95.

The printer/plotter adds 25 commands and statements to Pocket Extended Basic, including special statements for plotting 216-by-512-pixel graphics in red, blue, green and black with replaceable ballpoint pens, on 2¼" cash-register-type paper. Characters can be printed in nine different sizes, from 4 to 36 per line.

The "PC-1" is manufactured for Radio Shack by Sharp Corp., which markets it as the PC-1211 Pocket Computer. The PC-2 is also made by Sharp.

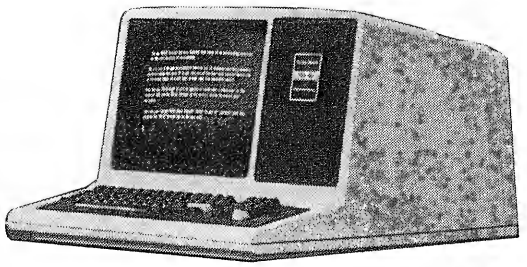


Figure 3. Radio Shack's DT-1 video data terminal, based on the TRS-80 Model III, is designed for multi-user applications of the Model 16 and as a time-sharing remote terminal.

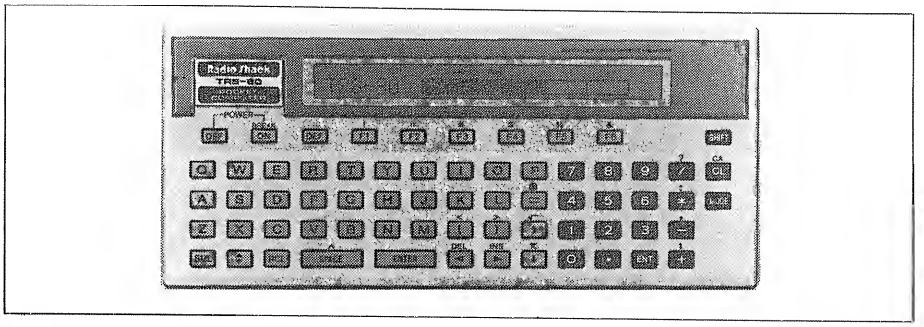


Figure 4. Radio Shack's PC-2 Pocket Computer is a close cousin to its first Pocket Computer; but the memory can be expanded internally.

Aluminum Sprayed on Plastic

If you've used a TRS-80 awhile, and are in the habit of resting your hands on the plastic surrounding the keyboard, you've probably found that the aluminum color isn't molded into the plastic—it's sprayed on.

So what can you do to keep it from wearing off? If you cover the two places where your palms rest on the surround, then when you try to remove the tape later, you'll take off more aluminum paint than if you hadn't used the tape. Any solutions? And has anybody found a spray-on or brush-on paint that matches the TRS-80's aluminum color?

Blank Program Lines Revisited

From Washougal, WA, Delmar D. Hinrichs writes:

"You did it the hard way! In your column in the Jan. 1982 issue of *Creative* (p. 180), you suggest adding blank lines to programs by adding spaces until the cursor goes to the next line. This works, but there is a better way.

"Instead of adding spaces to the program line, use the ↓ (down-arrow) key. This puts a line-feed character, ASCII 10, into the program line. Besides requiring fewer key strokes, this method of adding blank lines has two real advantages:

1. Fewer bytes are added to the program (I always seem to be short of memory).

2. The program will list on a printer the same as it is displayed on the video display.

"This method of using ↓ may also be used to break long lines to make them easier to read. The ↓ may even be used within a long ON/GOTO, ON/GOSUB, or DATA statement to make it look better. For such long lines, it helps to add a space or two after the ↓ to indent the later part(s) of the line."

Delmar is right, but only for printers such as the Radio Shack Line Printer VIII. If you've got a printer like Radio Shack's Line Printer II, which is really the Centronics 730, you can add a dozen line-spaces with the down-arrow, and that printer will ignore them completely, and print, one after the other, the two lines you had hoped would be separated by two inches of space. With the II/730 and others like it, you must add spaces the hard way: one at a time.

Disk Scripsit

Radio Shack's Disk Scripsit is almost exactly like cassette Scripsit (June 1980, p. 166), with only a few small differences. One is price: the Model I/III disk version of this word-processing program is $99.95; on cassette, for 16K Level II or Model III Basic, it's $39.95.

The manuals for the two Scripsits are almost identical, except where they involve loading the program. Also, the disk manual includes a page on "Helpful DOS Instructions," half a page on "Using Scripsit in Conjunction with Basic" (save all Basic files in ASCII format), additional error messages, and 2½ pages on "How to Modify Scripsit for Use With Special Printers" (with assembly-language routines), plus two addendum sheets that say, among other things, that the Scripsit disk "can be backed up or copied only twice" (really??).

You get the same three audio tapes with both versions, with six lessons that guide you through what, as the cover indicates, is actually more of a training guide than a manual. This makes it difficult for you to look up anything later; there's no index. So you usually have to depend on the Instruction Summary Card, which is very handy if you use Scripsit often enough for the summarized material to be meaningful.

All these differences aside; if you're writing anything more than a few pages long, Disk Scripsit is the way to go. If you've got a 16K TRS-80 and cassette Scripsit, you can keyboard only about 3½ double-spaced pages of text before the memory is full, and you have to save it on tape before you can add any more.

With Disk Scripsit, you can enter over 100 double-spaced pages on one 5¼" disk, and move paragraphs around much more easily than on tape, where it is cumbersome to move material from one recorded section to another.

Some of the features that make Scripsit such a pleasure to use include global search and replace, text centering, page numbering, hyphenation help, headers and footers, and justified margins.

In Praise of Scripsit

The following letter is from George Brubaker, a research associate in the Department of Chemistry at Wayne State University, who has a TRS-80 with a low serial number:

"I have followed your column since its inception (this is SN 7727 running Scripsit, the real point of my letter), and find it to be an excellent focus for those of us who are TRS-80 devotees as well as newcomers who are comparison-shopping.

"I've often been asked why I am such a devoted TRS-80 fan. Frankly, for two reasons, which might be worth restating in your column.

"1. This is the most reliable computer I have ever used. Obviously, it is old. It has never been serviced because of a failure. It has been 'lowercased' and it has had the keyboard replaced (after three years of full-time service), both by Radio Shack, and both times with a turnaround of 30 hours or less. In comparison, we are still trying to find reliable and fast service for our Apples (which need a great deal of service) and our PETs (which don't, but when they do....). *Service*, then, is a key point in Tandy's favor (at least in metropolitan Chicago, our full-time home).

"2. Scripsit is a nice word-processor, which has been reviewed several times. But in each review, a key point was missed, a point which also bears heavily on the overall utility of the TRS-80 computer. That is, Scripsit, Basic and Fortran can read and write common *full ASCII files* (in contrast with the more common compressed ASCII used internally by most Basic interpreters). In essence, that means you can write and edit programs as well as letters using Scripsit provided that the Basic or Fortran program is saved from Scripsit using the ?S,A option.

"Programs written in Disk Basic and saved with

SAVE "name",A

may be loaded into Scripsit for editing (or formatted printing). If you've ever wanted to change one line number in Basic, or change every occurrence of a string in a Basic program, you already have justification for getting Scripsit.

"If you've used both Scripsit and EDIT80 in the Radio Shack Fortran package, you can appreciate the power of that 'A' option. Scripsit tabs can be set for Fortran programs (or Pascal, if you like 'prettyprint listings'), and a program may be written and edited just like a letter. You needn't worry about the line numbers EDIT80 puts in your program. Forget 'em, because the Fortran compiler doesn't want them; they are used only to guide the programmer and EDIT80."

Survivor

I've got a complaint against the *Survivor* game in the Jan. 1982 issue of CLOAD. There's not much problem getting away from a piranha, pterodactyl, giant squid, tyrannosaurus or even a zombie as you move toward the treasure. Even a ghoul can be evaded. But as the treasures increase in value, the monsters get closer...(gulp)... and closer. The wraith and the demon can sometimes be outwitted, but the witches—oh my, the witches!—they stay close, they breathe down your neck, ready to leap on you the second you stray from the safety barrier around the perimeter, just as you're heading for the $3500 gold nugget or the $2400 sapphire broach.

The game's author is a fiend, an absolute fiend. He lets the early easy wins lull you into a false sense of confidence, so you're easy prey to the Green Witch, if you get that far, which is doubtful. Author John Howard (may he spend the rest of his life in Cleveland, writing tic-tac-toe games) has made sure that long before you escape the ultimate horror, the Green Witch, and make off with a total of $101,000 in treasures, you will die of heart failure or go berserk. What's more, this fiendish Aussie (of Toorak, Victoria), had the unmitigated gall to write *Survivor*, not in machine language, but in Basic.

Voyage of the Valkyrie

One of Leo Christopherson's best thus far is *Voyage of the Valkyrie*, the first game from the Advanced Operating Sys-

tems division of Howard W. Sams & Co., and "available at computer stores nationwide" with a suggested list price of $34.95 for cassette, $39.95 for disk.

"As you move through the island of Flugloy," according to the press release, "you encounter bird-like creatures which fly around your CRT in an effort to protect their castle's treasure."

Your first job is a game in itself: map the island. Half a dozen double-sided gridsheets are supplied. You teletransport yourself from one location to another, often right into a solid object; you'll be destroyed many times in your mapping.

Some map locations are defended; you fire at the birds with a complex weapon. There are ten castles to be captured, each containing gold; your score is the amount of gold you have at the end.

A full description of Leo's game could take pages. There's Wagnerian music played (via an external amplifier and speaker) "at various appropriate times during the campaign," ten levels of difficulty (with about 60 birds to defeat at level 0, up to about 600 at level 9), different birds with differing weapons and fighting abilities, energy to replenish, a complex Status Report, and ranks that you climb as you capture more gold, from private to prince regent.

The 12-page manual is as colorful as it is complete, with a diagram of the keyboard that shows very clearly what each relevant key is for.

As usual with Leo's games, the graphics are superb, as are all the other facets. This is another of the handful of games that create high standards for others to aim at.

Mind Thrust

Described on the back of the 12-page manual as "The first of its kind—an exciting game that lets you match wits with the computer," this $16.95 tape for Level-II 16K TRS-80 from Hayden Book Co. doesn't quite match that claim.

For one thing, there *are* other games, more exciting, that let you match wits with the computer. The most recent one reviewed here is *Iago,* (Feb. 1982), Datasoft's version of *Othello*, and highly addictive.

For another, *Mind Thrust* involves just a little too much luck. It's an 8-by-6-square board game in which you try to "complete an unbroken chain across the playing board" to win. "The player takes turns at either adding a piece to his chain or attacking the opponent's chain to make it shorter."

When you are attacked, you announce which of your pieces you choose to defend.

"If the point attacked is *not defended*, the piece in that square is replaced by an attacker's piece. All of the defender's pieces touching the attack point are removed from the board."

The catch is that skill plays very little part in deciding whether you are successful in defending your pieces; it is a random choice by the computer. If you like a mixture of "skill and luck" then perhaps you'll like this game, which has a "legalized cheating" feature: at any time you can switch sides with the computer. Any time you're about to lose, just switch and take over the winning side.

Short Program : Watch Close Now

Jeff Holman, a high-school senior in Monroe, NC, sent a short program that does something mysterious—if you don't know how it works.

"Our Sun Valley High School math club, of which I am a member, recently purchased a Level-II 4K TRS-80 for the members to use in their spare time. (We do not have data-processing classes.) It was not long before we started subscribing to *Creative Computing*.

"I have since written many programs (lately I have concentrated on those using graphics). Some that I am most proud of are a program that displays output alphanumeric characters as one-inch-high graphics characters as soon as the individual keys are hit (or twice the width if the SHIFT and right-arrow keys are hit) and an *Etch-A-Sketch* program that allows the user to draw pictures on the screen. When the picture is finished, the entire screen is 'saved' in a few strings that can be printed at any time. (It would be a lot easier if I knew how to use machine language!)

"I went off on a tangent with string-packing. Instead of poking graphics characters, a program I wrote pokes alphanumeric characters. This particular program is helpful when 'pictures' are stored in memory to be printed on a line printer. The characters can be typed on the screen to form the shapes desired as they are simultaneously 'packed' into successive strings.

"I have a zillion-and-one short programs I'd like to send in, but I'll just concentrate on one for his letter. I won't say what the program does (I love surprises!). The real 'meat' of the program is between lines 60 and 90; the rest is just to prepare the screen.

"I know it's a bit too slow for most programs, but I'll leave it up to your readers to 'juice it up.'"

```
10 CLS
20 FOR X=15360 TO 16384
30 POKE X,RND(64)+127
40 NEXT X
50 'OKAY—WATCH CLOSE NOW
60 FOR X=15360 TO 16384
70 XX=PEEK(X)
80 POKE X,191-(XX-128)
90 NEXT X
100 GOTO 100
```

If you have trouble figuring out how this works, change 16384 to 16383 in both places, and then ask yourself why the change makes a difference. If that doesn't help, look at the Level-II TRS-80 memory map in the Radio Shack manual. If you can write a very short program that produces the same effect as these ten lines, then you understand what's going on.

Line 80 reverses each displayed graphics character to its "negative" but only once. If you change 16384 to 16383 and change line 100 to

```
100 GOTO 60
```

the screen will alternate continuously between a display of graphics characters and their negatives. Changing line 100 to

```
100 GOTO 20
```

provides a different pair of positive/negative displays at each go-around.

NOTE: This short program works as described with Model-I Level-II Basic and with Model III Basic. Although with Model III Disk Basic, the "watch close now" effect isn't as dramatic, you may still have a problem figuring out what happened. □

TRS-80 Repairs • Reviews

July, 1982

Disk-Drive Repair

Late last year, drive 1 on my TRS-80 Model III began to act up, giving GAT, CRC and HIT error messages. Eventually it just wouldn't work at all. I tried to use it now and then, hoping that, like a toothache, the problem would go away.

When the problem persisted, I put the III back into its box and took it to the service department of the Radio Shack Computer Center at Fifth Ave. and 36th St. in New York. With the very kind permission of a Radio Shack vice president and of the center's service manager, Patrick Raeihle, I was allowed to watch computer technician Michael Simmons tackle the drive.

Mike told me that ordinarily he can align a pair of disk drives in about 20 minutes, once a Model III is unbuttoned. However, with all my questions and his very patient answers, it took several times that long.

Mike removed one screw from the back and ten from the bottom of the Model III. A warranty sticker covers one of those ten. Once you remove this sticker to do your own thing inside a Model III, you'll find you can't put it back on, and you've probably voided the remaining warranty.

What happens if you have installed a "foreign" modification in your Model III (or any other Radio Shack) computer? According to Pat, you have to take out the modification before Radio Shack will repair it. If you don't take it out before bringing your machine to be repaired, they'll do it for you, if they find they can't work around the modification, but will charge you for the removal, and won't re-install the modification. This is the official Radio Shack service policy, as stated in the back pages of the Model III manual, among other places.

After lifting off the top of the Model III, Mike tried a backup, "to verify the complaint." Although drive 1 wrote the backup, and a DIR could be read, the drive slowed down during the formatting, so Mike felt it might need alignment. He loaded the alignment program, which has seven parts, displayed on a menu and numbered 0 through 6.

(0) SPEED CAL/DEVIATION

Although there is a strobe disk on the drive (much like the one on some hi-fi turntables), a more accurate reading of the drive speed is obtained from a scale on the screen. The speed of my drive 1 was a little off, so Mike adjusted a potentiometer on the PC board, and brought the speed to 300 rpm.

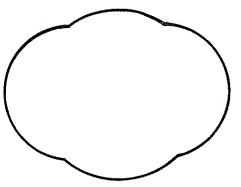
(1) CARRIAGE STEP

This test makes sure the head travels smoothly along its carriage between tracks 0 and 40, without binding.

(2) HEAD RADIAL/AZIMUTH

This test, which checks for centering, requires a special disk with a signal recorded on it for oscilloscope alignment (Mike used a 100-MHz HP 1740A). There's somewhat of an analogy with a hi-fi record; if the hole isn't in the exact center of the record, the sound suffers.

Track 16 is supposed to be in the center of the disk, and the test checks to make sure it is. Instead of using a circular pattern on the disk, alignment is simplified by checking a two-lobe pattern something like this:



Mike looked at the scope while running this test, and made mechnical adjustments on my drive until the amplitudes of the two lobes were equal. Then he stepped the head to track 0 and back, checked to see that the disk was still centered, and stepped the head out to track 39 and back, and did the same check again.

(3) TRACK 00 ALIGNMENT

A mechnical switch makes sure the head starts reading at track 0, which is where the boot track is; if this switch is out of alignment the Model III can't boot up, can't get started.

(4) INDEX/SECTOR TIMING

This test checks the timing of the small hole that is an inch from the center of the floppy disk. Sector timing is provided by a signal picked up by a photocell that detects the light from an LED on the other side of this index hole. My drive had to have its light path realigned, to line up the LED, index hole, and photocell.

(5) HEAD AMPLITUDE

This check on the read/write head detects if it's worn or dirty. The scope amplitude of the head on my drive was only about 160 to 170 millivolts, and should have been greater than 200mV. According to the missing index/sector timing being off, and a worn head.

Rather than replace the head, which is a $90 job because of the time required to make many precise adjustments, Mike followed standard Radio Shack procedure and replaced instead the mechanical portion of the drive, after removing the drive electronics, which are mounted on a PC board. "It's cheaper to change the whole drive mechanism," said Mike, "because the exchange price of the mechanism is $74. Head alignment is time-consuming and tricky."

Although the new drive mechanism had been tested when it was built, Mike ran quick checks on it after installing it in the Model III. What causes head wear? "Normal wear, or dirt grinding away, or excessive use of a head cleaner," said Mike.

(6) RAW DATA CHECK

Mike didn't perform this test, which he does only when there is a source-disk read error or destination-disk write error. He ran the first six tests on my other drive, drive 0, found the speed was OK but adjusted it a little anyway. The lobes were off in test 2, so he adjusted the

centering, and also aligned the light path for the index hole.

Diagnostic and Memory Tests

As a final test, Mike ran a diagnostic that writes into each sector on each track, and then verifies what is written, and follows with a memory test of ROM, RAM and video (normal, graphics, and alternate characters). The drives passed these tests, so Mike buttoned up the Model III and put a new warranty sticker over that one screw. The Repair Warranty covers 90 days for parts, 30 days for labor.

Labor time is $15 per half hour. A drive alignment is $25 per drive, a straight charge that includes labor time.

Common Disk-Drive Problems

How often do disk drives have to be aligned? That all depends, of course, on how much you use them, said Mike. The three most common disk-drive problems are those involved with tests 0, 2 and 4: mechanical adjustments of speed, centering, and index light path. The next most common problem is detected by test 5: head amplitude.

If, as some TRS-80 users have done after picking up their repaired computers, you take yours home without the protection of the original packing case, and either lay it on the floor of your car, or on the floor of your commuting train, one or more of those mechanical adjustments may go out of kilter. TRS-80 disk drives aren't like $60 cassette recorders, which can take a lot of bouncing around and still work.

Mike, incidentally, prefers the Model III, "because it has two internal drives and 48K of internal RAM without expensive interfaces, and it has room for boards such as the RS-232."

My thanks to Pat and Mike, who answered all my questions with great patience, in a 90-minute session after the 1:00 closing time on a Saturday last winter.

Beginner's Russian

Two $9.95 cassettes for learning to read Russian are available from your local software dealer, or from Instant Software (Peterborough, NH 03458). They require a 16K Level-II machine, and will run on the Model III.

Beginner's Russian teaches the shapes of 33 Cyrillic letters. You start with seven simple consonants, which are displayed 5/8-inch high, along with brief advice on how to pronounce each. Then you get ten vowels, followed by the third item on the menu, twenty "Words to Translate." These are presented in Cyrillic, and if you didn't memorize the letters, you can repeat menu items 1 or 2, or ask for HELP, which will display an abbreviated letter chart. If you don't quess what the Russian word is, you're asked to try again, and again, which is one way of making you learn the letters.

Consonants of "medium difficulty" are displayed later, as are the names (in Cyrillic) of two famous poets along with short biographies. The last part includes "difficult" consonants (but with no pronunciation help for seven of the eight), and the names and short biographies of Lenin, Stalin, etc. The last display says that if you want to continue the course, "purchase *Everyday Russian,* Stock #0137R."

The tape is interesting, but the author missed a major opportunity to help the user. He could have provided voice recordings of the proper pronunciations, right on the same cassette tape. Perhaps this can even be done as part of the program, with a voice pronouncing each letter after it is displayed, and a machine-language subroutine turning the cassette recorder on and off at the right times.

This would eliminate the need for pronunciation "helps" such as for the letter that looks like "bl" and which is "like the I in 'sin' but pronounced with the lips almost closed."

Everyday Russian

The second Russian tape provides names of foods (drinks, entrees, desserts, etc.), places to eat (restaurant, cafe, etc.), signs, and types of stores. Then it shows the Cyrillic alphabet, and turns the TRS-80 keyboard into a "Russian typewriter"; you press the keys, and Cyrillic characters appear on the screen, so you can write Russian words.

This, and the drill provided for each group of words are the unique features of these two cassettes; the rest can be studied easier, more cheaply and more extensively in a textbook. If the author had provided voice pronunciations on the tape, then these two cassettes could be highly recommended to computerniks. As it is, you're better off buying a record or cassette designed to teach spoken Russian. Nyet pravda?

Music Master

A third $9.95 cassette from Instant Software is *Music Master*, which contains a quartet of programs that will run on the 16K Model-I Level-II or Model-III TRS-80.

Micro Organ provides four octaves and three voices. The second display shows which keys go with which notes and which keys control which octaves and voices. (See Figures 1 and 2.)

The keys sound, as you play them, through your cassette recorder speaker or an external amplifier/speaker combination. The three voices don't sound at all like the instruments whose names they bear, but more like the trumpet, trombone and flute voices, respectively, of an inexpensive electronic organ. Which in this case isn't bad at all.

Kaleidopy is a combination kaleidoscope and player piano. The computer displays a four-way kaleidoscope graphics pattern and then "plays" that pattern, which sounds somewhat like the random "beep-boop-bop" of space movie robots.

You can change the pattern (and the sound too) either by the menu choices you make, or by rewriting certain lines in the Basic program. The brief booklet also shows how to change the machine language subroutine that creates the sound.

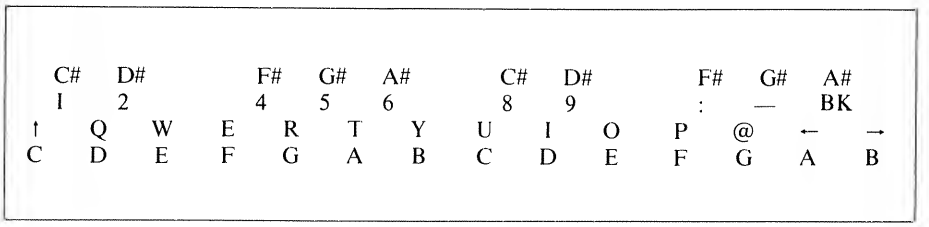
Composer permits generating somewhat random music "in any of fifteen different scales and three tempos," according to the leaflet, which also shows how to alter the machine language sound routine. Interesting, but don't play it very long around computerphobes or Mozart lovers.

Keymania is a game for up to four people very much like the hand-held game that challenges you to remember and repeat a short random tune played by the computer.

Three parameters can be controlled: level of difficulty (selects from 4, 8, or 13 notes); duration of sound (from short to long, with five choices): and interval between notes (from short to long, 1 to 5).

The computer asks how many people are playing, how many notes (from 10 to 50) are to be played each round, and

Figure 1.



whether you want the notes printed out as you press the keys while trying to duplicate the tune played previously.

Although *Kaleidopy* and *Composer* are mainly of interest to computer music buffs, and *Micro Organ* can be enjoyed by almost anyone, *Keymania* is not only a fascinating game, but can be used for ear-training in teaching music. The computer repeats the group of notes until you press the right keys.

Cheaper TRS-80 Computers

You've probably noticed those ads, in *Creative Computing* and other personal computer publications, offering TRS-80 computers at considerably less than the prices listed in the Radio Shack catalogs.

The 32K Model III with RS-232C interface and two disk drives, which lists at $2495, is advertised variously at $2059, $2100, and $2049.

A 16K Color Computer with Extended Basic, which lists for $599, is available at $459, $495, and $455.

No, they're not Radio Shack stores, these places in small towns in Massachusetts, Michigan, Georgia and elsewhere that advertise these cut-rate prices. That is, they are not owned by the Fort Worth-based company. They are franchised dealers in towns that Radio Shack considers too small to support a regular Radio Shack store. There seem to be about two "dealer and franchise locations in the USA" for every three "USA company stores."

Some Radio Shack catalogs carry a "Radio Shack USA Store/Dealer Directory," with an asterisk beside about 20 to 90 percent of the town names, depending on whether the state is California or Arkansas, for example. The asterisk refers to a line at the bottom of the page: "A Radio Shack Authorized Dealer is located in This Town -- May Not Stock Every Item in This Catalog."

Just below is another line: "Retail Prices in this Catalog May Vary at Individual Stores and Dealers." I presume, perhaps wrongly, that all Radio Shack-owned stores charge catalog prices, whereas the dealers set their own prices.

A Radio Shack dealer, if he sells mail order only, may decide to sell only Radio Shack products. If he has a store, he usually sells Radio Shack products along with non-Radio Shack items, such as hi-fi equipment, or even nuts-and-bolts hardware. Look again at the ads, and you'll see that the cut-rate advertisers are all in small towns, such as Littleton, MA; Milford, NH; Perry, MI; or Cairo, GA.

A Radio Shack company-owned store isn't supposed to sell you a TRS-80 at less than catalog price, or the manager is in trouble, as I understand it. But a franchised Radio Shack dealer, sometimes called a "Radio Shack authorized dealer," can sell a TRS-80 for any price he chooses, even for less than it cost him.

A franchised Radio Shack dealer, located in a small town in Georgia or New England or wherever, and selling mail order, has lower expenses, and can probably make more profit selling a TRS-80 PC-2 at $240 or whatever, than most large Radio Shack stores can at $280.

"Radio Shack prices are reasonable," says an observer of the personal computing scene. "There's another reason why mail order prices are low; they tell you to go to the manufacturer if you have a problem with your TRS-80 during the warranty period."

One way a franchised dealer can make even more profit is, for example, to install cheaper, non-Radio Shack disk drives in the Model III, or to sell you the drive kits and let you do the installation. The dealer will have to service such a machine, as Radio Shack won't. If you have a problem with the keyboard of such a Model III, and take it to a Radio Shack Computer Center, they can refuse to service the keyboard until the "foreign" drives are removed.

If you doubt this, read the service policy at the back of the Model III manual: "If any Radio Shack computer equipment has been modified so that it is not within manufacturer's specifications, including, but not limited to, the installation of any non-Radio Shack parts, components, or replacement boards, then Radio Shack reserves the right to refuse to service the equipment, void any remaining warranty, remove and replace any non-Radio Shack part found in the equipment, and perform whatever modifications are necessary to return the equipment to original factory manufacturer's specifications."

A call to Mark Daniels, president of Computer Discount of America, in West Milford, NJ, brought out this information: If you buy a TRS-80 from CDA and it needs fixing, you "take it to any Radio Shack store." This includes a Model III with disk drives, because CDA sells the III with Radio Shack drives. If you buy a TRS-80 with non-Radio Shack drives, says Daniels, you must return it to the place from which you bought it if you have problems, because Radio Shack won't service it.

This manifesto was expressed in the Sept. 1981 *TRS-80 Microcomputer News*, in an article by Jon Shirley, vice president of computer merchandising at Radio Shack, who said, in part, "Let's face it, mail order houses do not have local service if they have any at all. They offer a warranty, but to take advantage of it you must ship what you bought back to them. With a Model III disk drive kit, what do you send, the kits you just installed or the entire Model III? It's this lack of convenient service and retail store space that gives them the low overhead and low prices. Or in other words, you get about what you paid for. So caveat emptor," which means 'let the buyer beware.'

What are the advantages of buying from a franchised dealer? Lower prices, says Daniels, plus "you avoid the state tax, if you buy from out-of-state."

One item that adds to the overhead of a company owned store is the commission that salemen get, says Daniels. A mail order dealer pays no commission, thus increasing his profit.

Note that Radio Shack makes the same profit whether, for example, a PC-2 is sold by a dealer for $240 or by a company owned store for $280. Also, it's easier for Radio Shack to ship twenty TRS-80 computers to a dealer than two or three to a company-owned Radio Shack store; this seems to be the usual ratio, according to Daniels.

Figure 2.

Octaves:	Voices:
A - Lowest (1 Voice)	Z-Organ
B - Middle C (3 Voices)	X-Harpsichord
C - Highest (3 Voices)	C-Piano

Sharp Pocket Computer Manuals

The 124-page 26-3501 manual that comes with the Radio Shack PC-1 Pocket Computer (now $159.95, in the RSC-7 catalog) is written too compactly for most beginners, and is best suited for those with previous programming experience.

The six "Sample User's Programs" make a curiously eclectic mix: biorhythm, guess-a-number game, series-circuit impedance, days between dates, random numbers, and normal distribution and percentile. So Radio Shack brought out a $1.95 book of "50 ready-to-run programs in Pocket Basic. Business, education, and home applications." Not having seen this book, I can't comment on the programs or the mix.

However, I have seen the three books that come with the PC-1211 Pocket Computer from Sharp Electronics Corp. The PC-1211 is the same as the PC-1; Sharp makes both; the only difference is the labels.

The 46-page "Beginner's Textbook for Basic" provides the ABC's of Basic, and has a total of two dozen exercises at the end of three chapters, with solutions. This book is quite simple, with only one flowchart and nothing more complex than a one-dimensional array.

The 101-page "Instruction Manual" has many pages that are very much like those in the Radio Shack manual, and in general contains just about the same information, but without the six sample programs.

These six, and 128 more, are contained in the 307-page "Applications Manual," which divides the programs into ten groups: mathematics (simultaneous equations, eigenvalue, etc.), statistics (Poisson distribution, exponential regression), surveying (stadia calculation, closed and fixed traverse), civil engineering (stress, ridge deflection), electronics (series-circuit impedance, self-inductance), more civil engineering (velocity of flow, bending stress), mathematics (points of intersection of two circles, circle tangent to two lines), financial(present value of annuity, depreciation), chemical engineering (viscosity index, heat conduction), and games (cannon shot, treasure hunt). These are all presented in the same compact format as the six sample programs.

The manual for the TRS-80 Pocket Computer has been edited by someone with a more fluent and colloquial knowledge of English than whoever translated the Sharp PC-1211 Instruction Manual from the Japanese. Perhaps someone in Fort Worth reworked the Sharp manual into a lighter, more conversational style. The PC-1211 manual has some odd-sounding or stilted phrases, such as "in case of a number beyond above range" and "be sure to have occasional cleaning of the recording head." However, the translation is excellent most of the time.

Figure 3.

```
5000 DIM P(1024)
5010 Z=1
5020 FOR A=15360 TO 16383
5030 IF PEEK(A)=32 OR PEEK(A)=128
     THEN 5040 ELSE P(Z)=PEEK(A):
     GOTO 5060
5040 IF PEEK(A+1)=32 OR
     PEEK(A+1)=128 THEN 5070
5050 P(Z)=A+1
5060 Z=Z+1
5070 NEXT A

5100 CLS: INPUT "PRESS ENTER TO
     DUPLICATE SCREEN"; A$

5200 P=15360
5210 FOR A=1 TO Z-1
5220 IF P(A)<192 THEN POKE P,P(A):
     P=P+1: GOTO 5240
5230 P=P(A)
5240 NEXT A
```

All four manuals are printed in Japan, on paper and with typefaces and layouts that look a little odd to Western eyes. Which is not to detract from their being quite good—for the experienced computer user.

The three Sharp manuals are available at some Sharp dealers, at prices set by the individual dealer. The Sharp PC-1211 was available in the spring, from some New York dealers, for $139.95. For $254.95, they were also offering the Sharp PC-1500, identical with the PC-2 that Radio Shack offers for $279.95 in the RSC-7 catalog. As for what manuals accompany the PC-1500 and PC-2, I haven't seen them yet.

Short Program : Screen Saver

From Lowell, IN, Jim Peterson sent this:

"I am writing in reference to the *Storing Graphics* subroutine by W.A. Fronek in the Jan. 1982 *Creative* (p. 178). While this program will work, I have a faster routine, which uses less memory (dimensioning one array instead of two), and will save any alphanumeric characters as well as graphics.

"Purpose: This subroutine will save anything displayed on the screen. The codes for characters and graphics are transferred from video memory to the P array. Duplication is very fast when much of the screen is blank. The more the screen is filled, the slower the duplication. Saving of the screen generally takes about 30 seconds.

"Function: Scans video memory and stores the codes in the P array. Blank screen space is stored merely as the last blank location. When the duplication routine (5200-5240) finds a number in P larger than 255 (highest possible code in memory), it skips down to this location and continues with duplication. Unless every screen location is full, P will not be filled.

"This routine can be used with any of the saving routines shown with Fronek's program. Simply write Z and the P array to tape."

If you use Jim's routine, don't forget to add

```
135 B$=INKEY$: IF B$=" "
    THEN 140 ELSE 5000
```

if using the first wandering-pixel program (in the Jan. 1982 column), in which 130 is the SET line and 140 is the following line. Actually, this INKEY line can be inserted almost anywhere in the program, but somehow it seems to fit best, logically, after the SET line that puts a pixel on the screen. □

Micromouth Speech Synthesizer • Model III Features • Software Reviews

September, 1982

Micromouth

Connect ribbon cable on Micromouth to your TRS-80 Model I or III, run an audio cable from the output jack to a hi-fi amplifier or receiver, and then all you have to do is key in

OUT 127,17

and you'll hear a male voice saying "seventeen," very clearly.

If you use a small speaker, such as the two-incher in Radio Shack's "mini amplifier/speaker," the voice will sound a trifle gravelly. For best results, try a larger speaker, at least 5 or 6 inches in diameter.

Micromouth is available from Micromint Inc. (917 Midway, Woodmere, NY 11598) for the Model I ($150 kit, $175 assembled) and III ($200 assembled). You get a plastic box containing the circuit board, plus a connecting cable, separate power supply, and manual.

The Micromouth speech synthesizer is built around the Digitalker DT1050 integrated-circuit set from National Semiconductor, which has a stored vocabulary of 144 expressions. It can be connected to any computer that has an 8-bit parallel input/output port. Connected to the TRS-80, it requires only the simple Basic statement, OUT 127, plus a number from 0 to 143, to make it talk.

Digitalker synthesizes the human voice by waveform digitization, using pulse-code modulation (PCM) to create digitally-encoded speech, and storing the expressions in two 64K speech ROMs.

The vocabulary is limited, but synthesizers that store their vocabularies totally in ROM are generally less expensive than those which assemble phonemes to create any word of your choice. Also, they use a minimum of software.

The vocabulary of the standard Digitalker is best suited to monitoring instruments, to calculators, and to games, consisting as it does of the spoken numbers *one* through *twenty*, *thirty* through *hundred*, *thousand*, *million*, *zero*, letters *A* through *Z*, and words such as *ampere, comma, control, danger, error, flow, gallon, higher, kilo, minute, percent, please, pulses*, etc. Also included are 80-Hz and 400-Hz tones, plus sounds such as *ss* (to make plurals out of words on the list) and *re* as a prefix.

With this vocabulary you can easily create phrases such as "The time is 6:40 PM" or "Danger: a star is on the left at 1,000 meters." Simply put the digital addresses of the desired words in DATA lines, and use a timing loop to allow enough time for a word to be spoken before loading the next one. Without the pause between words, the whole phrase becomes a short, unintelligible bleep. For example, to program that first phrase, which gives the time, try

```
100 READ N
110 OUT 127,N
120 FOR X=1 TO 150
130 NEXT X
140 GOTO 100
150 DATA 138,139,96
160 DATA 6,22,47,44
```

Using a much shorter timing loop will shorten *forty* to *four*. However, although the longer loop provides enough time for *forty*, there's a little too much time between the shorter words. The preferred method of allowing time between words is to check the busy line before loading the next word. This way, speech can sound continuous no matter how long the individual words are. Simply replace lines 120 and 130 with

```
120 GOSUB 200
```

and add

```
200 IF INP(127)=254 THEN 200 ELSE
    RETURN
```

which will space the words just right.

The vocabulary can be extended by using a timing loop too short to allow all of a word to be spoken, and thus shorten *number* to *numb*, *meter* to *meet*, etc. Also, new words can be formed by combining shorter expressions, such as *extenuate* from *X, ten, U, eight*.

The first Micromouth expression called up by OUT 127,0 is "This is Digitalker," spoken by a female voice, which comes across clearer than the male voice used on the other expressions. Perhaps future ROMs will use that female voice; however, the only other speech ROMs available now are a pair (made by National Semiconductor) available from Netronics (333 Litchfield Road, New Milford, CT 06776) for $39.95 plus $1 for postage. The 135 word-vocabulary is also instrumentation-oriented, with words such as *adjust, caution, change, failure, fahrenheit, temperature, yellow, evaluate, pressure, window, water.*

Talking Hangman

A game using Micromouth is *Talking Hangman*; Micromint may have a few left, at $11 each. After you make your first mistake, a head appears in the noose and says, "Try again please." The talking head announces each letter you choose, and either says "Right on" if you get a correct one, or "Error--error-error" if not. If you lose, the word is spelled out vocally, letter by letter, as it is displayed on the screen. Hardly a major application of synthesized speech, but fun for the kids.

Model III Special Features

Several readers have asked for additional information on "the last six items on your list of special features" of the Model III (March 1982, p. 202). One reader thought that "apparently Radio Shack forgot to include this in their manual, and most owners don't know they can renumber lines, use hex constants, and so forth."

Well, I was wrong; actually, those six features are available only in Model III Disk Basic. They are among 25 "enhancements to Model III Basic" listed on page 93 of the Disk Basic manual, which says, "Disk Basic adds many features which

are not disk-related." Once again, those six features are:

- Hex or octal constants can be used (in the form &H7FFF and &777) instead of their decimal counterparts in your programs.
- REM lines or spaces can be automatically deleted from your programs, using the compression routine CMD"C",R to delete REMs, and CMD"C",S to delete the spaces. Using CMD"C" alone will do both.
- Arrays can be sorted; one-dimensional string arrays only, that is, using CMD"O" to sort ("order") them, specifying the number of items to be sorted and the subscript of the first element to be sorted.
- Program lines can be cross-referenced using CMS"X", which searches lines for occurrences of a reserved word or other string literal, and lists the "finds" on the display as five-digit line numbers.
- A string can be searched to see if it contains another string, using INSTR (which can be done in Level-II Basic using MID$ and LEN$; see this column for May 1982, p. 204, for an example).
- Program lines can be automatically renumbered in RAM, using NAME 600,500,10, for example, to renumber all lines from 500 up; the first renumbered line will become 600, and the following lines will be incremented by 10.

INSTRING

Somebody in Fort Worth forgot to include in the Model III manual the INSTRING subroutine that's in the Level-II manual, perhaps because there's the single-command equivalent (INSTR) in Disk Basic. But that's no help to those who have a Model III without disk.

The string-handling subroutine, INSTRING, tests to see if a substring is contained in a larger string. It consists of three lines that use LEN$ and MID$.

First assign the larger or "target" string to X$, and the substring to Y$.

The first subroutine line

```
1000 FOR I=1 TO LEN(X$)-
     LEN(Y$)+1
```

searches string X$ from its first character to the last character that still allows the substring to fit within the larger string. That is, if the substring is three characters long, and the target string is eight long, the search ends at the sixth character, because beyond that, there is no room in the longer string for all of the three-character substring.

The second subroutine line

```
1010 IF Y$=MID(X$,I,LEN$(Y$))
     RETURN
```

returns a value of I that gives the starting position of Y$ in target string X$, or a zero value for I if Y$ is not a substring of X$.

A third line is needed to end the subroutine:

```
1020 NEXT: I=0: RETURN
```

As the Level-II manual notes, a protective end-block is needed so that the subroutine is entered only by GOSUB:

```
999 END
```

The Level-II manual includes a sample program using the INSTRING subroutine:

```
 5 CLEAR 1000: CLS
10 INPUT 'ENTER LONGER
   STRING'; X$
20 INPUT 'ENTER SHORTER
   STRING'; Y$
30 GOSUB 1000
40 IF I=0 THEN 70
50 PRINT Y$; 'IS A SUBSTRING OF';
   X$
55 PRINT 'STARTING POSITION:';
   I,
60 PRINT 'ENDING POSITION:';
   I+LEN(Y$)-1
65 PRINT: PRINT: GOTO 10
70 PRINT Y$; 'IS NOT CONTAINED
   IN'; X$
80 GOTO 10
```

Electric Webster

Cornucopia Software (Box 5028, Walnut Creek, CA 94596) has a new version of *Microproof* (See "Hte Proofreader Porgrams," March 1982), called *Electric Webster*, which adds several features. It can display the dictionary for locating correct spellings. It has a "new precise symbolic dictionary that will not miss an error." It is "even faster than *Microproof* (formerly the fastest available)." And it has two additional options: simple grammatical checking, and automatic hyphenation. *Electric Webster* is $89.50 for the TRS-80 Model I or III, $149.50 for the Model II. The two options are $35 each; the correction option is still $60.

Cornucopia claims *Electric Webster*, using its 50,000-word dictionary, can proofread a several-page letter in 20 seconds.

TRS-80 in London

A recent visit to the Tandy Computer Center on Buckingham Palace Road in London brought out some interesting facts. For one, it's not called a Radio Shack Computer Center (nor a Centre, either). For another, it sells the Model I TRS-80.

Although the Model I could not be sold in the USA after the end of 1981 because of FCC rules on radio-frequency radiation, there are apparently no such rules in England. So Tandy continues to manufacture the Model I for the British and similar markets. "We'll keep on selling the Model I for at least five years," a salesman said. "We have no plans to discontinue it; the keyboard is such a good seller. We'll keep up with the Model I disk drives and expansion interface and monitors and so on, for those who want to upgrade." The TRS-80 runs on 240-volt, 50-cycle current in the UK, by the way.

The best-selling software packages in this center are *Scripsit*, *Profile II*, *VisiCalc*, and accounting packages. The accounting software is not the American versions, "because we use different tax rates, account structures, and so many other details that are different," said the salesman, adding, "Our accounts packages are written in the UK." They carry names such as *Sales Ledger*, *Stock Control Purchase Ledger*, *Nominal Ledger*, and *Corplan*. That last one is a simulation for management decision-making.

The Last One sells for £300 (about $540) in Tandy Computer Centers; it's a menu-driven program generator, which asks many questions about what kind of program you want, and turns out a program written in Model II Basic. "The story is," a salesman told me, "the group that wrote it went looking for somebody to support them during the early days, and Tandy UK loaned them several Model II TRS-80s."

There are about 16 Tandy Computer Centers, some 200 "ordinary stores," and only about 30 franchised dealers in the UK, meaning England, Scotland, Northern Ireland and Wales.

The Color Computer is sold with a different TV set, because of the totally different PAL system, which uses 625 lines on the screen; the US NSCT system uses 525 lines.

The RSC-6 catalog available in the UK looks almost exactly like the US version, with a few differences; the prices are in pounds, the accounting programs are different, the Model I is included, and the Videotex systems are omitted.

Although several years ago the TRS-80s were just about double the US prices, prices are now coming down. In catalog RSC-6, UK hardware prices are 24 to 54 percent above US prices.

Prestel in the UK is similar to Compuserve in the US, but because it has a different format, Prestel can't be accessed by a TRS-80. Prestel is run by the post office, and has "about 10,000 pages of games, databases, etc."

Commenting on customers, a salesman said, "Half the time, the average man who walks in has too much information. He's spoken to somebody in a pub, who

told him to get this or that. Or he may have been told by a friend that he really needs CP/M, and it takes quite a while to tell him it's not quite so. Re-education is the first step in many cases. Many come in, though, wanting *VisiCalc* or a word processor, and they ask for it straight-out."

Two London branches have classrooms with 16 Model III TRS-80s in each, for courses in Basic appreciation, *Scripsit*, *VisiCalc*, etc., which cost $50 to $300. *Scripsit* for the Model III is a three-day course, for $250.

Tandy has two repair services in London. They'll send a service engineer to your location, if you have a service contract. Otherwise you take your TRS-80 to the nearest Tandy store, and "a bloke comes around twice a week to pick them up."

Tandy UK has a high-resolution display for the Model I, from a Bristol-based company, providing 384 by 192 pixels. However, it's character-addressable, rather than by individual pixels. For £180 (about $325) you get 128 user-defined characters. High resolution for the Model III is also available from the same Bristol company, Microware Computing Services, but Tandy UK isn't marketing it, "because there seems to be a Model III version of hi-res supported by Fort Worth," which should be available later this year.

An *Applications Software Sourcebook* is one of the few UK-generated books at the Tandy Computer Centers. This one provides "Over 500 UK Program Listings," mostly for business-accounts programs, and all with UK addresses as program sources.

According to the London *Financial Times*, sales figures for June 1981 show that Tandy was leading the personal computer market with an installed base worth $385 million, Apple was second with $290 million, and PET third with $140 million. "But IBM could take second place by the middle of 1983." More recent figures show Apple ahead of Tandy.

Short Program : Prime Numbers

From Cavan, Ontario, Canada, R.W. Crawford sent this:

"Just a little while ago I was browsing through my past issues of *Creative Computing* and came across 'Short Program 11' in the July 1980 issue (p. 162).

"The program was submitted by Jim Raden and was described by him as not being '...useful at all, except maybe for some type of game.' Perhaps there was some hastiness on his part in saying this.

"I have modified the program to read as follows:

```
10 DIM PN%(200): CLS
20 FOR A%=1023 TO 2 STEP -1
30  FOR B%=15359+A%*2 TO 16383
    STEP A%
35   IF B%>16383 THEN 60
40   POKE B%,170
50  NEXT B%
60 NEXT A%
70 C%=1
80 FOR A%=1 TO 1023
90  IF PEEK(15359+A%)<>ASC(CHR$
   (170)) THEN PN%(C%)=A%: C%=C%+1
100 NEXT A%
110 CLS: D%=C%
120 FOR C%=1 TO D%:
     PRINT PN%(C%);: NEXT
130 GOTO 130
```

"I realized that his program started off by plotting the points that were multiples of 155 and continued to plot the multiples of each integer in ever-decreasing values from the initial value. It took only a short time to realize that if the screen wasn't cleared, the open points after the run of the program would be prime numbers.

"I haven't read the latest articles about finding prime numbers, but perhaps this item would lead to the development of another way to search for them up to a certain value."

For those without that 1980 issue of *Creative*, Jim Raden's original program was:

```
10 FOR A%=155 TO 1 STEP -1
20 FOR B%=15360 TO 16383 STEP A%
30  POKE B%,191
40 NEXT B%
50 CLS
60 NEXT A%
70 GOTO 10
```

□

Graphics • DOSer • Help Numbers

October, 1982

High Resolution

If you're into graphics at all, you're probably frustrated by the low resolution of TRS-80 black-and-white graphics, which as a consequence are very limited in the degree of detail they can provide. Lines are straight only when perfectly horizontal or vertical, circles aren't really circles, and most three-dimensional graphics are out of the question.

Most of the problem is the large size of the graphics blocks (called "pixels" in the trade, short for "picture elements"); the rest is their rectangular shape. In a space 7.5" across and 6.625" down, Radio Shack has placed 128 pixels across, and 48 down. Thus each pixel measures 0.0586 by 0.138 inch—about the size of the hole in an IBM card—giving it an aspect ratio of 3:7. A much smaller pixel, preferably a square, would be much better for graphics. However, if Radio Shack had provided more pixels, the price of a TRS-80 would be higher, and most users probably aren't interested enough in graphics to want to pay for more than the 6114 pixels they've got now.

For those who want more pixels, there are several ways to get the higher resolution. Two companies sell install-it-yourself boards, Radio Shack offers high-resolution graphics for the Model II, and as I reported in the previous column, Radio Shack will offer a similar option for the Model III later this year. I said the Model III hi-res would be available this fall, but according to the latest information, it may not be until winter. After all, *SuperScripsit* was supposed to be

available by 12/30/81, according to the RSC-6 catalog; right now Fort Worth is saying June 1982. So please don't bug your dealer or Fort Worth with questions about when you'll be able to get Model III hi-res; it may be a little late.

Let's look at each of the high-resolution systems individually.

Mikeeangelo Graphic System

For $369, you can raise the resolution of your TRS-80 from the standard 128-by-48 pixels, to 384-by-192 or 191-by-192 if you have a Model I, or to 512-by-192 or 256-by-192 pixels if you have a Model III; you choose the resolution you want via software. That's up to twelve times as many pixels for a Model I, or up to sixteen times as many for a Model III.

This is done with the Mikeeangelo Graphic System, from Mikee Electronics Corp. (Box 3813, Bellevue, WA 98009), which provides 12K bytes of memory in an outboarded enclosure that connects to your TRS-80 via a ribbon cable. For the highest resolution, 512 by 192 pixels in a Model III, 98,304 bits are required, which is 12K bytes. Back in 1977, when the Model I was introduced, that 12K would probably have cost you an extra $290. Not many users would have paid that much for high resolution five years ago.

The manual provides large photographs showing exactly how to install Mikeeangelo, which involves opening the TRS-80, making 18 soldered connections and cutting one trace (or one IC pin). A magnifying glass is supplied, to make sure you leave no solder bridges, and that you've made all the right connections. You also remove the Z80 MPU and plug it into the socket on the interface board. However, unless you've had some experience in soldering IC circuits, you may not want to take a chance on making a mess of your TRS-80, although you may know somebody who can handle the job for you. There's also the problem of voiding the warranty if you (or your friend) open the TRS-80 case. Furthermore, if your TRS-80 needs repairs later, Radio Shack may first have to remove Mikeeangelo, and will charge for the removal.

A demo tape comes with Mikeeangelo and includes some plotting applications. A driver program allows you to SET, RESET or test (with POINT) every dot; LINE draws a line between any two dots. Inverse video provides black-on-white graphics (Figure 1). Through output port 254, you select normal, high or very-high resolution with or without inverse video.

Once Mikeeangelo is installed, you write programs in Basic or machine language; examples of both are provided, but not enough to really show you what the system can do.

The Mikeeangelo brochure explains

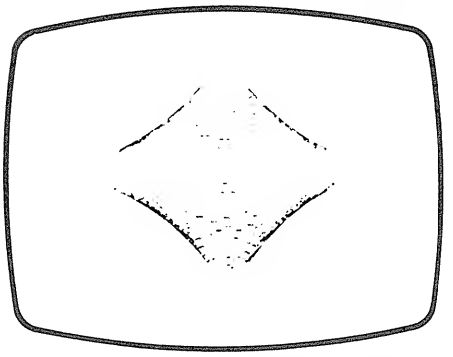


Figure 1. Example of high-resolution graphics, as drawn using the Mikeeangelo board.

why the Model III has higher resolution than the Model I: "The Model III uses an 8 x 12 character matrix instead of the 6 x 12 matrix of the Model I. If you multiply the number of dots along the top of the character cell (eight for the Model III and six for the Model I) by the number of characters per line (64), you get 512 for the Model III and 384 for the Model I. This is the number of dots along the X axis. Next, multiply the number of dots along the side of the character (12) by the number of lines on the screen (16) and you get 192. This is the number of dots along the Y axis. The Model III has a resolution of 512 x 192 dots and the Model I has a resolution of 384 x 192 dots."

Although the Mikeeangelo manual doesn't give enough programming help, there's another source of software for this hi-res system.

Rescom

Bryan Mumford has come up with a number of firsts in TRS-80 software, such as the first disk-cataloging program, the first print spooler, and the first disassembling single-stepping debugger. There's another one, which I've never seen in his ads, the High Resolution Command Module, perhaps because it's very specialized: it's meant to be used with Mikeeangelo.

The first paragraph of the five-page manual for the module says, "The Mikeeangelo high-resolution interface for the TRS-80 Models I and III is a welcomed product. It is well designed and gives us excellent graphics capability. It is, however, like any high-resolution system, awkward to use without the proper software. The purpose of this *High Resolution Command Module* (hereafter called *Rescom*) is to add a group of graphics commands to the existing Basic language. There are 14 commands available, and they will greatly simplify your high-resolution programming." *Rescom* is $24.95 for either Model I or III, from Mumford Micro Systems, Box 400, Summerland, CA 93067.

Included among the 14 *Rescom* commands are: clear the hi-res memory, turn on (or off) one pixel, test one pixel, draw (or erase) a line from one point to another, draw a line from the present point to the past point, print the hi-res image on an MX-100 (or an MX-80 with Graftrax), select hi-res or mid-res, select reverse (or normal) video. That's 12 of the 14; the other two are involved with displaying the hi-res memory, or making it invisible.

E/RAM

You may have seen, last year, ads for E/RAM hi-res graphics for the Model-I Level-II TRS-80, offering 256 x 192 pixels, for $350 from Vern Street Products: The Computer Store in Tulsa, OK.

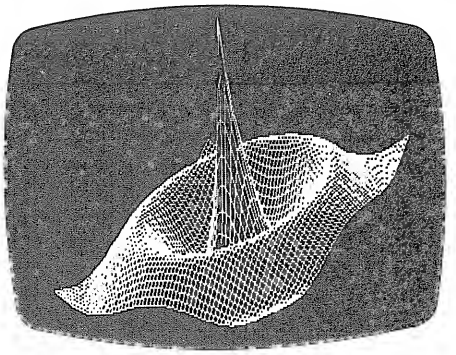


Figure 2. Hires graphics for the Model II TRS-80 provides 153,600 pixels plus eleven new graphics commands in Basic.

Those ads disappeared, and were replaced, early this year, by ads for E/RAM for the Model II, at $495 from Keyline Computer Products (manufacturer of the $350 E/RAM), also of Tulsa. When the Model II ads also disappeared, I called The Computer Store, and was told, "The designer of the hi-res for the Model II gave up when Radio Shack came out with their own hi-res II. He's looked into hi-res for the III, but it was too difficult, no room."

Radio Shack's Model II hi-res (Figure 2), which provides 640 x 240 pixels (for $499 plus installation), is their own design, I understand, and so is the forthcoming Model III hi-res.

There's one more source of hi-res graphics for the Model III, and it requires no soldering.

Grafyx Solution

With the Grafyx Solution add-on circuit board ($299.95 from Micro-Labs Inc., 902 Pinecrest Drive, Richardson, TX 75080), you get three resolutions: high (512 x 192), medium (128 x 192) and low (128 x 96).

There is no soldering with Grafyx Solution, although you do have to cut one trace and remove two ICs from their sockets. Instead of soldering, you clip seven micro-clips onto IC pins; the seven are at the ends of wires connected to the PC board you install in your Model III's innards

These clips are the "E-Z" type used to connect test instruments into densely-packed circuit boards. You press the plastic plunger, and a little hook slides out; you hook it onto an IC pin or component lead, release the plunger, and the test-lead is firmly attached, with no chance of flailing around, as can happen when hand-held test-prods slip (and sometimes zap ICs). However, you may not care for the idea of seven micro-clips dangling inside your Model III case. In one instance, the manual warns you to "be sure to keep the plastic clip away from the socketed memory in that corner since some brands of ICs get quite warm."

One more thing: you need a hacksaw. Before you install the Grafyx board, you must make room for it by removing the metal RF shield. Afterwards, you make a cut in the shield and bend back part of it a quarter of an inch, so it's out of the way of the new PC board.

Once you've installed the Grafyx Solution, which contains 12K bytes of additional read/write memory, you use the supplied software to add hi-res graphics to Basic programs (Figure 4). There are commands to enable (or turn off) the hi-res display, clear the hi-res screen without affecting the standard text characters (the hi-res graphics screen is displayed on top of the normal character display), set the desired resolution, plot a point, read point status, draw a line between two points, complement every point on the hi-res screen for an inverse display, copy the contents of the hi-res and text screen to a printer with graphics capabilities, draw a box whose diagonals ends are at two given points, draw a circle with a given radius and center, shade in the shape surrounding a point (half-shade or solid-shade), copy the contents of a screen rectangle into an array, and take the contents of the array and display it at a given position.

Eighteen demonstration programs and eight utilities are supplied, along with a program that provides an 80-character display for business forms and word processing. The manual goes into theory of operation and assembly-language techniques, and ends with a list of five graphics programs available from Micro-Labs: *Bizgraph, Draw, Mathplot, 3Dplot,* and *PCHAR* (for creating character sets). Only two are priced, at $34.95 each.

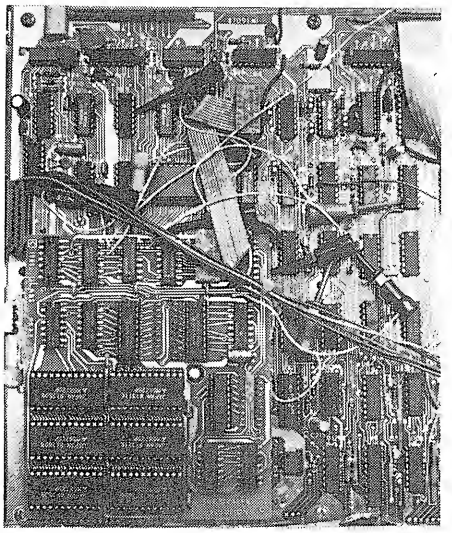


Figure 3. Seven micro-clips connect Micro-Lab's Grafyx Solution board electronically to the TRS-80 Model III's circuit board

The Choice

So there you have the three choices: Mikeeangelo at $369 with a top resolution of 512 x 192, with soldering but with Rescom software; Grafyx at $300 with the same top resolution, without soldering but with micro-clips; and, later this year, Radio Shack's own hi-res offering, with an unknown price and resolution. You may want to wait to see what comes out of Fort Worth, but if you can't, you've got two ways to go.

TRSDOS

When you use a TRS-80 without disk drives, the computer is controlled by the internal ROM (read-only memory), which contains the Basic originally written by Microsoft for the TRS-80. The Basic can't be changed without switching ROMs, and anyway, Microsoft Basic is highly efficient. The highly ingenious "tight code" written by Paul Allen and Bill Gates (which was the beginning of the Microsoft empire) takes up much less memory than most (if not all) of the other Basics.

However, when you move up to Disk Basic, you get into TRSDOS, the disk operating system that performs all the housekeeping required to run the system efficiently. TRSDOS, which Radio Shack provides with TRS-80 disk systems, leaves something to be desired. Even Radio Shack has admitted it from time to time, and has had to publish several TRSDOS releases to clear up various goofs. Even then, various features that many TRS-80 owners consider important are missing from TRSDOS, and several software houses have taken advantage of the lacunae.

Three disk operating systems for the TRS-80 were reviewed back in September 1981 (p. 152): NewDOS by Apparat; TRSDOS 2.3 from Radio Shack; and VTOS, which was renamed LDOS by Lobo International. Author Stephen Kimmel noted several "shortcomings" in TRSDOS, remarking that Apparat "had a different answer that was more to my liking," and said VTOS (now LDOS) "is extremely easy to use," having a 40-page operator's guide instead of a 200-page manual. After specifying various features, Kimmel said, "It should be obvious that both VTOS 4.0 and NewDOS/80 offer significant enhancements on TRSDOS and hence offer more commands.

DOSPLUS

Another very popular TRS-80 disk operating system is *DOSPLUS*, found at computer dealers, or available from Micro-Systems Software (5846 Funston St., Hollywood, FL 33023). *DOSPLUS* 3.3, an older version I started with last year, offered these features:

- RESTORE: a KILLed file can be brought back to life with this unique utility, unless you've written over it with another file.
- DEVICE: displays on the screen all I/O devices connected to your system, and their driver addresses.
- FORMS: goes TRSDOS FORMS several better by adding a value for number of *printed* lines per page and performs top-of-form on the printer, plus several more.
- CRUNCH: compresses Basic programs by removing unnecessary blanks and remarks.
- SPOOL: permits text printing while enabling the program currently running to proceed to the next step.
- TRANSFER: moves all user files from one disk to another.
- Lower-case detection: automatically displays lower-case letters if the lower-case mod is installed (on Model I systems).
- Built-in screen printer: press two keys (S/P, SHIFT/down-arrow, or SHIFT/CLEAR, depending on model), and the information on the screen will be printed out.

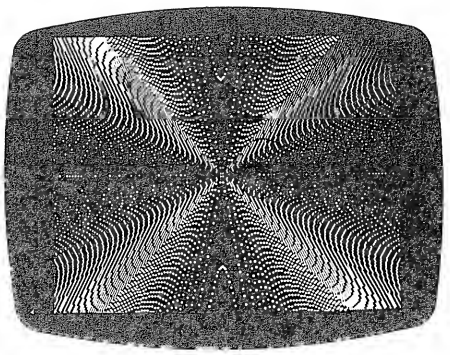


Figure 4. High-resolution graphics using Micro-Lab's Grafyx Solution; note the small size of the dot in this four-way kaleidoscope.

The latest version, at this writing, is DOSPLUS 3.4, which includes all the 3.3 features described, and adds a handful more, including a Basic array sort (multi-key, multi-array), random-access and ASCII modification on diskdump, and a dozen more.

DOSPLUS 3.4 is $149.95, which includes a 240-page manual; you can upgrade from 3.3 to 3.4 for $49.95.

DOSPLUS Basic

Included in DOSPLUS is Disk Basic, which, according to the 3.4 manual, is "a set of enhancements to the Model III ROM Basic, resident upon the disk. It contains features to allow input/output to disk files for data storage and will allow you to load and run Basic programs that are stored on the disk. Disk Basic is completely memory resident on the DOSPLUS system diskette, and comes in two forms—Basic and TBasic."

The DOSPLUS Basic has extended commands and DOS commands, and is less memory-efficient than TBasic (Tiny Basic), which lacks the DOSPLUS system commands and extended features such as extended error messages.

The advantage of TBasic is simple, according to the 3.3 manual: "In a 48K machine, after loading in DOSPLUS and TBasic, you will have just over 40K of free memory left for your program. With regular extended Basic you only have about 37K."

DOSPLUS FORMAT

After writing this column on a Model III TRS-80 under DOSPLUS, I use the FORMAT utility to set up a disk for single density, and then copy the column and also the book reviews onto this single-density disk. That's because the Alpha-Comp phototypesetting machine at *Creative Computing* uses output from an LNW80 computer, which is electronically identical to a TRS-80 Model I, and thus requires single-density disks. (This phototypesetting system was described in detail in March 1982, p. 200.)

Yes, TRSDOS has a FORMAT utility, but it's for double-density disks only. DOSPLUS asks, in addition to diskette name, master password, and which drive contains the disk to be formatted, two unique questions: how many tracks do you want, and do you want single or double density. That feature, without which I couldn't write this column in a format usable by *Creative Computing*, and the RESTORE utility, which brings back to life those files I didn't mean to KILL or PURGE, are alone enough to make DOSPLUS worth having. But they're only the beginning. If you use a TRS-80 more than an hour or two a week for anything other than simple programs, try DOSPLUS. You might not need the FORMAT or RESTORE commands, but there are others that are bound to be valuable.

Incidentally, if you see a list of DOSPLUS features in a Micro-Systems Software ad, many of these are identical with TRSDOS features; the list is partly meant to show how compatible DOSPLUS is with TRSDOS, usually without pointing out which is which. You'll have to compare features one by one to figure it out.

Time Manager

Radio Shack's *Time Manager* is "a personal calendar/reminder and electronic notebook," according to the RSC-7 catalog. It is designed to help you "organize your time and business information, and create permanent records." For $99.95 you can manage your time and appointments (Figure 5), and track job schedules and costs, as the catalog puts it.

Time Manager "organizes daily schedules, quickly summarizing and displaying appointments, tasks to be done, and project information. It can keep records of expenses or mileage for income-tax reports, document important dates, monitor projects and deadlines, and provide time and expense accounting information for billing or internal reports," according to the press release.

Minimum system requirements are a Model I or III with 48K memory, and two disk drives. Optional equipment recommended includes an 80-column printer, a third disk drive, and an amplifier/speaker "allowing *Time Manager* to provide audio responses."

The program is quite extensive and includes four diskettes (one with the program, one for data, and two for backups), a 76-page manual, and a reference card containing all the commands and a brief definition of each.

Calendar

First you call up a calendar by inputting the month and day. When the calendar for the month is displayed, a cursor indicates the day selected; it can be moved to any other day by using the four arrows. Using the right and left arrows, in conjunction with the shift key, moves the calendar from month to month.

Time Manager has two levels, Month and Day. To move from the Month (the calendar display), press Enter, and the Day level appears, showing, via the sample data diskette provided, daily-agenda entries (if any) for the day selected by the cursor.

Each entry consists of four parts: priority, permanence, category, and text. Priority has five levels. Permanent entries are holidays, birthdays, etc. Any of 26 categories (A through Z) can be assigned; T is suggested for telephone calls to make, H for holidays, Y for personal items, etc. The text of an entry can be up to 49 characters long.

Up to 127 entries per day can be put into *Time Manager*, although only 11 will be displayed at a time. A message appears on the bottom line of the screen to tell you how many more entries, if any, are not yet displayed for a particular day. New entries are easily made, or old ones modified.

Now that you've filled out your calendar, you may need to locate an item somewhere on it. To do this, you can select a category, or keyword, or priority, or combinations of the three. Then you can use the automatic scan, which will stop at the first day (forward or backward, as you choose) that contains the type of item you selected.

Totalling and Accounting

Time Manager can also be used for general accounting, and to provide totals for categories such as income received, hours worked and expenses, which sounds like a natural for the self-employed. Up to nine separate accounts can be maintained. Totals for each account can be obtained for a day, a month, the entire year, or any specified time period within the current year.

In the accounting display, there are columns for the account number, account description, sum, multiplier, and total. Running totals are available if desired.

Other Features

With *Time Manager* in Notepad mode,

you can fill the screen with miscellaneous information, such as phone numbers, addresses, lists, reports, assignments, etc. Eight Notepad screens are available, along with a screen editor to modify and display them. Each notepad contains up to 15 lines of 64 characters each.

Two methods of printout are provided. Any of the displays can be transferred to a printer via Screen Printing. To print a Day Level display with more than 11 entries, or to print several displays, or a series of selected entries, Global Printing is available; you select start and end points.

An alarm buzzer can be sounded (if you enter the time in the opening screen) to remind you to make a phone call, for example.

Six "flags" can be set to select special features: uppercase only; expert user (suppresses error messages); etc.

The program can display any calendar page from January 1901 to December 2155, a timespan long enough to take care of the all the appointments for five or six generations. The number of days between any two dates can be calculated.

Entries may be moved from one day to another within a month, or copied from one day to another, or moved from one month to another.

TRS-80 Manager Series

There are more features, but these give you a good idea of what can be done with *Time Manager*, which is designed to "talk" to the other two programs in Radio Shack's TRS-80 Manager Series:

Personnel Manager, at $99.95, keeps track of people, by organizing employee information. Files can be searched for a specific person, product or service. Data can be organized to evaluate vendors, or employee performance.

Project Manager, also $99.95, evaluates and schedules projects, by providing TIME, TASK, PERT and GANTT charts. Projects can be viewed graphically with regard to time, sequence, personnel, materials, or resources. The effect each task has on completing the project can be shown, as well as the effects of task changes.

Dates and schedules entered on one of these three interactive manager programs can be transferred to another, updating appropriate information. All three require a 48K two-disk Model I or III; a printer is optional.

About Those 800 Numbers

You may have read, in Radio Shack's *TRS-80 Microcomputer News* for April 1982, that the toll-free 800 numbers for Computer Customer Service were discontinued on the first of June in favor of

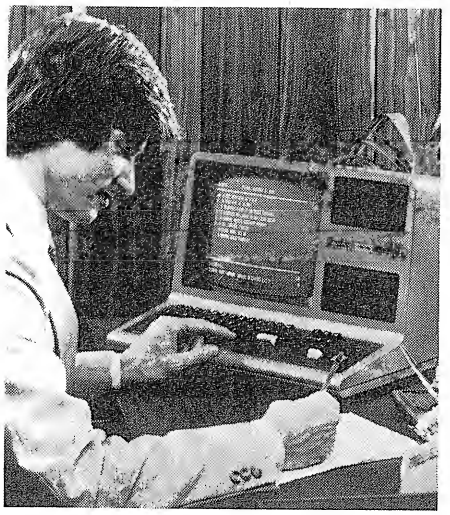


Figure 5. Using Time Manager, this man is scheduling his appointments for next Friday, from "meet with boss at 9 AM" to "call service station."

regular numbers you now have to pay for.

The reason is not because of the cost; the new system will cost Radio Shack more than the 800-number WATS lines. Every time Radio Shack put in more WATS lines, the volume of calls went up, higher and higher, and even with 57 lines there was the same long wait to talk with somebody. Many of the 800-number calls were "trash calls," from people who wanted to know if they could get a particular item at their local Radio Shack store, or some such question. The new system will hopefully eliminate most of the trash calls, and provide more time to answer "concerned calls," from people with serious questions about their TRS-80 computers.

The 800-number system was expensive, costing Radio Shack well over a million dollars a year. The new way, even though the user pays for the call, will be more expensive for Radio Shack because they've committed themselves to having about 180 people on hand to answer the calls. Radio Shack plans to put a Computer Center "in every major (and many not-so-major) markets in the country," and a Customer Service Representative in most of those Computer Centers, in the hope that users will be encouraged to contact their local service reps who can, if necessary, get help from Fort Worth at no extra cost to the user. If the system works as planned, you should be able to get help for the cost of a local call, with a minimum of delay. That is, if we all limit our calls to questions about using Radio Shack hardware and software.

Another reason for the change is that many people, and especially businessmen who own a Model II, for instance, would prefer to pay for a call to find out quickly what they want to know, rather than wait 15 or 20 minutes, or longer, on a free 800-number line for the answer.

About These 817 Numbers

Just in case you didn't see the May 1982 issue of the *TRS-80 Microcomputer News*, Radio Shack has assigned area 817 numbers to seven main groups of customer-support personnel:

- (817) 870-2041—ModelI/III Business
- (817) 870-2042—Model II/16 Business
- (817) 870-2044—Languages and Compilers
- (817) 870-2150—Color Pocket Computer
- (817) 870-2571—Hardware and Communications
- (817) 390-3302—Educational Software
- (817) 870-2271—Games, Books and New Products

These numbers, which became effective on or before June 1, 1982, are to be called between 8:00 a.m. and 5:00 p.m. (Central Time) if, as the *Microcomputer News* puts it, "you elect to call us instead of your local Radio Shack Computer Center Customer Service Representative."

The Navy might abbreviate that to COMCENCUSSERREP, which makes more sense than COCESUSER.

Remember, if we use these seven numbers only when our problems can't be solved by our local Radio Shack store, Computer Department, or Computer Center, the waiting time for help on an 817 line will be kept to a minimum.

New Color Computer

By now you know that the rumors about Radio Shack dropping the Color Computer were only half true. When they recently cut the price of the 4K Color Computer to $299 from $399, it was a preliminary to offering the 16K version for $399.95. Selling a 16K Color Computer for only 95 cents more than the 4K model will, of course, kill sales of the 4K version.

The switch is to bring the Color Computer in line with the competition, such as the $100 Timex/Sinclair 1000 (only 2K, but the price is right), and the 5K VIC-20 and 16K TI 99/4A (both selling for less than $300). □

TRS-80 Word Processing • Money Management

November, 1982

Printers

Radio Shack offers a variety of printers for the TRS-80, and many more are available from other sources. Lowest in cost are the dot-matrix printers, such as Radio Shack's Line Printer II (June 1980, p. 170), first seen at $970 in the RSC-3 catalog of 1979, and last seen at $799 in the RSC-5 catalog of 1981.

Line Printer II, which was the Centronics 730 with a Radio Shack label, was limited in features. It printed upper and lower case letters, 50 a second, in a 7 x 7 dot-matrix format with no descenders, which wasn't all that easy to read. It could also print expanded (wide) characters under software control (Figure 1). That's all it could do, but it operated in both friction-feed and pin-feed modes, and was Radio Shack's cheapest printer using ordinary paper (there used to be several inexpensive printers using aluminum-coated paper less than 5 inches wide). Although not intended for heavy, constant use in a business environment, Line Printer II was (and is) a real workhorse for limited applications, and I have been using mine to print out program LISTings, in uppercase letters, for this column since the May 1980 issue.

In the latest catalog I have on hand, RSC-7, the lowest-price dot-matrix printer is Line Printer VII, at $399. It is similar to Line Printer II, with no descenders for its regular and expanded characters, and prints 30 characters a second. However, it has a bit-image mode that allows printing high-density graphic information, such as from a high-resolution Color Computer display, at 3780 dots per square inch.

Line Printer VIII

For twice as much money, you can get a Radio Shack dot-matrix printer with a fascinating variety of features. Line Printer VIII (Figure 2), at $799, offers proportional spacing, elongated or condensed characters, underlining, superscripts, subscripts, backspacing, 32 special and "international characters" (letters with accent marks) for European use, 64 Japanese katakana characters, and 30 graphics characters (Figure 3). The printer has a word-processing mode, for high-quality printing, using a variable dot-matrix format, with up to 9 x 23 dots; it also has a fast printout, in data processing mode, using an 8 x 9 matrix.

There is a catch, though. Those fine features aren't all that easy to use. What is worse, the manual makes them seem even more complicated than they really are, having been written more for an engineer than for a businessman or a hobbyist.

The first dozen pages, all about setting up Line Printer VIII, are straightforward and fairly simple. Then the manual gets into operating the printer, and within a page or two manages to confuse most readers with some rather obtuse sentences and long paragraphs about control codes, character font selection, a mystifying page on line feed, and then some really mysterious esoterica about positioning, proportional spacing and such. A computer user buys what he thinks is a fairly simple printer, and then finds out, from the manual, that he has something about as simple to learn to use as a Learjet.

But wait, there seems to be some hope. Page 35 is the beginning of a section on Programming Information, and for a moment you think, "Here's where they make it simple for those of us who didn't major in Computer Science." Wrong again! Fourteen examples are given. Not "printing examples," but "programming examples," which should be a clue that there is no simple way to use Line Printer VIII. Turns out you have to use LPRINT and CHR$ to do anything other than simple, straightforward printing. However, one of the most maddening things about this maddening manual is that there is no simple chart of the CHR$ commands. You have to figure out which is which from the programming examples. That is, unless you realized that the decimal codes cleverly concealed in the Control-Code Summary chart back on page 24 are the CHR$ codes to use.

Okay, once you figure out the proper CHR$ codes, you realize they have to be used *each and every time* you want to switch from one mode to another. To print a simple chemical formula involving five compounds using subscripted numbers and superscripted plus and minus signs, takes 35 CHR$ codes. To make things worse, the whole thing is in Basic, as are all the "programming examples." Suppose you want to use some of these fascinating features in *Scripsit*. There isn't one single mention of *Scripsit* in the entire manual. If you have high blood pressure and no engineering degree or electro-mechanical inclination, don't open this manual; it is the most complicated I have yet seen come out of Fort Worth. Apparently they don't believe in trying out their manuals on ordinary folk before publication.

To start elongated printing, you have to use

CHR$(27); CHR$(14)

and to go back to normal printing, use

CHR$(27); CHR$(15)

while to start proportional spacing you need

CHR$(27); CHR$(17)

and to go to condensed printing requires

CHR$(27); CHR$(20)

and then, to get back to ordinary printing

CHR$(27); CHR$(19).

To print the chemical formula for water, H_20, takes

LPRINT"H"; CHR$(27); CHR$(28); "2"; CHR$(27); CHR$(30);"0"

with the CHR$(28) required for a "half forward line space," meaning moving the paper up half a space, and the CHR$(30) required for a "half reverse line space," which moves the paper back down the half-space. Got all that? If you do, and are willing to go through it all, and in Basic yet, you're a better man than I am, Gunga Din.

Yes, some of the features—but only a few—can be used in Scripsit, which you have to find out by experimenting.

However, as usually happens when Radio Shack omits an important feature,

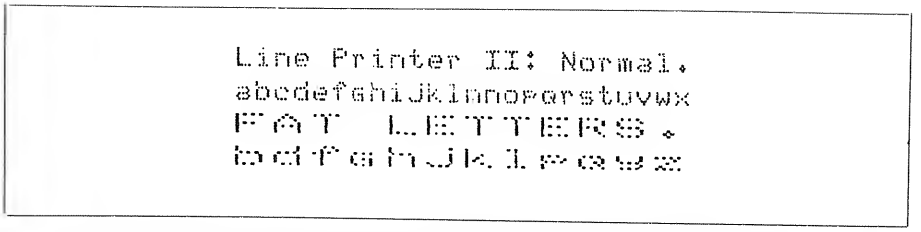


Figure 1. Examples of printing with Line Printer II, showing the lack of descenders in both regular and expanded characters.

such as not telling you how to use a printer with *Scripsit*, an independent, outside company offers help.

Scriplus To The Rescue

One of the most interesting and useful print utility programs available is from Powersoft (11500 Stemmons Expressway, Dallas, TX 75229): *Scriplus 3.0* ($39.95 plus $5 for shipping and handling).

Scriplus 3.0 is "a modification to *Scripsit* that lets you take advantage of the special functions, features, and print formats of your printer while your document is being printed," as the Powersoft ad puts it. Although designed specifically for Epson printers, it works with any printer that accepts CHR$ commands for control.

Scriplus permanently modifies the disk version of *Scripsit* to allow you to: send commands to the printer to activate special formats and functions; stop the printer so you can insert text, align forms, or change print-heads; optionally select line feed after carriage return; list alphabetically a disk directory from within *Scriplus* and fetch, chain or kill any file right from the display; and edit inserted text before resuming printing. *Scriplus* modifies all versions of *Scripsit* for Model I or III machines, allows you to use Model I *Scripsit* on a Model III, and works with all well-known Model I or III operating systems, including LDOS, DOSPLUS, NEWDOS, TRSDOS, and DoublDOS.

How Scriplus Works

The way *Scriplus* works is simplicity itself. You just use the control codes, without CHR$, in *Scripsit*. For instance, to print elongated characters, just put

27,14,

ahead of the words to be stretched. And if you want to go back to normal printing in the same text, just put

27,15,

in front of the part of the text to be printed normally. Characters that don't appear on the keyboard, such as bracket can be printed out by using their ASCII equivalents in the control code lines. A question mark stops the printer so you can change printwheels, insert text, or position the paper for forms alignment. You can read the disk directory from within *Scriplus* by pressing BREAK and then typing a colon followed by the drive number.

These are only a few of the many features offered by *Scriplus*, which are described in rather small print, and not always written clearly enough for the uninitiated (but most beginners aren't ready for *Scriplus* anyway), in a little 14-page manual that often goes into detail, with a variety of examples.

Sound effects are included: you get a plunk-plunk sound as the ball bounces around the game area, and various witchoo-witchoo, wowee and beeoop sounds as the ball hits targets. The little eight-page manual says "a good player should score about 30,000" points. Well, I didn't get near that, but the game is addicting enough that I didn't want to try for the 30,000, or for the 80,000 an excellent player is supposed to be able to score, because I could get hooked, and there were two more Acorn programs to check out. One of the hooks of this game is that a fair amount of luck is involved, because the path of the ball is beyond your control—and depends only on how far back you pulled the "plunger"— until it gets within range of the flippers. You keep telling yourself luck will be entirely on your side in the *next* game, and luck,

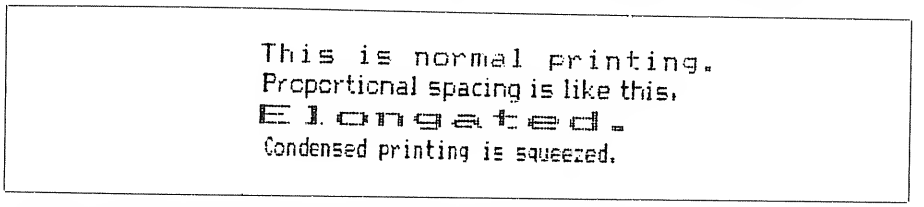


Figure 3. The character set of Line Printer VIII includes Roman letters in various shapes, Japanese characters, and graphics blocks.

Figure 8. These 15 options are displayed in the program menu for Money Manager.

```
(A) ADD NEW ENTRIES TO THE FILE
(B) DELETE/MODIFY CHECKBOOK ENTRY
(C) REVIEW ANY CHECKBOOK ENTRY
(D) CHANGE/DISPLAY BALANCE
(E) PRINT ALL CATEGORIZED ENTRIES
(F) RECONCILE CHECKBOOK
(G) REVIEW COMPLETE CHECKBOOK FILE
(H) REVIEW CHECKBOOK FILE BY MONTH
(I) REVIEW CATEGORIZED ENTRIES
(J) REVIEW DEPOSITS
(K) REVIEW BANK CHARGES
(L) REVIEW MISCELLANEOUS WITHDRAWALS
(M) REVIEW OUTSTANDING CHECKS
(N) MODIFY/LIST CATEGORIES
(O) DISPLAY CATEGORY/MONTH MATRIX
```

combined with your ever-increasing skill as you play more and more, will give you a much higher score than ever before.

Lost Colony

The Acorn ads describe the game thusly: "You are the Economic Manager of the world's first space colony. The next support ship from Earth isn't due for another 15 years and you have instructions either to make things go better or get out of office in shame. You must assure the survival of this struggling space colony—it's all up to you. You'll be presented with the human, natural and industrial resources of the planet. You must allocate labor, explore new territories, decide on production quotas, determine pay scales and taxes for the most productivity—you're armed with maps and charts. 10 levels of difficulty."

The 23-page booklet goes into a great amount of detail, and the first page notes this is "a complex simulation. It is suggested that you read this documentation carefully before attempting serious play." As you play, you are presented with chart reports involving five industries: farm, minerals, energy, manufacturing, and transportation. You decide who works where, what is manufactured, how many factories are built, what the standard of living is, how much capital equipment, distribution of consumer goods, what is stockpiled, what tax rates are changed, when to explore new territories, how to settle new territories, etc. You get a year-end report when you have done everything, and then you can begin a new year, in which you try to do better. But if the standard of living isn't up to the workers' expectations, they will strike or throw you out of office. How's that for a challenge?

The game is almost entirely based on displayed reports (Figure 7) which show, in numbers and bar graphs, how you have allocated resources. The manual provides minute detail on how to play the game, and also provides a couple of pages of hints, such as "Explore the continent and settle only the lush new areas," "Plan on growing more food than you need," and "Watch the bar graphs in the various reports to get an idea of how developments in one industry affect the others."

If you are into resource management simulation, the space age features of *Lost Colony* can be quite intriguing; similar simulations may well be in actual use 100 years from now, or whenever space colonies become practical. The game isn't all reports; there is a small, fascinating map of the continent on "Warren's World," the name bestowed by author

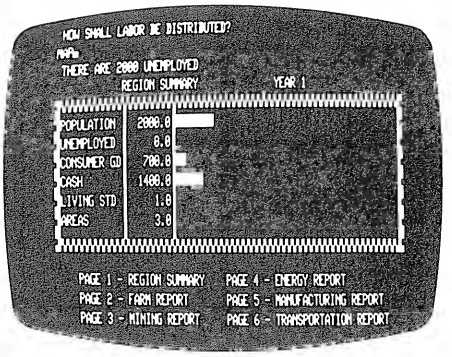


Figure 7. One of the reports generated by Lost Colony, with a bar graph showing several of the resources.

David Feitelberg on the colonized small planet, which he may have named after Jim Warren, one of the personal computer pioneers. The map is divided into 45 areas; at any stage of the game you can call up the map and ask for a display of either the settled areas or the resource (raw materials) sites.

Money Manager

This is a menu-driven personal finance management program that allows you to define your own categories of expense (up to 99 of them) and has a built-in category for tax-deductible items. *Money Manager* permits storing data in as many as 100 checkbook entries per month (250 with a 48K system, but you will need more than one disk drive; each check-writing year has its own diskette). Automatic withdrawals can be specified, and lump payments (such as often occur with credit cards) can be broken down into several categories. The program was developed for output to an 80-column lineprinter, and allows formatted printouts by category and time period. (If you use a printer with less than 80 columns, you must change printer formatting in the program.)

If you are using a TRS-80 Model III, you must use the Convert utility to make this Model I disk compatible with your machine. Then with a RUN"MONEY" the checkbook program loads into memory and executes. When you press ENTER, the first menu (Figure 8) is displayed.

Although option A may seem the first to call up, you must first assign a list of categories, using option N. "The ability to establish categories reflecting your own needs and goals is one of the features that makes *Money Manager* so valuable," says the 18-page manual, adding "We can't decide how much you should spend on what—this isn't a budget-making program—but once you have decided, you can use the categories to monitor how well you are doing. The printing capability makes it possible to know where your money is going as well as whether a category needs breaking down further or eliminating."

The manual notes that perhaps you don't want to begin by making out a budget. No problem; just plan and enter your categories, and the program will let you monitor your expenditures. "If you want to use your actual living expenditures as a basis for planning later, you'll find yourself in possession of much valuable information about your lifestyle."

So first you assign categories. The program reserves the first category for tax-deductible items, but from then on, you're on your own, and can add or delete categories, or change their names, or display or print them. So you set up categories for rent, phone, utilities, entertainment, transportation, etc.

Then you get into option A, adding new entries. This is where the bulk of the work is, adding newly written checkbook entries and deposits to the file. The first time around, the program asks how many disk drives you are using, what the check-writing year number is, and what your checkbook balance is. The manual notes that "The maximum balance allowed in the TRS-80 version is $999,999.99," which may be a little low for some readers, but which should be adequate for the majority.

You enter the check number, date, description, amount, category, and whether the item is tax-deductible. Then you get a new balance display, and the program asks if the information is correct and if you have any more entries.

Other options permit you to delete or modify a checkbook entry, review any checkbook entry, display or change the balance, print all categorized entries, and so on down the line of over a dozen options.

In option F, for balancing your checkbook, all outstanding checks are displayed. You indicate the cancelled checks, and if the program-generated balance matches the balance printed on your bank statement, you can enter the cancelled checks into the file. If it doesn't, you go back to the main menu to start looking for the error.

If you are willing to spend the time entering all your expenditures and deposits into *Money Manager*, you may be quite surprised to find out where your money is really going. And then, even if you are not on a budget, you may decide to spend a little less here, or a little more there. At this point in the economy, can

you afford *not* to know where your money is going?

Short Program #33: Guessing Game

Can you predict what sort of pattern this graphics program will generate?

```
100 CLS
110 X=RND(45)
120 Y=RND(X)
130 SET(X,Y)
140 GOTO 110
```

The answer is very simple, but you may have to RUN the program first, and then figure out why the pattern looks the way it does. Add

```
135 RESET(X+1,Y+1)
```

to turn it into a constantly-changing pattern. □

Leo Christopherson • New Games • Blank Program Lines • PC-1

December, 1982

Radio Shack Signs Christopherson

Leo Christopherson, one of the top computer-graphics artists in the world of personal-computer games, has taken a leave of absence from teaching to write games for Radio Shack.

Leo is famous for programs that are outstanding for their animated graphics and ingenuity, such as *Android Nim* (June 1979, p. 125), *Cubes* (Aug. 1980, p. 162), *Snake Eggs* (Aug. 1980, p. 162), *Lifetwo* (Aug. 1980, p. 162), *Dancing Demon* (Oct. 1982, p. 178), and *Voyage of the Valkyrie* (June 1982, p. 220).

Leo has been teaching mathematics and "sometimes science" to seventh and eighth graders in a junior high school in the Franklin Pierce School District outside Tacoma, Washington.

For a year, and possibly a second year, Leo will write graphics games, primarily for the Color Computer, working at home at his own pace. "They have left it pretty much up to me, as to what I'll work on," he told me. As part of his contract, Radio Shack donated 17 TRS-80 Model III computers to his school district, to be used in a network arrangement.

Radio Shack had tried in the summer of 1981 to get Leo to write full-time for them, but by then he had already signed his annual teaching contract. They asked again later, and he got a leave of absence. Both Radio Shack and the school district have agreed to a second year of full-time games programming, if Leo wants it.

Radio Shack is not the only computer manufacturer, of course, to offer Leo a job. Atari wanted him to travel around the country, giving lectures and seminars on what an Atari computer can do, but he didn't want to travel that much.

Keep your eyes open for Leo's programs under the Radio Shack label.

Two Games from Liberty

Let's look at the first two programs marketed by Liberty Software Co. (635 Independence Ave., S.E., Washington, DC 20003), which specializes in games and educational programs. The two are games that can be played on either a Level-II Model I or Model III TRS-80 with 16K of memory: *Alien Armada* ($13.95 tape, $17.95 disk) and *Golfer's Challenge* ($13.95 tape). If you can't find these at your local computer store, Liberty can supply them, if you include $2 for shipping and handlng.

Liberty Software, by the way, is the label Acorn Software Products (also of Washington) is now using on games, because Acorn is moving "in the direction of software for more personal use, such as *Superscript* and *Instant Sort/Search Database*," one of their people told me. (Those two Acorn programs will be reviewed here in a few months.)

Alien Armada

This is a fairly standard arcade-type game, where you use the left and right arrows to move the "rocket base" left and right, and the spacebar to fire rockets at the attacking aliens, who drop bombs that can wipe you out (with accompanying sound effects) unless you move fast. If you are quick enough to wipe out the armada, along comes another, and so it continues, one alien fleet after another, as long as you live.

This is a very fast-moving real-time machine language game, for one or two players, requiring concentration and quick reflexes, even at the beginner level. Once you develop some skill, you can

move to the two higher levels of difficulty, which are guaranteed to make your heart pound even if you are an expert at such games. Warning: You probably shouldn't play *Alien Armada* if you are not supposed to get excited.

Golfer's Challenge

This game of golf, for one or two players, generates 18 new holes, complete with sandtraps, waterholes, and trees, every time you play. You have a choice of 13 clubs (a driver, three woods, eight irons and a wedge) and of seven directions to aim the ball.

The holes are drawn for you, as straightaways or doglegs, one at a time, and the display shows the tee, 150-yard distance markers from the hole, various fairway hazards (T for a tree, S for sand, W for water), the green, and a number that represents the distance from tee to hole.

You select a club and direction of aim, using a table that indicates the possible yardage of each. The ball doesn't always go where you want it to, of course. When you get on or near the green, you don't putt; the computer takes over and assigns two or three shots to make the hole. After each hole, a scorecard is displayed.

Sound effects are available through earphones or the recorder AUX jack, and consist mostly of a chirping, accompanied by a flashing pixel, to show where your ball is.

The game is interesting for a round or two, but unless you are a golfer, or want to learn something about the game, or get some pleasure playing this against an opponent, you probably won't play it very long. The graphics are clever, but the ball doesn't sail through the air, and the clubs are never seen on the screen.

Blank Program Lines III

In writing about how to put blank lines between sections of your program listings, the June 1982 column (p. 217) said Radio Shack's Line Printer II ignores down-arrows, and that you must, when using the II or other printers like it, "add spaces the hard way: one at a time."

Don Davidson of Silver Creek, NJ has a marvelously simple solution:

"You had a suggestion for a better way to put blank lines in program listings using the down-arrow. It did not work, however, on a Line Printer II.

"It is almost as easily done on a Line Printer II, though. Just add a space after the down-arrow. For example, to leave two spaces, just add (down-arrow)(space) (down-arrow)(space) at the end of the line."

It works. Of course, after the (down-arrow)(space), you have to press ENTER, to get the prompt for the next line.

PC-1 as Portable Data-Logger

In the PC-2 section of the RSC-7 catalog, it says "Radio Shack will introduce an RS-232C interface and software to allow the PC-2 to function as a terminal." Well, you can get a hardware/software package right now that will let you use the earlier PC-1 Pocket Computer as a portable data-logger.

With the PTR Interface from Protean Scientific (Route 13, Lincoln, NE 68527), you can use the TRS-80 PC-1 to transfer data tapes from the Pocket Computer to a TRS-80 Model I or III. The PTR Interface connects directly to cassette recorder cables of the desktop computer, and doesn't have to be removed during normal cassette operation. An LED flashes to indicate the transfer of data. The interface operates from batteries, and thus cuts down on cable clutter.

The PTR Interface reads both alpha and numeric data, so you can either load datafiles as they occur on the tape, or search for datafiles by name. You can load complete files, or specify the number of memories you want transferred. Checksum errors identify invalid data.

Protean Scientific provides a sample Basic program which calls an assembly-language routine. The PTR package, which is $99.95 plus $2.50 for shipping and handling, includes the assembled electronic interface, a cassette tape with the assembler and Basic programs, and a detailed 15-page user manual. The software works with a Model I or III and is supplied in three versions—for 16K, 32K and 48K systems—on the same tape.

According to a letter from Protean Scientific President Robert K. Nickel, the interface "is essentially the outgrowth of a need we perceived for a low-cost data-logger for geophysical measurements. We believe that there are many possible applications in fields such as geology, archeology, botany, zoology, and agronomy. Virtually any specialized study which involves the collection of data in field or laboratory environments could realize greater efficiency and accuracy by using a pocket computer to record data." There are also, of course, many business uses for a data-logger, such as taking inventory.

Using the PTR Interface

The PTR Interface connects between a cassette player and the cassette port of a TRS-80 Model I or III. The interface modifies the data signals from a PC-1 tape to be compatible with the tape-reading circuit of the desktop TRS-80 and the PTR assembly-language program.

The software consists of two programs, the PTR assembler program which transfers data from the PC-1's data tape to the memory of the desktop computer, and a Basic demonstration program which shows how PTR is called and how the data received from PTR is converted into numeric and string Basic variables. Part or all of the Basic program can be incorporated into your Basic programs.

First you connect the PTR Interface in series with the TRS-80 cassette cable and the earphone jack of the cassette recorder. Using the SYSTEM command, you load whichever version of the PTR assembly language program is appropriate to the memory size of your Model I or III. Then load the Basic program, and make a couple of small changes in the listing, depending on the size of your memory size.

Now you are ready to go. Turn on the interface switch, put a PC-1 data tape in the cassette recorder, and RUN the Basic program. You will be asked to enter the name of the file you want to read, and how many registers you want to read from the data file. Pressing ENTER will result in all registers being read.

If you have entered TESTX as the file name, the screen will display

```
SEARCHING FOR:    TESTX

NO DATA STREAM
```

until the first file is found. If there are other files on the tape before TESTX, the display will show their names to indicate which files are being skipped and also show the contents of the data registers as they are passed; for example:

```
SEARCHING FOR;    TESTX

SKIPPING:         TESTB

+1.782010312D-02
```

When the desired data file is reached, the display changes to

```
LOADING:          TESTX
```

meanwhile also displaying an error count and register count. As soon as the file is loaded, the display changes completely, and the register contents are displayed; for example:

```
REG. 1= SCA5
REG. 2= 3423
REG. 3= 93275
```

and so on, until the contents of all selected registers are displayed. Errors are flagged and counted.

That is all there is to it; using the PTR Interface is about as simple as possible. The manual explains all operations in a very clear way, and also includes several pages on Theory of Operation, plus a listing of the Basic program, should you want to use it in one of your own programs.

Radio Shack/CitiLine Credit Card

For several months now, there has been a new credit card available "for qualified consumers who want a revolving loan account with Citibank...for big-ticket purchases from Tandy's Radio Shack stores. The card can also be used to purchase financial services," according to the press release.

The new card, called Radio Shack/ CitiLine, is said to be the first such national "co-branded" bank card. It is accepted only at Radio Shack stores "in the 48 contiguous states," and is for financing a purchase of $225 or more. Borrowers pay about two percent a month; there is no annual fee.

Now you can get "instant credit" for buying a Radio Shack hi-fi or music synthesizer, or even a TRS-80 computer.

Line Printer VIII Graphics

Last month's column featured Radio Shack's dot-matrix Line Printer VIII (p. 306), which offers proportional spacing, elongated or condensed characters, underlining, superscripts, subscripts, and many more features, including 30 graphics characters.

Each of the 30 characters consists of six vertical dots in various combinations, which the printer manual says can be used "for drawing pictures, figures, or graphs." Fifteen of the characters are four small squares in all their combinations (the last of which is a larger square), eleven short lines and sets of intersecting lines that can be used to construct forms, and four triangles that face in different directions.

Bill Fronek, of Houston, TX, sent the above program and RUN to give some idea of what can be done with columns of dots, but in this case, it's all possible combinations of seven dots.

Line 50 puts the printer in graphics mode, which is a bit-image printing mode, and thus quite different from the character printing mode ordinarily used. In graphics mode, printable data must be in the range 128-255; you can print any combination of seven dots in a dot column. Each dot is controlled by one bit in the data byte, in which the high bit

```
10 '----GRAPHICS MODE PRINTABLE DATA IN THE RANGE 129 - 255 ---
20 '-----------FOR TRS-80 LINE PRINTER VIII---------

30 LPRINTCHR$(30);' (END GRAPHICS MODE TO ENABLE LPRINT A)
40 FOR A=129 TO 255
50 LPRINT A; CHR$(18);' (LPRINT A, THEN ENTER GRAPHICS MODE)
60 LPRINT CHR$(255); ' (LPRINTS VERTICAL LINE FOR REFERENCE)
70 LPRINT CHR$(128); ' (LPRINTS SPACE)
80 LPRINT STRING$(10,A);'(LPRINTS 10 DOT LONG STRING OF CODE A)
90 LPRINT CHR$(30); '(END GRAPHICS MODE TO ENABLE LPRINT A )
100 NEXT A
```

```
129 ▔ 130 ▔ 131 ▔ 132 ▔ 133 ▔ 134 ▔ 135 ▔
136 ▔ 137 ▔ 138 ▔ 139 ▔ 140 ▔ 141 ▔ 142 ▔
143 ▔ 144 ▔ 145 ▔ 146 ▔ 147 ▔ 148 ▔ 149 ▔
150 ▔ 151 ▔ 152 ▔ 153 ▔ 154 ▔ 155 ▔ 156 ▔
157 ▔ 158 ▔ 159 ▔ 160 _ 161 _ 162 _ 163 _
164 _ 165 _ 166 _ 167 _ 168 _ 169 _ 170 _
171 _ 172 _ 173 _ 174 _ 175 _ 176 _ 177 _
178 _ 179 _ 180 _ 181 _ 182 _ 183 _ 184 _
185 _ 186 _ 187 _ 188 _ 189 _ 190 _ 191 _
192 _ 193 _ 194 _ 195 _ 196 _ 197 _ 198 _
199 _ 200 _ 201 _ 202 _ 203 _ 204 _ 205 _
206 _ 207 _ 208 _ 209 _ 210 _ 211 _ 212 _
213 _ 214 _ 215 _ 216 _ 217 _ 218 _ 219 _
220 _ 221 _ 222 _ 223 _ 224 _ 225 _ 226 _
227 _ 228 _ 229 _ 230 _ 231 _ 232 _ 233 _
234 _ 235 _ 236 _ 237 _ 238 _ 239 _ 240 _
241 _ 242 _ 243 _ 244 _ 245 _ 246 _ 247 _
248 _ 249 _ 250 _ 251 _ 252 _ 253 _ 254 _
255 _
```

(value 128) is always on, so that, in effect, the printout above represents the binary numbers 1 through 127.

Line 80 prints a 10-dot-long string of columns. Note that, if you select the right dot combinations, and string them together properly, you can create almost any dot-matrix character you wish, from foreign alphabets to tiny pictures.

The catch with trying to print the 30 graphics characters referred to above is that the manual doesn't tell you how to do it. There is a table of printing codes, but these codes range between 225 and 254, which are the same codes used for bit-image printing. The Line Printer VIII manual is highly frustrating, to say the least.

Short Program: Keyboard Graphics

A letter from Cincinnati includes a program for the Model-I Level-II computer:

"Did you know that it is possible to place graphics on the screen from the keyboard? The following program is one method of doing it and can be used in all sorts of games.

```
5 CLS: PRINT
10 INPUT "ENTER A NUMBER
   BETWEEN 64 AND 69"; X
15 CLS: PRINT
20 B$=INKEY$
30 PRINT CHR$(ASC(B$)+X); " ";
40 GOTO 20
```

"This program gives you direct control over 58 of the graphics characters at one time, using shifted and unshifted entries (there is a total of 63 graphics characters). In this range of X, all of the letters of the alphabet give graphics as well as (sometimes) # @ = ?<>. I would appreciate knowing if someone can figure out how to control all of the characters."

If you have written a game or two using INKEY$, you may realize, without running this program, that it won't work. It was typed, and since few of us are great typists, extra care must be taken when typing up a program instead of using a printer.

RUN this program, and you get

?FC IN 30

or on a Model III,

Illegal function call in 30

which may, of course, lead you to think something is wrong with line 30. What's that pair of double-quotes for? Nothing, apparently, so take them out. And you get the same error message.

If you assume the error is in line 30, you might pare the problem down to what may seem the heart of the problem:

```
20 B$=INKEY$
30 PRINT ASC(B$)
```

Run this and you'll get the same error message, which is somewhat mystifying because, according to the Level II manual, this pared-down version of line 30 is OK.

So, finally, you look at page 5/5 of the Model II manual (page 166 of the Model III manual), and you find, "INKEY$ returns a one-key string determined by an instantaneous keyboard strobe. If no key is pressed during a strobe, a null string (length zero) is returned....Because of the short duration of the strobe cycle (on the order of microseconds), INKEY$ is invariably placed inside some sort of loop, so that the keyboard is scanned repeatedly."

After reading the rest of the text about INKEY$, you soon realize all you need add to the Cincinnati program to make it work is a simple loop:

25 IF B$="" THEN 20 ELSE 30

Enter a 64 after you RUN the program, and keys A-Z plus # and @ will give you a total of 54 graphics characters, with the graphics codes 129-156 and 160-186. That's using the SHIFT key, too.

If you enter a 69 after you RUN the program, you shift the "window" five places to the right, so to speak (so to write?), and by using keys A-Z plus = @<>? @ you obtain 59 graphics characters with codes 129-160 and 165-191. That takes care of all 63, counting the other set also.

So what the last sentence in the Cincinnati letter must mean is: Can you write a program that will generate all 63 graphics characters from the keyboard, without having to use ENTER or anything similar in line 10? The answer involves a knowledge of ASCII codes below 32, and between 91 and 95, inclusive. Assuming, of course, that there is a program that will do the trick. □

Mystery: What was the original, working version of the Cincinnati program? Was the one-space string at the end of line 30 originally part of something like line 25? What do the two PRINT statements do for the program?

Hi-Res Model III • Copy Art

January, 1983

High Resolution

High-resolution graphics for the Model III was first mentioned in the September 1982 column (p. 211), after I'd heard in London that it was forthcoming from Fort Worth. The second mention was in the October 1982 column (p. 284), in which I described the Mikeeangelo Graphic System ($369 for up to 512 x 192 pixels; recently renamed Mikeegraphic and reduced to $340), and Micro-Lab's Grafyx Solution ($299.95 for up to 512 x 192 pixels), and said readers might want to wait for Radio Shack's offering.

Last September, Radio Shack's new RSC-8 catalog included several new products, among them high-resolution graphics for the Model III, at $369.95 (plus installation) for 640 x 240 pixels (Figure 1). That comes out to about 85.33 pixels per inch horizontally, 36.23 per inch vertically, which does indeed sound like it could provide what the catalog calls "amazingly fine detail," for "sophisticated business graphs, tables, charts, maps, illustrations, geometric patterns—and animation!"

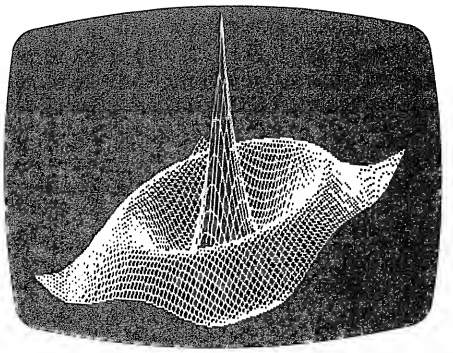


Figure 1. High-resolution graphics, as offered for the Model III TRS-80, provides 153,600 pixels for fine detail.

The package includes a 32K RAM board, manual and diskette with Graphics Basic and a library of assembly language subroutines. The Graphics Basic includes commands for drawing a circle, arc or ellipse; drawing a line between points; shading an area in one of several available patterns; turning an individual pixel on or off (with PRESET); putting the bit-pattern contents of an array onto the screen (useful for animation); and turning the graphics screen on or off.

The RSC-8 catalog says the Model III hi-res graphics package is "available 12/30/82." Mebbe. Anyway, it offers almost 55,300 more pixels than either Mikeegraphic or Grafyx Solution, for $30 more than the first, and $70 more than the second. However, let's wait to see how the Radio Shack hi-res software compares. Byron Mumford's *Rescom* (Oct. 1982, p. 286) has 14 commands for Mikeegraphic; and the software supplied with Grafyx Solution looks even better,

with just about every one of the Radio Shack Graphics Basic commands, plus some others that Fort Worth doesn't seem to have included: complement every point on the hi-res screen for an inverse display; draw a box whose diagonal's ends are at two given points; and copy the contents of the hi-res and text screen to a printer with graphics capabilities. (However, it may be possible to create an inverse display by using the POINT and PRESET commands). We'll see, as soon as I can get the Radio Shack hi-res graphics board installed in my Model III.

Note that the high resolution (640 x 240) provides exactly five times as many pixels in both the horizontal and vertical directions as does the standard resolution (128 x 48). Which would mean, if the hi-res graphics area is the same rectangular shape, with the same aspect ratio, 3:7, but only a fifth as large. However, looking very closely at Figure 1, you can see that the pixels are almost square (rounded, in the photo), which makes graphics easier to create. And one of Radio Shack's software people confirms that the hi-res pixel is indeed almost square, with a 4:5 aspect ratio. How come? Well, he says the hi-res graphics area may be bigger than the lo-res graphics area. We'll check that out later, after acquiring a hi-res board.

The 153,000 pixels take 153,000 bits of memory to store. If Radio Shack uses a 32K graphics memory board, that means it provides 32,000 times 8 bits, or 256,000 bits, which means there are 102,400 unused bits on that board. So why not put only 20K, or 160,000 bits of RAM memory, on the graphics board? According to a Radio Shack source, it may simply be more economical "to use the chips we chose," which must mean they need stock only one size of RAM chip, perhaps a 64K chip, for the hi-res board. Somebody will undoubtedly find a way to use the 102,400 idle RAM bits.

The graphics memory, by the way, is independent, and can overlay text in the regular video memory, so graphics and text can be combined in one display.

The assembly language subroutines are there only if and when you want to use the graphics commands from a language other than Basic, such as Fortran or Cobol. If you use only Basic, you'll never need those subroutines.

Hard Disk Drive for Models I and III

Forecasting some Radio Shack products isn't all that difficult After all, the RSC-7 catalog included hi-res graphics for the Model II, with exactly the same resolution and Graphics Basic commands as for what was to come later for the Model III (but at $499).

It was only a matter of time before Radio Shack offered such a highly desirable item as hi-res graphics for the Model III. The RSC-6 catalog offered a hard disk system for Models II and 16 with 8.4 megabytes of storage for $4495. Again, it was only a matter of time before a similar hard disk system was available for the Model III; the RSC-8 catalog has it, for the III (and I, with an adapter), at $2495 for 5 megabytes of storage (Figure 2). I had known quite some time before this writing that Radio Shack had ordered hard disk drives for the Model III (no, I didn't get the information from Fort Worth), but there was no point saying anything before now, because Radio Shack has enough problems without people calling up to ask when the hard disk drives are coming. The RSC-8 catalog gives all the details, plus an availability date of 11/15/82.

Incidentally, those prices for the hard disk drives are for the primary drives, which include a hard disk operating system. The secondary drives are less ($3485 for the II, $1995 for the III); up

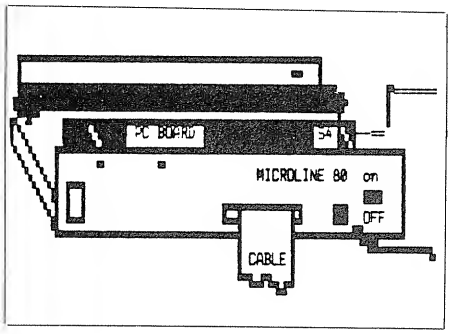


Figure 3. This partial illustration from the CopyArt *manual, showing how to set up an Okidata Microline printer, was created with* CopyArt.

to four hard disk drives can be attached to a Model I, II, III, or 16 TRS-80, for a total of 20 or over 30 megabytes of storage, depending on the model.

Why use a hard disk (also called a Winchester) system? You can eliminate most, if not all, of the tiresome business of having to put one floppy disk after another into your drives (except when it comes to backup disks). This may not be a big deal if you are a hobbyist, but for a business with many records, hard disks save a lot of time, not only because most floppy disk programs and data can be transferred to hard disk, but because they are then available much, much faster. On the Model II floppy disk drive, for example, the transfer rate is 500 kilobits per second. The Model II hard disk transfer rate is 4.34 megabits per second, providing what the catalog calls "extremely fast access to programs and data."

Aluminum Sprayed on Plastic II

The problem of the aluminum paint wearing off the keyboard of your TRS-80 if you rest your hands on it too much was mentioned in June 1982 (p. 217). The question was asked, "Has anybody found a spray-on or brush-on paint that matches the TRS-80's aluminum color?"

Michael B. Rowe, P.E., of Simplified Software Systems, which sells computers, accessories and software out of Hickory, NC, wrote:

"In one of your recent columns, you addressed the problem of the paint being rubbed off the TRS-80. Being in the software business, we use our machines constantly, and as a result we had the same problem. Our solution was arrived at by buying several different brands of aerosol-type aluminum sprays, and using molded-plastic outlet boxes for tests. What we decided on proved to be a satisfactory color match, and probably a better finish than the original. The procedure requires that you follow these steps carefully.

1. If the computer is out of warranty, remove the aluminum shell from the keyboard—or, in the case of the Model III, the whole top—as a unit. *Caution*: Be extremely careful when removing the top from the III to be sure you do not damage the neck of the CRT.

2. Use extra-fine (220 to 600 grit) sandpaper to slightly roughen the surface to be coated. Carefully remove all sanding dust!

3. Apply Krylon, Dull Aluminum 1403, to the area to be coated, in several light coats rather than one heavy coat. It dries quite rapidly, so only 20 to 30 seconds is required between these light coats.

4. After the dull aluminum spray has dried for several minutes—preferably not more than four or five—apply several light coats of K-Mart brand Fast-Drying Spray Enamel in CLEAR U3733. This may result in slightly more surface sheen than the original, but generally blends nicely. On our Model IIIs we did not have to paint the whole top to get a satisfactory appearance.

"I might add that the black plastic keyboard bezel on both the I and III can be removed and recoated if necessary (someone got white paint on one of

ours), using Martin-Senour Vinyl Color Spray #7977, Jet Black. Note that is Jet black, not Gloss black.

Before trying this on your TRS-80, you might want to work up a good spraying technique (slow and steady) on some scrap plastic or whatever. If you try the above solution, or have others, please let us know.

CopyArt Word/Graphics Processor

According to the ads, *CopyArt* provides "the new dimension in word processing," which is graphics; you can put words and pictures on the same screen or page (Figure 3). That is, if you have a Model I or III TRS-80, 48K of memory, and at least one disk. And *CopyArt*, which is $149.95 from Simutek Computer Products Inc. (4877 E. Speedway, Tucson, AZ 85712).

CopyArt combines a word processor very much like Radio Shack's *Scripsit*, a graphics mode for using the cursor like a pen, drawing lines with the four directional arrows (similar to Etch-A-Sketch); a second graphics mode that creates large-size letters and numbers; creates large-size letters and numbers; and extensive printout capabilities. *CopyArt* was written with a special version of Simutek's ZBasic compiler, which is advertised as "the world's fastest TRS-80 Basic compiler," and which will be reviewed here at a later date.

CopyArt Word Processing

If you have used *Scripsit*, *CopyArt's* word processing capabilities will be very familiar. Some are just about the same, many are new or improved, and only a couple are not quite as good as their *Scripsit* counterparts.

As in *Scripsit*, you use CN=Y for centering, JU=Y for justifying text, LM=5 for left margin at 5, RM=65 for right margin at 65, HD=2 for using the next two lines as headings, PG=10 for starting page numbers on page 10, etc. As in *Scripsit*, you can scroll text (roll the screen up or down the text), move to the top or bottom of text (or to the left or right side of text) using combinations of arrow keys and the shift key, and do global search and replace.

Moving text around is simpler: you just put any text to be moved into a buffer, move the cursor to a new position, and unload the buffer, which places the text in the new position. With the down-arrow, you can move down a screenful at a time, instead of just a line at a time.

To emphasize text, you can simply put SE=Y in the format line (for single emphasis), which double-prints a line and thus makes it darker, or use DE=Y (for double emphasis), which will quadruple-print a line and make it extra dark. To underline text, simply put a ± at the beginning and end of phrases you want underlined.

Two of the best *CopyArt* word processing features are for deleting or inserting characters, and operate more as they do in commercial word processors. Press D, and everything to the right of the cursor moves to the left and disappears, as though swallowed up by the cursor. Press I, and blank spaces pour from the right side of the cursor.

A Directory feature calls up a menu allowing you to get a listing of all the various files on as many as four disk drives, or find out how much space you have left on a diskette.

Hitting the H key calls up help: four pages of display listing the various commands for word processing and printer format, plus some tips. As the manual puts it, "To help you remember the various commands we've included a reference cards." On the screen, that is. Fine, but I would still like a separate card, rather than having to call Help every time I need a simple assist. For me, the lack of a reference card is the only serious problem with *CopyArt*. The index to the 104-page manual is quite good, but using a manual to look up commands takes too much time.

The only *CopyArt* commands that seem to work better in *Scripsit* are two of the simplest: cursor movements left and right, which don't operate as smoothly. Generally speaking, *Scripsit* seems more professional and tidier in operation than *CopyArt*.

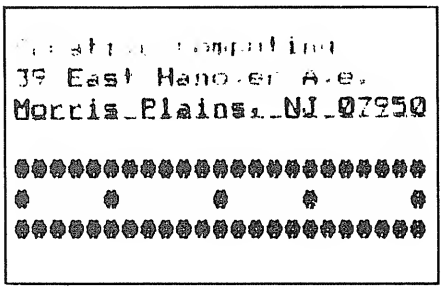


Figure 4. CopyArt *can print with varying degrees of emphasis, underline, and can approximate graphics blocks with 0 over #.*

CopyArt Line Graphics

Hit the break key and then G, and you are into graphics mode. Now you can use the cursor like a pen (or an eraser), drawing lines (or deleting them). Hit D, and as you move the cursor with the arrow keys, it leaves a trail. Use E to get into erase mode. By switching back and forth from D to E, you create graphics. To move the cursor around without drawing or erasing anything, use M. To speed up the cursor, hit F. That's all there is to the draw-a-line graphics, but you can do a great deal with these four commands.

CopyArt Character Graphics

Hit @ and Y, and you can create big graphic characters automatically, for

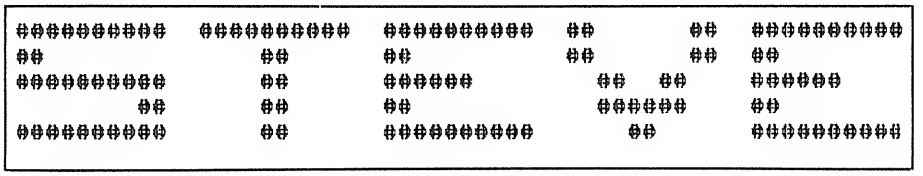


Figure 5. Using pseudo-graphics, CopyArt *can print a good approximation of solid letters.*

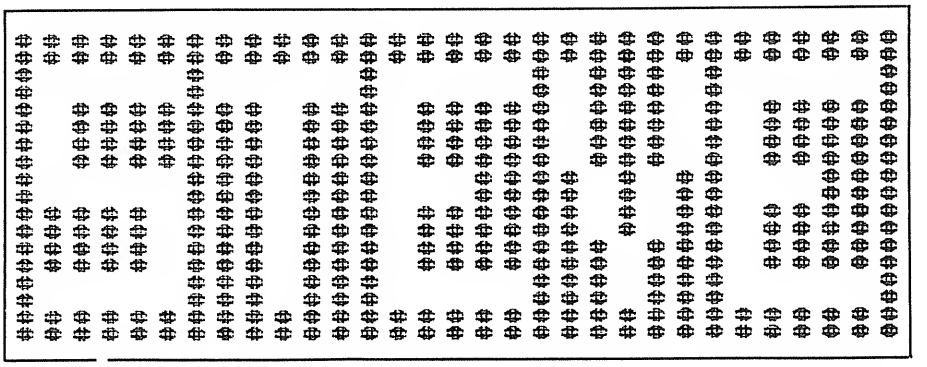


Figure 6. These inverted characters were originally printed vertically with CopyArt.

headlines or banners or whatever you need. The *CopyArt* disk includes a basic set of characters, including the alphabet and numerals, which you can display and print out in various sizes.

After you hit Y, you type in a letter, word or phrase at the bottom of the screen. You specify the height and width of the characters, whether you want them printed horizontally or vertically, and in black on white or inverted (white on black).

These characters can be printed out quite large, especially vertically, where you can print fairly long banners. If you have a printer—such as the Okidata Microline or Epson—that supports the TRS-80 graphics character set, then your banners will be printed with letters of solid black.

On other printers, *CopyArt* uses "pseudo-graphics," by printing, instead of a solid black rectangle (which is equivalent to a white rectangle on the screen), a # overprinted by a 0.

Although at this moment I have three printers on hand, not one of them can print a graphics block. So I had to use pseudo-graphics to see what *CopyArt* can do. In the first example (Figure 4), I put a row of pigeonholes below *Creative's* address, with the first line printed normally, the second with single emphasis (printed twice), and the third using double emphasis (printed four times) and underlined. Although the four boxes were exactly the same size in the original, the printout is a pseudo-graphics approximation, with two sizes of boxes.

The second example (Figure 5) shows my first name printed horizontally with small pseudo-graphic characters. The third example (Figure 6), shown horizontally, was originally printed vertically and inversely.

Actually, in that last example, three sets of the name were printed vertically. You can put a word on the screen, then move it over to the right using the insert command to insert blanks, then put another word on the screen, push that into the middle, and add a third word alongside. Incidentally, as the manual notes, the *CopyArt* graphic characters are meant to be used as a framework; you can use the graphics mode to change the basic letters into almost any style you want. "For example, once you have the characters on the screen, you might patch them so they look a little more like Old English type.

Advanced CopyArt

The imaginative manual has a section on some fancy features, such as killing the linefeed for creating new characters with overstrikes, turning off printing so as to print only a portion of a document, using *CopyArt* as an editor, and, if your printer uses special control codes, inserting them directly into text.

When you want to use a control code, you hit @ and C. A message appears at the bottom of the screen:

Cntrl codes?

At which point you simply enter the code. The Cetnronics 737, for example, will start underlining if it is given control code 14. So you type 14, and hit the enter key. Depending on the printer and the code, a character (sometimes a rather strange one not in the regular set) may or may not appear on the screen. It won't be printed out, but will affect the text following it.

That's most of the *CopyArt* features, which are many and varied, and which you can take days or weeks exploring before you find out all the program can do. (It can simulate a page up to 255 characters wide, for one thing.) If the price seems high, consider that Radio Shack's *Scripsit* alone is $99.95; for an extra $50 you get a word processor with many features not found in *Scripsit*, plus a graphics generator with a great many fascinating capabilities. If you want to combine words and pictures, *CopyArt* provides what may well be the only way to do it directly.

Short Program : Calendar

Max Seim of Stillwater, MN wrote to say he "recently picked up the latest issue of *Omni* magazine and quickly flipped to my favorite section: Games."

"I was very interested in the unique mathematical way they showed to determine the day of any given date. I decided to use this method to produce what I think is the shortest Basic program ever written that does it.

```
10 CLS: DIM N(12)
13 FOR X=1 TO 12: READ N(X): NEXT
15 FOR X=0 TO 6: READ D$(X): NEXT
20 INPUT "ENTER MONTH,DAY,YEAR (MM,DD,YY) ";M,D,Y
25 Y1=Y
27 IF Y/4=INT(Y/4) THEN 29 ELSE Y=Y-1: GOTO 27
29 Y=Y/4
30 T=Y+Y1+N(M)+D
32 T1=FIX(T/7): T2=T-(T1*7): IF Y1=00 THEN 50
40 IF Y1/4=INT(Y1/4) AND M=1 THEN T2=T2-1 ELSE
   IF Y1/4=INT(Y1/4) AND M=2 THEN T2=T2-1
50 REM CHANGE THIS LINE FOR 1800'S AND 2000'S
60 PRINT "THE DAY IS: "; D$(T2);".": PRINT
70 T=0: GOTO 20
100 DATA 1,4,4,0,2,5,0,3,6,1,4,6
110 DATA SAT,SUN,MON,TUE,WED,THU,FRI
```

"*Note:* The data are important and must be correct. These twelve numbers are the key numbers for each month used in the formula in lines 29-32. That's the complete listing . . .all nine lines of it!"

The program as it sits will determine the day during the 1900's only. Line 50 must be changed for the 1800's and 2000's:

For the 1800's: 50 T2=T2+2
For the 2000's: 50 T2=T2-1

"The program could be easily modified to make these adjustments automatically. Simply input the year as four digits instead of two."

"1900 and 1800, although divisible by four, are not leap years. The program allows for this in the last part of line 32. Years such as 2000 and 4000 are leap years because they are divisible by 400. If you wish to input the year 2000, make sure the last part of line 32 is removed. The last part of line 32 simply jumps over the leap-year determiner if you input 00 for the year.

"If you have any questions, see pages 152-3 of the November 1981 *Omni* magazine."

Max's original program (which of course he submitted some time ago) did contain only nine lines, but I stretched it out a little to make it more readable and to fit this column.

The calendar item Max refers to in *Omni* is called "You too can be an idiot savant," and tells how, by memorizing a method and certain numbers, including those in DATA line 100, you can amaze people by giving the day for any particular date, with just a few moments of mental calculation. The method is fairly simple, but you'd have to use it quite often to be able to remember it for any length of time. □

Pt-210 Portable Data Terminals • A Secret Program and Password

February, 1983

New Product Preview

Last September, New York's St. Regis Hotel was the scene of the annual Tandy Corp./Radio Shack New Product Preview. This preview included a roomful of new products that appear in the 1983 Radio Shack general merchandise and computer products catalogs, displayed for examination by security analysts and the press.

In addition to new stereo equipment, modular telephones, auto anti-theft alarms and such, the display included high-resolution graphics for the TRS-80 Model I and III, several new printers, Winchester hard-disk drives, the Portable Data Terminal, and several software products. An adjoining room offered the opportunity to play a variety of Color Computer games, to try out Logo or watch it in operation, and to examine a prototype videodisc/computer system.

Portable Data Terminal

The PT-210 Portable Data Terminal, priced at $995, is Radio Shack's first venture into the portable printing terminal market. It includes a full QWERTY keyboard, a very quiet 80-column dot-matrix thermal printer, and an acoustic telephone coupler.

The terminal, as a Radio Shack executive put it, is "an approach to the Teletype market as a low-cost alternative to what's available. Radio Shack stores are everywhere, so it's easy to get the terminal fixed when necessary."

You can use the PT-210 at your office to communicate with your TRS-80 at home, if your computer is turned on, and is equipped with communications software and an auto-answer modem such as the Modem II. The terminal can be used for timesharing, and by at-home programmers.

Using the PT-210, you can communicate over a telephone line with just about any computer equipped with a modem, from a TRS-80 to a mainframe. Add the $69.95 RS-232C interface and you don't need the telephone coupler any more; just wire the terminal to a direct-connect modem such as Radio Shack's $149 Modem I or $249 Modem II.

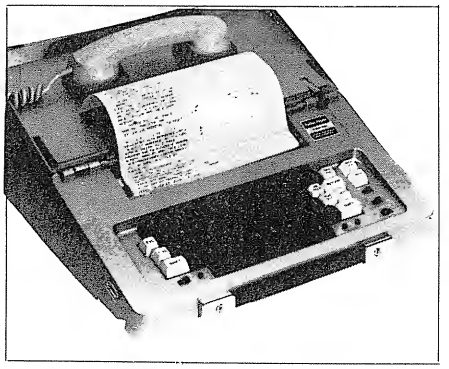


Figure 1. The PT-210 Portable Data Terminal weighs 15 pounds, prints quietly, and communicates with almost any modem-equipped computer.

At the preview, the PT-210 was connected to a mainframe computer in California, via a data-formatting accessory.

Videodisc and Computer

One of the most fascinating preview displays was a prototype combination of a videodisc player and a TRS-80 Model III computer, engineered at the University of Nebraska and envisioned for the industrial training market. A special circuit board mounted in the Model III permits the computer CRT to show pictures from the videodisc player. Thus the screen can display either television pictures or computer characters.

Although the prototype was programmed to run by itself, to demonstrate continuously the basic principles by switching back and forth from TV pictures to computer displays, it could have been programmed for interactive use. For example, a picture of a relay, with its contacts labelled, could be shown from the videodisc, and the computer could then ask a multiple-choice question on how to wire the relay for a particular purpose. Then, depending on the response, the computer could call up videodisc frames to either tell the user he was right, or show him exactly why he was wrong.

But don't count on the videodisc/computer combination becoming a product; it's still in prototype for evaluation.

Covering Up Program Lines

Dennis Tanner, Radio Shack's Manager of Educational Products Development, demonstrated Logo for me. He also showed me how to cover up program lines with meaningless words, so that a secret program can be concealed. First you write a program line, such as

```
10 PRINT ''HELLO''
```

and after you ENTER it, you call it up with

```
EDIT 10
```

and then hit X to move the cursor to the end of the line. Hit SHIFT 7 to write a single quote (also called apostrophe), which is the abbreviation for REM.

Now comes the basic secret. While holding down the shift key, press the left-arrow key, which backs up the cursor—without erasing any characters—until it gets to the beginning of the line. Then you can enter any phase or group of characters you wish, such as

```
NOW IS THE TIME
```

by writing it over the old line, followed by hitting the ENTER key.

If you do a RUN, you don't get

```
SYNTAX ERROR IN 10
```

as might be expected, but rather

```
HELLO
```

which is what the original line called for.

If you LIST line 10, the part before the apostrophe will show on the screen, but so briefly you won't be able to read it. However, if you do an

```
EDIT 10
```

and use the spacebar to look at the line letter by letter, you'll see the cursor move to the right as it traces the original line, then move left to the beginning of the line and, moving right once more, trace the cover-up words. What you have created, of course, is a line that includes a REM statement that includes several backspaces. When you LIST it, the line folds over and conceals the characters you placed to the left of the apostrophe.

You can use

```
:REM
```

instead of the apostrophe, but that takes more space and time. When backspacing the cursor, the SHIFT key must be held down while the back-arrow key is being operated, otherwise the line will be erased. And you can't simply write a line, add an apostrophe, and back up the cursor with the SHIFT/left-arrow combination; this will erase the line. You must write the line first, and then go into edit mode.

Note that you can cover the original program entirely with text, or you can use the original line numbers and make the program look entirely different, using a cover-up program that would work, or one that wouldn't. All sorts of fiendish things can be done with this idea, such as altering just one line in a friend's program.

Model III For Store Inventory

While at the Radio Shack preview, I spoke with Jon Shirley, vice president of Radio Shack Computer Merchandising, and author of the "View From the Seventh Floor" column in the "TRS-80 Microcomputer News."

He told me that all 4400 Radio Shack stores in the continental U.S. and Puerto Rico use TRS-80 Model IIIs every night to call orders in to headquarters in Fort Worth, and to transmit daily reports and payroll information. The Computer Centers are on the same system, each using a Model II. The information is processed in Fort Worth by a Tandem "NonStop" computer system, which uses a pair of processors arranged so that one will continue to operate if the other should fail.

Orders can be transmitted faster to warehouses now, eliminating several costs, as well as keypunch errors, and improving inventory turnover. Program updates can be quickly sent to stores, as well as inventory information. An electronic-mail system will eventually be installed for transmitting individual messages to stores.

Radio Shack will change, and perhaps by now already has, to an auto-dial system with automatic polling, so nobody has to be at the stores to transmit data to Fort Worth, as long as the Model III or II is left on.

Shirley has a Model II in his office, as do many other people in Fort Worth management. Radio Shack is working toward networking these machines via Datapoint's ARCnet.

String Packing Using Debug

Back in the December 1980 column (p. 194), we discussed how to create graphics using string packing, which was invented by Leo Christopherson, the ace graphics artist and game programmer who was signed up by Radio Shack last year (Dec. 1982, p. 408).

Leo's string packing assigns characters in the string. Then you look into memory, with the variable pointer VARPTR, to find out where the computer stored the string, and you just POKE the graphics codes into the string.

Dennis Tanner is quite a programmer himself, having been a Radio Shack programmer for a year after leaving teaching, and before moving into the Educational Division. He invented a new way to pack strings, using *Debug* in Model III TRSDOS, which eliminates the use of VARPTR, because it is not modifying the program in memory, but modifying the program on disk. Dennis says he taught this method to Leo Christopherson, and that it was due to appear in Radio Shack's "TRS-80 Microcomputer News," where you may have seen it by now.

First you write a program in Basic, such as

```
10 A$="
20 PRINT A$
30 PRINT LEN (A$)
```

then save it with

```
SAVE ''DENNIS'', A
```

which puts it in ASCII format.

Return to TRSDOS by using

```
CMD''S''
```

(you can leave out the second pair of quotes) and then type

```
DEBUG
```

followed by

```
F
```

which calls up the File Patch utility that allows you to modify the contents of a diskette file, and which will cause the *Debug* program to respond with the prompt

```
FILESPEC?
```

So you enter the name of the file to be patched

```
DENNIS
```

and *Debug* sets up a full-screen display that shows the first 256 bytes in the DENNIS file. The top line will display at left, in part, 3130 2041 243D 2220 2020 2022 and display at top right the ASCII equivalent

```
10 A$="      "
```

The 20 is a space, in hex code, and there are four of them, one for each blank space between the quotes in line 10.

Next you press

```
M
```

to get into Modify-Memory mode. Now you can pack the string with whatever you like, by simply writing new hex values over the old. Use four

```
BF
```

for instance, and then

```
ENTER
```

to keep all the changes made (by updating the diskette file) and to get out of the Modify-Memory mode. Press the BREAK key twice to return to TRSDOS. To see the effect of packing the string with BFBF, do a

```
RUN ''DENNIS''
```

and the display will show

instead of only the 4 that was displayed when the original program was run. BF is the hex-code equivalent of decimal 191, which is the graphics character consisting of all six blocks. Note that you could also have changed the line numbers, to 100-110-120, or whatever you wish.

That's how easy it is, with string-packing using *Debug* on a TRS-80 Model III. One big advantage of using *Debug* is that, as you write new hex values into the blank spaces between the quote marks, you can immediately see what the equivalent graphics characters will look like, over at the right side of the screen, where the ASCII translation is shown. Just remember that you have to store the original program in ASCII format.

Color Upgrades Cost Less

Now that the 16K Color Computer is available for $399.95, or 95 cents more than the 4K Color Computer previously was, the cost of upgrading your 4K Color Computer to 16K has been reduced from $99 to $49. The Extended Color Basic ROM Kit (which requires a minimum of 16K) is still the same $99, but both the added memory and the Extended Basic can be installed for a $30 labor charge.

If you are into color graphics, the additional features are well worth the money. You get high-resolution graphics (up to 256 by 192 pixels); automated drawing of lines, rectangles and circles; a DRAW mode for simplified sketching; as well as additional functions such as SOR, EXP, LOG and more.

One of the few problems is the manual. The 4K manual, "Getting Started with Color Basic," seems to have been written for children, which may be better than if Radio Shack had tried to write it for adults, because anybody ten or older can understand this manual. Except for most of the section on string operators, that is. And the manual makes no mention at all of about 20 functions, such as LIST, ABS, and SIN.

As elementary and as frustrating as the 4K manual is, it's better than the "Going Ahead with Extended Color Basic" manual. In some places the Extended manual just barely mentions a feature, and in others makes another feature sound highly complicated.

For example, the manual is too complex when trying to explain why you are limited to displaying two colors in some circumstances, or why you need to reserve pages, or why PCLEAR or PCOPY is needed. On the other hand, the manual keeps mentioning all sorts of goodies, but doesn't give much detail beyond one simple illustration each. Several programs cry out for an explanation, but none is given for most of them. There is great power in Extended Basic graphics, but the Radio Shack manual only begins to hint at it.

Non-Radio Shack Color Books

As noted before, Radio Shack's delinquency in generating decent manuals for all too many TRS-80 products provides work and profit for freelance writers. Just look at page 41 of the RSC-8 catalog, which has at least half a dozen books written by outsiders—books of the caliber the Radio Shack manuals should be.

```
10 CLS
20 I$=INKEY$: IF I$="" THEN 20
30 IF I$=CHR$(8) AND A$<>"" THEN PRINT CHR$(8);:
   A$=LEFT$(A$,LEN(A$)-1): GOTO 20
50 IF I$=CHR$(13) PRINT A$: END
60 IF ASC(I$)<32 OR ASC(I$)>127 THEN 20
70 A$=A$+I$
80 PRINT "*";: GOTO 20
90 GOTO 20
```

Listing 1.

A very good book published recently by John Wiley & Sons is "TRS-80 Color Basic," by Bob Albrecht. However, it doesn't get into Extended Color Basic, because the author knew his friend Don Inman was writing a book on the subject, which Reston published and which will be reviewed here. Meanwhile, the Albrecht book is highly recommended for its fine, extensive coverage of Color Basic; look for it in the Book Review section of a future issue.

Short Program

Sometimes you don't want to display certain numbers or letters when you use the keyboard. For example, when you are logging on to a timesharing system, it will usually ask for your password. You don't want somebody standing near your terminal to see your password and perhaps use it later, so the system usually doesn't print it. Or if you use a sidewalk banking terminal (known as an ATM, or "automatic teller machine," in the trade), you don't want people in line behind you to see your " personal security number" as you key it in. So the terminal may display asterisks instead of your number.

How can you make your TRS-80 do the same thing? Dennis Tanner had written the "cover-up" program so quickly, I asked if he could write another that would print asterisks instead of whatever is being entered from the keyboard. Within a few minutes he wrote a program that worked perfectly, and which would have takenme much longer to write. It appears in Listing1 with only a smidgen deleted to simplify it.

The use of INKEY$ in line 20 lets you build a string of characters in line 70. These characters are limited, by line 60, to those whose ASCII values are between 32 and 127, but line 80 prints an asterisk for each key you press. When you hit the ENTER key, line 50 senses the carriage return (ASCII code 13), and prints the string of characters right after the row of asterisks that represents those character.

What does line 30 do? It lets you backspace to correct any mistakes you may make in entering your secret number. ASCII code 8 "backspaces and erases current character," according to the Level-II manual. Can you figure out how the LEFT$ and LEN string operators work here?

You can change line 80 to display any character you wish in place of the asterisk. By changing the numbers in line 60, you can restrict the "secret number" to letters only, or numbers only, etc. You can turn this short program into a subroutine, perhaps as part of an ATM program for a banking simulation, and use the secret number within the program without ever displaying it.

Why can't line 30 be split in two? Rewrite it in two parts, using lines 30 and 40, and try to figure out what happens when you try to RUN this new version. □

Spectrums • Spirals • Tandyvision

March, 1983

TRS-80 Computer Scultpure

For $9.95 you can buy, postpaid, a paperweight and/or conversation piece that is, according to the press release, "a contemporary caricature of the popular TRS-80 and a happy user" (Figure 1).

This sculpture of a Model III and a computernik with no neck and not much torso (for $9.95 you were expecting Michaelangelo?) is an "antique gold-finished plaster statue . . . a hefty $4^1/_2$ inches long." Ten or more are $7.95 each, from Brian Productions, 2949 Southfield Rd., Xenia, OH 45385.

Brian told me the sculpture is plaster of paris (so don't drop it), which he has cast commercially in quantity and then sprays black and gold himself. He's been selling such sculptures for ten years, starting with an NCR Century. He also sells similar sculptures of the Apple, IBM System/3, IBM 370, a plain unnamed computer (which looks somewhat like an old IBM 360), and another labelled, "I hate my computer," which resembles nothing at all.

Figure 1. One of the half-dozen computer sculptures offered by Brian Productions resembles the Radio Shack TRS-80 Model III.

Audio Spectrum Analyzer

Although Radio Shack's $19.95 plug-in *Audio Spectrum Analyzer* Program Pak for the Color Computer is described in the catalog as "the perfect way to test your stereo equipment for maximum performance," it's quite entertaining just to watch the vertical bars dancing like a colorful fountain in response to the music.

First you connect your stereo amplifier to the Color Computer by patching the headphone output of the amplifier to the black plug on the cassette recorder that normally goes into the EAR jack, using a connector that has a quarter-inch stereo jack on one end and an eighth-inch miniplug on the other. With the other end of the cable plugged into the computer, and the program ROM cartridge inserted in the right-side slot, all you do is turn on the stereo and the computer, and soon you see a very colorful display (Figure 2) of 27 vertical bars that represent one-third-octave segments of the nine-octave range from 31.5 Hz to 12,500 Hz.

The vertical color bars are calibrated in dB, which is a relative measurement; the program locks onto the loudest one-third-octave present and scales its measurements from that reference.

As you watch, you learn about the characteristics of various types of music, and before long you might even be able to identify with the sound turned off classical music, rock, and jazz. Press the K key, and you get an "audio kaleidoscope" that produces changing visual images that depend on the distribution and intensity of the audio signal. At first the pattern seems truly random, but after a while, you learn to differentiate between various types of music. Even noise has its own pattern.

The color bars that represent the instantaneous energy distribution display the real-time peak response. Press S to get the averaging Slow RMS response, for measuring the effect of speaker placement or tone adjustments.

Measuring power is difficult, because the peaks pass quickly. You can lock onto the highest peak level reached in each band by pressing P to show the Peak levels in the signal. In Fast mode, you see which frequencies demand the most power; in Slow mode, the average distribution of the signal and the general relative response of the system under test.

To freeze the display for analyzing a musical passage or an instant in time, just press the space bar.

Measuring the Electronics

Put a familiar record on your turntable, set the tone-altering controls to flat response, set *Spectrum Analyzer* to Slow response with the Peak display turned off, and play the entire selection. Adjust the amplifier volume to show maximum activity on the screen.

Then replay, in Peak Hold mode. At the end of the selection, the Peak indicators show the maximum average energy reached in each band. You can now play the record again and use your tone controls or graphic equalizer to change the sound, and see their effects on the music by comparing the new peaks to the original peak indicators.

Measuring the Audio Chain

By connecting *Spectrum Analyzer* to a low-powered amplifier driven by a microphone, you can measure room and speaker response. Try the microphone in various parts of the room, and use your tone controls and/or graphic equalizer to try to match the response achieved in the flat-response test from the headphone output jack.

TRS-80 Graphics—With Disk

One of the best books on TRS-80 graphics is called just that: *TRS-80 Graphics: For the Model I and Model III,*

by David A. Kater and Susan J. Thomas. Published by Byte/McGraw-Hill at $12.95, it is also available in a Radio Shack cover for $10.95 at Radio Shack stores.

The book covers all the basics such as SET and RESET, strings, PEEK and POKE, and machine language graphics and sound, then gets into geometric shapes and function plots, statistics, graphs, games, kaleidoscopes, plotting art, etc. Animation is explained by using an eye-blinking and antenna-twitching beast called Critter. (If several of the graphics examples look slightly familiar to those of who have been reading this column for a while, it is because *Creative* is one of the places the authors looked for ideas, and then improved upon what they found.)

On the last page in the book, the authors (whose address is Box 1868, La Mesa, CA 92041) offer a cassette or disk containing 35 of the major programs, for $14.95. This is highly recommended if you hate to key in programs.

The 35 recorded programs are meant to be used in conjunction with the book, mainly to save you time in typing and correcting program errors as you go through the text, and thus "speed up your learning." The authors want you to run a program, think about what you see, read the line-by-line analysis, then go over the program to make sure you understand what makes it work.

Most of the programs display graphics that don't do anything other than show a graph, plot, sample menu, dragon, or lion's head, which is all the text calls for. But several do something other than just show a static display. One is Sketch; you use the arrow keys to draw whatever you like, in white on black or vice-versa, with counters keeping track of where the cursor is. Inkblot automatically draws a randomly-generated four-way kaleidoscopic pattern, by using the arrow keys (see the Short Program at the end of this column).

Spirals

Another active program is Spirals, from the section on circular graphics, using polar coordinates to create one new spiral after another, by increasing the angle of rotation for each new display.

You enter D, the "angle between points in degrees" (line 10) and indicate whether the spiral is to be drawn as a continuous line or as points (line 20). The spiral is begun at the center of the display by the SET in line 40.

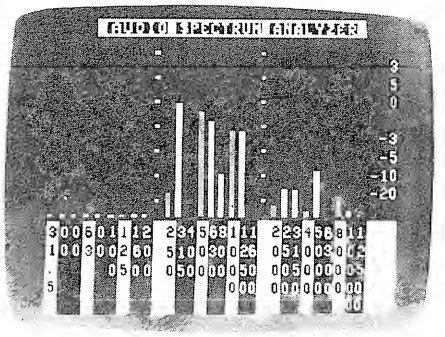


Figure 2. The energy distribution of music or speech is displayed as vertical color bars a third of an octave part.

The equation for the spiral is in line 70; additives 64.5 and 24.5 keep the spiral centered in the display; the .43 factor compensates for the 3:7 aspect ratio of the rectangular graphics block (how?). When the spiral reaches the edge of the display area, line 80 stops the drawing, jumps the program to a time delay in line 210, and after several seconds the next spiral is drawn, with D augmented by 1, which increases the angle between the points and thus draws a different spiral, some of which look rather odd.

Note that if the spiral is drawn as points, line 90 skips over half the program. Can you figure out how lines 100-190 are used to draw the spiral as made up of line segments?

For ordinary spirals, try a value for D such as 3, using P or L. For something quite different, use angles such as 88 or 122 degrees, first with lines, then with points. Notice the big difference in the appearance of spiral displays when using lines or points, when the angle is between 60 and 90. Why is this? Note how the spiral smooths out, using lines, when the angle increases from 315 toward 340 degrees.

To remove the delay between finishing one spiral and starting the next, replace the delay in line 210 with REM (why not just delete the line?). To increase the speed of generating spirals, so you can more quickly see the effect of using various angles and either lines or points, reduce the 200 in line 60 to 25 or even 20 (after writing 210 REM).

When the program reaches line 220, it increases the value of D by one and jumps to line 30 to draw the next spiral automatically. But if you want to change the variables, press the spacebar, and when the current spiral is finished, the program jumps to line 10.

Note the difference between the "spiral" drawn with lines when D is 90 and when D is 90+360, or 450. Why are these spirals different?

Tandyvision One

Radio Shack's new $249.95 computer-based electronic video game, Tandyvision One, is a custom-labelled Mattel Intellivision. According to the press release, "it can run all Mattel Intellivision and Sears Super Video Arcade cartridges." The game comes with *Las Vegas Poker* and *Blackjack*; 15 Intellivision cartridges are available through Radio Shack stores and dealers, ranging from $17.95 for *Checkers* to $34.95 for *Sea Battle* or *Star Strike.* Additional Intellivision game cartridges are available through Radio Shack's spe-

```
10 CLS: INPUT "ANGLE BETWEEN POINTS IN DEGREES"; D
20 INPUT "POINTS (P) OR LINES (L)"; S$
30 CLS: X1=64: Y1=24
40 PRINT D "DEGREES": E=D/57.296: SET(64,24)
50 IF S$="P" THEN PRINT "POINTS" ELSE PRINT "LINES"
60 FOR A=0 TO 200 STEP E
70 X=A*COS(A)+64.5: Y=A*SIN(A)*.43+24.5
80 IF X<0 OR X>127 OR Y<0 OR Y>47 THEN 210
90 IF S$="P" SET(X,Y): GOTO 200
100 IF X1<>X THEN 130
110 IF Y1<Y THEN S=1 ELSE S=-1
120 FOR Z=Y1 TO Y STEP S: SET(X1,Z): NEXT Z: GOTO 200
130 M=(Y-Y1)/(X-X1)
140 IF ABS(M)>1 S=ABS(1/M) ELSE S=1
150 IF X1>X THEN S=-S
160 FOR Z=X1 TO X STEP S
170 SET(Z,M*(Z-X1)+Y1)
180 NEXT Z
190 X1=X: Y1=Y
200 NEXT A
210 FOR I=1 TO 3000: NEXT
220 IF INKEY$=" " THEN 10 ELSE D=D+1: GOTO 30
```

Listing 1. Spirals.

cial ordering service, the "Intellivision Cartridge Hotline."

The 16-bit, microprocessor in Tandyvision One controls sound effects, music, color and high-resolution animated graphics.

Intellivision has been available at Macy's in New York for $229.95 less a $50 rebate, for a "final cost" of $179.95.

Color-Computer Hi Res

Arnold Kahn writes from Chevy Chase, MD:

"I have followed your column with particular interest in matters relating to the TRS-80 Color Computer. I noted your comments on high resolution on the 4K machine, in the March 1982 issue." (Page 202 includes a program that accesses the hi-res mode without Extended Basic, and asks if anybody had figured out how to access the mode fully in the same way.)

"A careful reading of the Radio Shack introductory manual shows that a full screen at highest resolution requires 6K of RAM for the display alone. Since the 4K machine doesn't have it, there is no way to put up a full screen of high-resolution material. Also, because of the VDG-SAM (video display generator-synchronous address multiplexer) architecture, it is not possible to divide the screen into a graphic part and an alphanumeric part, the way some machines allow.

"The best route for the 4K owner is to upgrade the machine to 16K with his own two hands. It is easy. The cost these days is about $16 for eight 4116 200-nsec memory chips to replace his eight 4027 chips. He'll need to see the Radio Shack Technical Manual for the Color Computer (which he should have anyway) or compare with a 16K machine. In addition to replacing the socketed chips, the owner must move two jumper plugs that are clearly marked. No soldering is required. The machine will then have 16K memory, and will be the best bargain-priced computer in the world. (It will not have Extended Basic, and the warranty will be dead.)

"Upgrading to 32K by piggybacking chips is possible, but it takes soldering. A 32K upgrade without soldering is in the form of a plugboard from Computerware.

"Extended Basic comes in a plug-in ROM, but costs about $100, if you can get it. It is worth having for the editing capability alone, but then the cost is almost up to that of buying the machine fully equipped." The Computerware memory board for expanding from 16K to 32K, which I haven't seen, is $79.95 plus shipping, from Box 668, Encinitas, CA 92924. They have a catalog of products for the Color Computer.

```
10 CLS: DEFINT A-Z
20 X=0: Y=0
30 P=PEEK(14400)
40 I=P AND 96: J=P AND 24
50 H=SGN((I-63)*I)
55 K=SGN((J-15)*J)
60 IF X+H<0 OR X+H>127 THEN 30
70 IF Y+K<0 OR Y+K>47 THEN 30
80 SET(X,Y): X=X+H: Y=Y+K
90 SET(X,Y): SET(127-X,Y)
100 SET(X,47-Y): SET(127-X,47-Y)
105 IF P=1 I$=INKEY$: GOTO 130
110 RESET(X,Y)
120 GOTO 30
130 I$=INKEY$
140 IF I$="" THEN 130 ELSE 10
```

Listing 2.

Short Program #37: Four-Way Doodle

Going back to the *TRS-80 Graphics* book, let's look at the doodle program in Listing 2.

You use the four arrow keys to move a blinking cursor left or right, up or down, and as you do so, the pattern you create is duplicated, in mirror fashion, in the other three quadrants, so the total effect is that of a four-way kaleidoscope. The pattern begins in the four corners, as initiated by line 20. Because the cursor can move anywhere on the screen, it may cross over into another quadrant. But you have only to look for the blinking cursor to locate what might be called the "master quadrant," the quadrant to concentrate on.

Press two arrow keys (such as the up and right arrows) simultaneously, and the pattern line will be drawn at an angle. When you're through drawing, press ENTER, and the cursor will stop flashing.

Line 30 tests location 14400 to see if it contains a value other than zero. The Radio Shack manuals are rather vague on what values will be stored in location 14400 when the arrow keys are pressed. You can find out easily enough by using this short program:

```
10 PRINT PEEK (14400);
20 GOTO 10
```

and while it is running, press any of the four arrow keys. You will soon see that the arrows put these values in location 14400: up, 8; down, 16; left, 32; right, 64.

Lines 40 and 50 "show a compact way to determine which of the arrow keys have been pressed," the book says. "The sign function is used in line 50 to determine an increment of -1, 0, or +1 for X and Y. P AND 96 is stored in I in line 40. The AND is used to mask out everything except 32 and 64 which represent the left and right arrows, respectively. If I is zero, neither the left nor the right arrow key was pressed, so H becomes zero. If I is 64, H will be +1, and if I is 32, H will be -1. K is calculated similarly. P AND 24 is stored in J. The AND tests for 8 and 16, which are returned by the up and down arrows. Then K is determined by -1, 0 or +1."

If that is not clear, it is partly because some is developed previous to that point in the book, and partly because the authors seem to take it for granted you will understand the Boolean algebra in line 40. Yet the Radio Shack manuals are very skimpy on Boolean: the Model III Basic manual provides one chart that is almost meaningless in its brevity; the Level II manual has almost the same chart, plus ten examples that are more mystifying than helpful because they are not explained thoroughly enough.

If the left arrow has been pressed, P will be 32. If we AND this value of P together with 96, we get

```
 96 = 1100000
 32 =  100000
      -------
      100000 = 32
```

If the right arrow has been pressed, then P will be 64, and if we AND this together with the same 96, we get

```
 96 = 1100000
 64 = 1000000
      -------
      1000000 = 64
```

Both examples show that if you AND a 1 with a 1, you get a 1, but if you AND a 1 or a 0 with a 0, you get a 0. Because 96 is the sum of 32 and 64, a single AND expression using 96 can test for both values to determine if the left or right arrow has been pressed.

If either has, then I is 32 or 64 (line 40) and H is -1 or +1 (line 50). If no arrow key has been pressed, then P is zero, I is zero, and H is zero.

The same logic operates for the up and down arrows, in the second halves of lines 40 and 50, to decide whether K will be -1, O, or +1.

If adding the values of H or K to the X-Y position of the cursor means the cursor will touch the boundaries of the graphics area (X between 0 and 127, Y between 0 and 47, inclusive), line 60 or 70 causes the program to jump back to line 30 to wait for you to press an arrow key that moves the cursor away from (or along) the boundary. Otherwise the values of H and K will be added to the X-Y position of the cursor, and lines 80-100 will add pixels in each quadrant in accord with which arrow keys are pressed.

When you are "done with your cre-

ation, press ENTER. The cursor will stop flashing so that you can admire your work. Then, any key will clear the screen and start the program again," the book says.

When you press ENTER, P becomes a 1, as you will find if you run the two-line program and press ENTER. Line 105 will then cause the program to jump to line 130, skipping over the RESET in line 110 that causes the cursor to blink, thus stopping the blinking and putting a hold on everything until you press any key, which causes a jump to line 10, to clear the screen and restart the program.

If all that was easy, then you might try answering these questions: 1) What is the value of P when two arrow keys are pressed simultaneously? 2) What happens when you press both the left and right arrow keys simultaneously? 3) Does one of those two keys have precedence, and why? 4) What does the DEFINT in line 10 do for the program? □

$15.95

THE CREATIVE TRS-80

Edited by Ken Mazur

As a compendium of articles, columns, reviews and tutorials written by leading TRS-80 programmers and reviewers, The Creative TRS-80 is a valuable reference guide for every TRS-80 owner.

Nowhere else will you find such comprehensive and thorough discussions on the capabilities of your machine in one handy volume.

Topics covered include:

- Hardware
- Software
- Graphics
- Music
- Programming Tips
- Word Processing
- Education

There is also a lengthy section of revised and updated reprints taken from Creative Computing's popular monthly column, TRS-80 Strings.

And much MORE!!

CREATIVE COMPUTING PRESS

ISBN 0-916688-36-4