80 Microcomputing 9/81 #21

September 1981
US $2.95

80 microcomputing T.M.

the magazine for TRS-80* users

A WAYNE GREEN PUBLICATION

ROBOTICS

Take this hundred
...if you can
This could be yours, but probably never will. This $100 bill is the reward we offer for the discovery of a legitimate bug in a DOSPLUS DISK Operating System. To qualify you must be:
1. A registered DOSPLUS owner
2. Able to demonstrate and document a real bug, not simply a design shortcoming.
We're convinced that DOSPLUS is the fastest, most powerful, and easiest to operate system on the market. DOSPLUS works! And works right! For the businessman and hobbyists, the speed and simplicity cannot be beat. For the BASIC programmer, NO-ONE can offer you more than we do. And for software authors, compatible kernel systems are available for distribution.
NO-ONE ELSE can give you!
•The same high level of Model I/Model III diskette and program compatibility.
•True, complete BASIC program chaining with files open and variables saved!
We have it all. Features galore AND error-free operation. If we're wrong, and you can prove it, we've got a brand new $100 bill for you.
We're not conceited, we're convinced!
Order yours today, and soon you will be too!!
MICRO SYSTEMS SOFTWARE
5846 Funston Street, Hollywood, Florida 33023
Phone: (305) 983-3390
MSS inc.
384
VISA
DOSPLUS
CALL TOLL FREE FOR FAST SERVICE (800) 824-7888, OPERATOR 193FOR VISA/MASTERCHARGE/C.O.D. ORDER
California dial (800) 852-7777, Operator 193, Alaska and Hawaii d
(800) 824-7919, Operator 193
TOLL FREE LINES WILL ACCEPT ORDERS ONLY!
For Applications and Technical information call (305) 983-3390 or drop us a card.
Dealers inquiries invited

TRS-80* COMPUTING EDITION

©1981 Percom Data Co., Inc. **The Percom Peripheral** 35 cents

Percom's DOUBLER II™ tolerates wide variations in media, drives

GARLAND, TEXAS — May 22, 1981 — Harold Mauch, president of Percom Data Company, announced here today that an improved version of the Company's innovative DOUBLER™ adapter, a double-density plug-in module for TRS-80* Model I computers, is now available.

Reflecting design refinements based on both theoretical analyses and field testing, the DOUBLER II™, so named, permits even greater tolerance in variations among media and drives than the previous design.

Like the original DOUBLER, the DOUBLER II plugs into the drive controller IC socket of a TRS-80 Model I Expansion Interface and permits a user to run either single- or double-density diskettes on a Model I.

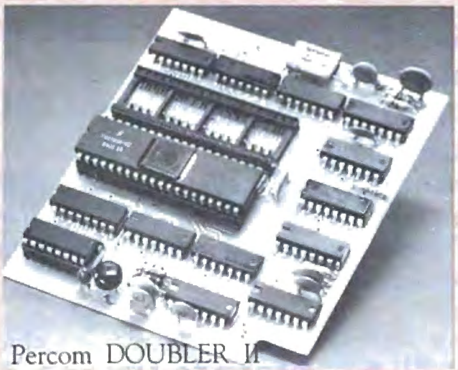
With a DOUBLER II installed, over four times more formatted data — as much as 364 Kbytes — can be stored on one side of a five-inch diskette than can be stored using a standard Tandy Model I drive system.

Moreover, a DOUBLER II equips a Model I with the hardware required to run Model III diskettes.

(*Ed. Note:* See "OS-80™: Bridging the TRS-80* software compatibility gap" elsewhere on this page.)

The critical clock-data separation circuitry of the DOUBLER II is a proprietary design called a ROM-programmed digital phase-lock loop data separator.

According to Mauch, this design is more tolerant of differences from diskette to diskette and drive to drive, and also provides immunity to performance degradation caused by circuit component aging.



Percom DOUBLER II

Mauch said "A DOUBLER II will operate just as reliably two years after it is installed as it will two days after installation."

The digital phase-lock loop also eliminates the need for trimmer adjustments typical of analog phase-lock loop circuits.

"You plug in a Percom DOUBLER II and then forget it," he said.

The DOUBLER II also features a refined Write Precompensation circuit that more effectively minimizes the phenomena of bit- and peak-shifting, a reliability-impairing characteristic of magnetic data recording.

The DOUBLER II, which is fully software compatible with the previous DOUBLER, is supplied with DBLDOS™, a TRSDOS*-compatible disk operating system.

The DOUBLER II sells for $2[illegible]5, including the DBLDOS diskette. NOW $169.95!

✓1

Owners of original DOUBLERs may purchase a DOUBLER II upgrade kit, without the disk controller IC, for $30.00. Proof of purchase of an original DOUBLER is required, and each DOUBLER owner may purchase only one DOUBLER II at the $30.00 price.

The Percom DOUBLER II is available from authorized Percom retailers, or may be ordered direct from the factory. The factory toll-free order number is 1-800-527-1592.

Ed. note: Opening the TRS-80 Expansion Interface may void the Tandy limited 90-day warranty. Circle 258 on inquiry card.

Circuit misapplication causes diskette read, format problems.

High resolution key to reliable data separation

GARLAND, TEXAS — The Percom SEPARATOR™ does very well for the Radio Shack TRS-80* Model I computer what the Tandy disk controller does poorly at best: *reliably* separates clock and data signals during disk-read operations.

Unreliable data-clock separation causes format verification failures and repeated read retries.

CRC ERROR–TRACK LOCKED OUT

The problem is most severe on high-number (high-density) inner file tracks.

As reported earlier, the clock-data separation problem was traced by Percom to misapplication of the internal separator of the 1771 drive controller IC used in the Model I.

The Percom Separator substitutes a *high-resolution* digital data separator circuit, one which operates at 16 megahertz, for the low-resolution one-megahertz circuit of the Tandy design.

Separator circuits that operate at lower frequencies — for example, two- or four-megahertz — were found by Percom to provide only marginally improved performance over the original Tandy circuit.

The Percom solution is a simple adapter that plugs into the drive controller of the Expansion Interface (EI).

Not a kit — some vendors supply an untested separator kit of resistors, ICs and other paraphernalia that may be installed by modifying the computer — the Percom SEPARATOR is a fully assembled, fully tested plug-in module.

Installation involves merely plugging the SEPARATOR into the Model I EI disk controller chip socket, and plugging the controller chip into a socket on the SEPARATOR.

The SEPARATOR, which sells for only $29.95, may be purchased from authorized Percom retailers or ordered directly from the factory. The factory toll-free order number is 1-800-527-1592.

Ed. note: Opening the TRS-80 Expansion Interface may void the Tandy limited 90-day warranty. Circle 508 on inquiry card.

All that glitters is not gold

OS-80™ Bridging the TRS-80* software compatibility gap

Compatibility between TRS-80* Model I diskettes and the new Model III is about as genuine as a gold-plated lead Krugerrand.

True, Model I TRSDOS* diskettes can be *read* on a Model III. But first they must be converted and re-recorded for Model III operation.

And you *cannot write* to a Model I TRSDOS* diskette. Not with a Model III. You cannot add a file. Delete a file. Or in any way modify a Model I TRSDOS diskette with a Model III computer.

Furthermore, your converted TRSDOS diskettes cannot be converted *back* for Model I operation.

TRSDOS is a one-way street. And there's no retreating. A point to consider before switching the company's payroll to your new Model III.

Real software compatibility should allow the *direct, immediate* interchangeability of Model I and Model III diskettes. No read-only limitations, no conversion/re-recording steps and no chance to be left high and dry with Model III diskettes that can't be run on a Model I.

What's the answer? The answer is Percom's OS-80™ family of TRS-80 disk operating systems.

OS-80 programs allow *direct, immediate* interchangeability of Model I and Model III diskettes.

You can run Model I single-density diskettes on a Model III; install Percom's plug-in DOUBLER™ adapter in your Model I, and you can run double-density Model III diskettes on a Model I.

There's no conversion, no re-recording.

Slip an OS-80 diskette out of your Model I and insert it directly in a Model III.

And vice-versa.

Just have the correct OS-80 disk operating system — OS-80, OS-80D or OS-80/III — in each computer.

Moreover, with OS-80 systems, you can add, delete, and update files. You can read *and write* diskettes regardless of the system of origin.

OS-80 is the original Percom TRS-80 DOS for BASIC programmers.

Even OS-80 utilities are written in BASIC.

OS-80 is the Percom system about which a user wrote, in Creative Computing magazine, ". . . the best $30.00 you will ever spend."†

Requiring only seven Kbytes of memory, OS-80 disk operating systems reside completely in RAM. There's no need to dedicate a drive exclusively for a system diskette.

And, unlike TRSDOS, you *can* work at the track sector level, defining and controlling data formats — in BASIC — to create simple or complex data structures that execute more quickly than TRSDOS files.

The Percom OS-80 DOS supports single-density operation of the Model I computer — price is $29.95; the OS-80D supports double-density operation of Model I computers equipped with a DOUBLER or DOUBLER II; and, OS-80/III — for the Model III of course — supports both single- and double-density operation. OS-80D and OS-80/III each sell for $49.95. ✓429

PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE. PRICES DO NOT INCLUDE HANDLING AND SHIPPING.

PERCOM DATA COMPANY, INC. 211 N. Kirby Street Garland, Texas 75042 (214) 272-3421

80 Contents

PUBLISHER
Wayne Green

EXECUTIVE VICE PRESIDENT
Sherry Smythe

ASSISTANT TO PRESIDENT
Matt Smith

CORPORATE CONTROLLER
Charles Garniss, Jr.

ASSOCIATE PUBLISHER
Edward Ferman

ASSISTANT PUBLISHER
Jeff DeTray

DIRECTOR OF MARKETING
(603) 924-7296
Debra Boudrieau

BULK SALES MANAGER
Ginny Boudrieau

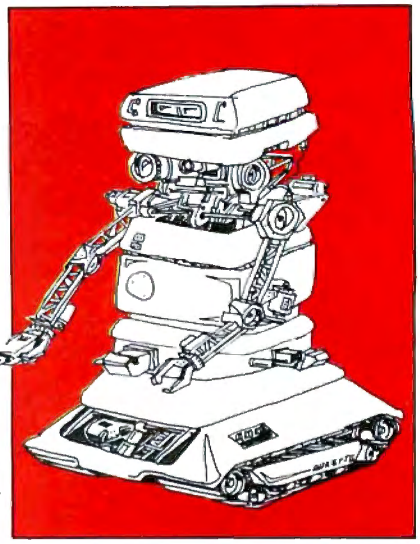
ADVERTISING SALES
(603) 924-7138
John Gancarz
Hal Stephens

COORDINATION
Penny Brooks

OFFICE MANAGER
Marcia Stone

ABC *Paid Audited Circulation*

80 Microcomputing (ISSN ·0199-6789) is published monthly by 1001001 Inc., 80 Pine St., Peterborough NH 03458. Phone: 603-924-3873. Second class postage paid at Peterborough, NH, and additional mailing offices. Subscription rates in U.S. are $18 for one year and $45 for three years. In Canada, $20—one year only, U.S. funds. Canadian distributor: Micro Distributing, 409 Queen St. West, Toronto, Ontario, Canada M5V 2A5. Foreign subscriptions (surface mail), $28—one year only, U.S. funds. Foreign subscriptions (air mail), $60—one year only, U.S. funds. In Europe contact Monika Nedela, Markstr. 3, D-7778 Markdorf, W. Germany. In South Africa contact *80 Microcomputing*, P.O. Box 782815, Sandton, South Africa 2146. Australian Distributor: Electronic Concepts, Attention: Rudi Hoess, 55 Clarence Street, Sidney 2000, Australia. All U.S. subscription correspondence should be addressed to *80 Microcomputing*, Subscription Department, P.O. Box 981, Farmingdale, NY 11737. Please include your address label with any correspondence. Postmaster: Send form ·3579 to *80 Microcomputing*, Subscription Services, P.O. Box 981, Farmingdale, NY 11737.



APPLICATIONS

CONSTRUCTION

EDUCATION

GENERAL

GRAPHICS

REVIEW

TECHNIQUE

DEPARTMENTS

PUBLISHER/EDITOR
Wayne Green

MANAGING EDITOR
Debra Marshall

SENIOR EDITOR
Pamela Petrakos

TECHNICAL CONSULTANT
Jake Commander

PRODUCTION EDITOR
Susan Gross

NEWS EDITOR
Bert Latamore

REVIEW EDITOR
Michael Nadeau

NEW PRODUCTS EDITOR
Janet Fiderio

TECHNICAL EDITORS
Chris Brown
G. Michael Vose

EDITORS
John P. Mello Jr.
Betty Thayer

LAYOUT EDITORS
Joan Ahern
Bob Dukette
Sharon Phinney
Sue Symonds

TECHNICAL CONTRIBUTING EDITOR
Dennis Kitsz

EDITORIAL ADMINISTRATION
Pat Graham
Nancy Noyd

DESIGN ASSOCIATE
Diana Shonk

PRODUCTION MANAGER
Nancy Salmon

ASST. PRODUCTION MANAGER
Michael Murphy

AD GRAPHICS MANAGER
Robert Drew

AD COORDINATOR
Patty Mackowsky

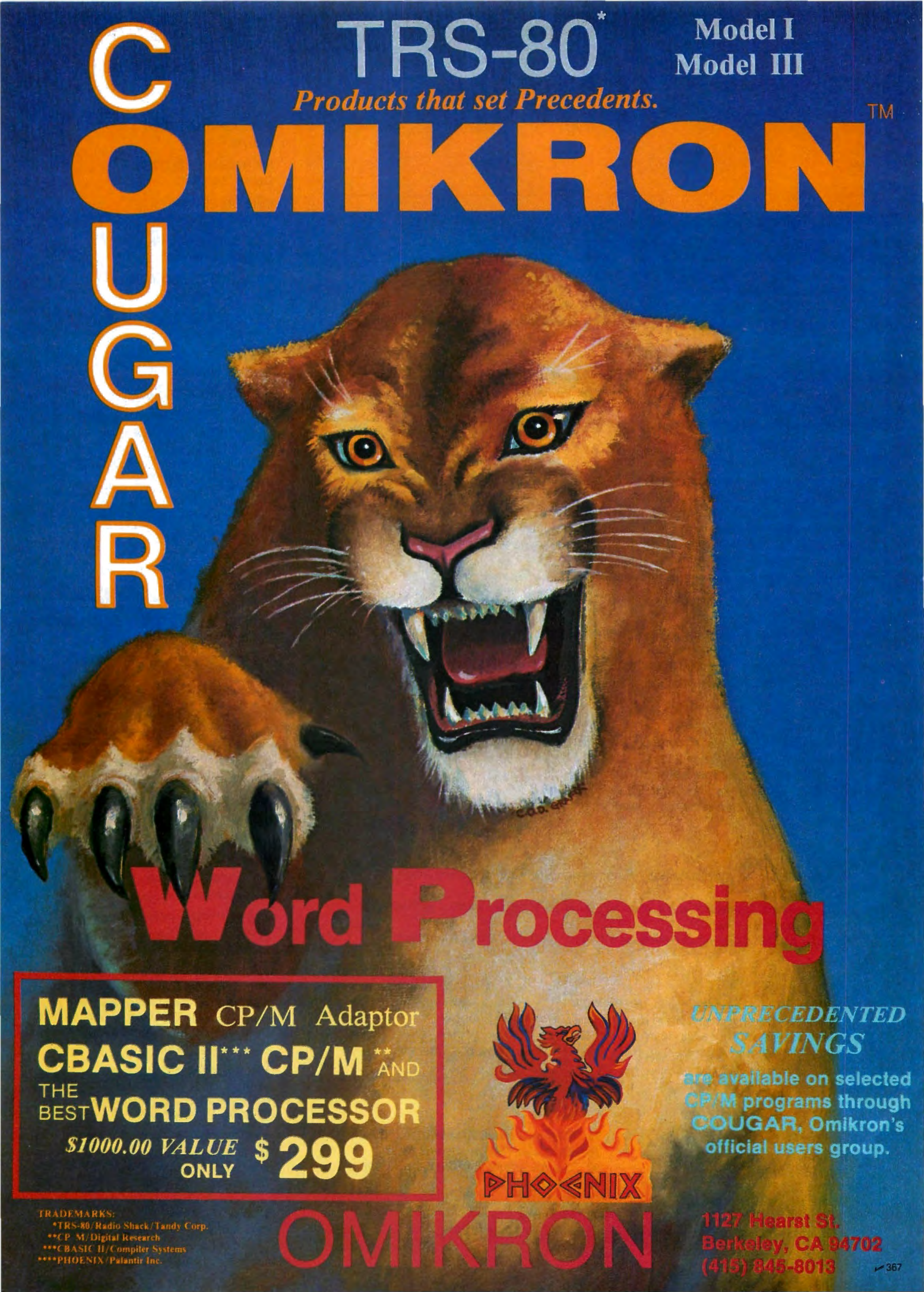
ADVERTISING PRODUCTION
Steve Baldwin, Bruce Hedin, Jane Preston, Dennis Christensen

PRODUCTION DEPT.
Fiona Davies, Linda Drew, Sandra Dukette, Kenneth Jackson, Ross Kenyon, Theresa Ostebo, Dianne Ritson, Deborah Stone, Judy Wimberly, Donna Wohlfarth

PHOTOGRAPHY
William Heydolph, Terrie Anderson, Bill Suttenfield, Paul Babich, Thomas Villeneuve

TYPESETTING
Sara Bedell, Michele Desrocher, David Hayward, Stephen Jewett, Mary Kinzel, Ann Lord, Kelly Smith, Karen Stewart

Cover by Alex Stevens


COUGAR
TRS-80*
Model I
Model III
Products that set Precedents.
OMIKRON™
Word Processing
MAPPER CP/M Adaptor
CBASIC II*** CP/M** AND
THE BEST WORD PROCESSOR
$1000.00 VALUE ONLY $299
PHOENIX
OMIKRON
UNPRECEDENTED SAVINGS
are available on selected CP/M programs through COUGAR, Omikron's official users group.
1127 Hearst St.
Berkeley, CA 94702
(415) 845-8013
367
TRADEMARKS:
*TRS-80/Radio Shack/Tandy Corp.
**CP M/Digital Research
***CBASIC II/Compiler Systems
****PHOENIX/Palantir Inc.

80 REMARKS

by Wayne Green

"Others may use their Sony Walkman cassette players for rock, mine is belting out Trail by Jury and Ruddigore."

"If you wish in the world to advance
Your merits you're bound to enhance
You must stir it and stump it
And blow your own trumpet
Or trust me you haven't a chance."
W.S. Gilbert, Ruddigore

Back in my formative days, I was exposed to a rabid Savoyard and became a carrier of this form of mental illness: hence the opening ditty. By the age of twelve I could sing the Nightmare song from Iolanthe (this is one of the faster Gilbert and Sullivan patter songs). In high school I joined the Savoyards club and we put on The Mikado and Pirates. Hello all you G&S fans—and to the uninitiated, you're *missing* something.

Others may use their Sony Walkman cassette players for rock, mine is belting out Trial by Jury and Ruddigore, with a bit of bluegrass now and then.

Green Tech

Some months ago I got to stewing about the shortage of microcomputer-oriented people. Sure, we have a hundred thousand hobbyists, but nowhere near the number of micro-educated people our growing industry really needs. Most schools, even though buying micros, have not been teaching the skills which are needed in our industry.

There are over 2,000 stores selling computers at present. Estimates are that this will grow, as software becomes more available, to over 50,000 stores. We're going to need store managers, technicians, salesmen, programmers, and so on just to fill this need. In order to move the programs and hardware from the manufacturers (or importers) to the stores we will need distributors or reps and these people, too, will have to have an understanding of micros.

Even with massive importing of hardware from Japan there will still be a need for large numbers of people to run the smaller businesses which will build up around these systems. Look at the host of firms which are making nice livings providing accessories and software for the TRS-80; when we also have IBM, DEC, DG, Panasonic, Casio, Sony, Sanyo, Hitachi, Toshiba, Sharp, Quasar, Pioneer, Sansui, Sord, Mitsubishi and others, we're going to have a big market and need a lot of trained people.

I have been looking for someone who could start work on plans for a technical institute which would teach the fundamentals of microcomputers. My concept is to have a blitz course which would last 18 months to two years, thus keeping down the cost of college. The course would run straight through, with no vacations, no beer bashes, no soccer teams, no dances...just hard work and plenty of it. It would run six days a week, about twelve hours a day.

The courses would cover both the hardware and software aspects of micros, complete with extensive lab work. There would also be an array of business courses so that graduates would be able to work their way up into the management of businesses, rather than being primarily technicians or engineers. This is the area where most firms have failed, rather than in technical designs. We would teach such things as business law, personnel management, marketing, salesmanship, corporate financing, accounting, advertising and promotion, writing, speed-reading, gamesmanship, taxes, quality control, packaging, automation, and so on.

One of the problems many students have when they graduate from school is finding work when they have had no practical experience. This is the Catch-22 situation. With Instant Software growing rapidly, we may be able to intern students there part time so they will have had professional business experience for their resumes. They would be able to do service work on the myriad of computers in the Instant Software lab, help with evaluation of submitted programs, write programs in their spare time for possible publication, work on making production masters of the programs, duplication, quality control, packaging, advertising, promotion, sales, and so on. With Instant Software sales heading toward $10 million and more, there will be plenty of work for hundreds of students.

It will be a year or two before this plan will get off the ground, but a staff is starting to work out the details. ■

International Computer Students

On a recent trip to St. Lucia, a small island in the Caribbean, I talked with the people there and found that it is presently very difficult for students from smaller foreign countries to come to the U.S. for an education. Yet it is in these countries where it is needed the most, and where it can have the best long-range effect, not only for the country involved, but for the U.S.. Russia figured that out a long time ago and they regularly bring in foreign students for a Russian education. The investment would be small to bring students to the U.S. and put them into schools, particularly such as the one I envision.

Smaller countries must have some way of keeping up with technology if they are not to fall further and further behind. And if the U.S. does not help these smaller countries, particularly in our hemisphere, you can bet that Russia and Cuba will. You can also bet that the students will get more than a technical and business training.

Having visited quite a few small countries, meeting hams and computer people in these areas, I feel that this is important enough to warrant being considered part of our defense budget. Helping countries to grow and spreading the friendly American influence is a way to save on future military expenses.

In most cases we wouldn't even have to do more than provide the educational cash up front, just as we are doing for American students. These students would be delighted to do just about anything to earn as much of their keep as possible while they are in school here, and repay the rest of the investment when they return home. We might start thinking in terms of more schools where students have an opportunity to work part time to help pay for their education.

By the way, I want you to know that the hams of St. Lucia, several of whom are already into microcomputers, pulled out all

Continues to page 67

EDUCATION 80

by Earl R. Savage

"A flow chart can be of significant help in planning a program and 'deciphering' one someone else wrote."

The use of a split screen can greatly enhance the effectiveness of many types of instructional programs. Suppose you wish to discuss or ask questions about a chart, diagram, quotation or math probiem. The normal procedure is to keep the figure before the student during the discussion. The figure is usually at the top of the display and a series of statements or questions below it.

That works fine, except that as you continue to write at the bottom of the display, the figure scrolls off the top. You need the lower part of the display to scroll independently of the upper part. This can be done but it is a complex maneuver on the Model I. Fortunately, there are several simpler ways to accomplish the same effect.

Several Solutions

We should pause here to note the Model III has this function built in. It allows you to protect up to seven upper lines from scrolling just by POKEing a digit into a memory location. If you need more than seven lines space you must resort to programming techniques.

The least effective solution is to use a subroutine to draw and re draw the figure. When you wish to change the bottom of the display, clear the screen and call the subroutine to re draw the figure before writing again at the bottom. This procedure is slow and it requires unnecessary memory overhead.

The most common solution is to simply write over the old information at the bottom of the display. Not only is this sloppy in appearance, but you must remove all the old writing. The presence of old and new writing together will only confuse the student.

The simplest and most effective solution to split screen operation is to use the control code CHR$(31). For example, this statement will clear the display from the beginning of line 10 to the bottom:

```
PRINT@576,CHR$(31);
```

Note the trailing semicolon. Failure to include it will cause problems with the figure.

This statement not only clears the last seven lines but leaves the cursor at print location 576 (beginning of line 10). Without taking further action, you are ready to write at the top of the cleared area.

If your program will use this procedure several times, place it in a subroutine for efficiency. In this case, make the print location a variable like this:

```
2080 PRINT@P,CHR$(31);:RETURN
```

You can now clear various portions of the screen as your needs change. The subroutine would be called by: P=576: GOSUB2080. Once the value of P is established, you need not specify it again until it is to be changed.

This technique will permit you to clear from any point on the display. If you need only to specify line beginnings, this change in the subroutine will save a few bytes of memory:

```
PRINT@(L-1)*64, CHR$(31);:RETURN
```

Simply specify the line number for L. If you will remember to start counting lines at zero instead of one, write your subroutine this way:

```
PRINT@L*64,. . . . .
```

With this technique, you can erase and re use the bottom portion of the display as many times as the program plan requires. In effect, you have split-screen operation without the hassle of machine code routines.

In a true split screen, one part scrolls independently of the other, which isn't the case here. When writing on the last line, be careful not to cause the display to scroll or you will lose part of the upper figure. Judicious use of trailing semicolons on the last line will prevent scrolling. You can also simply leave the last line blank.

Using Flow Charts

It seems that a large portion of the folk in educational computing ignore flowcharting altogether, or at best view it with suspicion and rarely use it.

If you fall into that category, you're missing out on a valuable tool. A flow chart can be of significant help in planning a program and "deciphering" one someone else wrote. In both instances, you can get lost easily.

Short, simple programs or those progressing from start to end with few side trips can be kept in mind quite well. It is with more complex programs that flow charts are really needed.

The diamond symbol (Fig. 1) is used at each program decision point. There is a question in this box and, depending on the answer, program action may proceed in one of several directions. The diamond box will therefore have more than one arrow leading from it.

The rectangular box symbol is used for most of the actions in any program. It indicates there is no question or condition

Continues to page 67

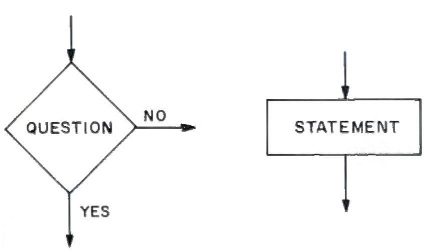


Fig. 1. Flow Chart Symbols

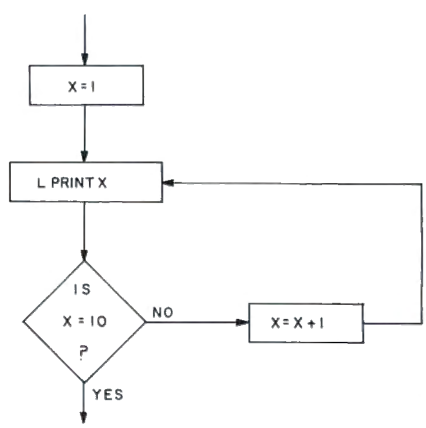


Fig. 2. Charting a For. . . Next loop

80 INPUT

"(He) seems to be under the illusion I threw a few integrated circuits into a bag (and) shook it up. . ."

Joystick Not Foolproof

In response to Dennis Kitsz's letter expressing concern over what he calls theoretical problems with my joystick construction story in the June 1981 issue, I am sure the readers understand the point Mr. Kitsz seemed to miss. The joysticks were intended only to be used with a suitable program to draw or play games and not to be left on line when running other programs or when using additional peripheral devices. That is why there is no off-on switch. The user must unplug the joystick interface to disconnect the batteries, if not they will go dead in which case the unit will not operate and a "Theoretical Bus Contention" would not be possible. That is why I said in the article "practically foolproof." (Sorry Dennis, I didn't mean to fool YOU.)

When a joystick program is running it is impossible to CLOAD at the same time. And Inp(1) will not open the cassette port; Inp(255) does that. I see no problem there.

As far as the hex inverters are concerned, all that they do is pull some of the data lines down to ground potential; this is also done by the keyboard and all other input devices, and is not harmful to the computer.

In theory an AA cell is 1.5 volts, but in reality it provides somewhat less voltage. Four AA cells connected in a series and operating under a load, such as the joystick interface, provide only 5 volts. I have tested the device with several brands of batteries and all developed less than 5 volts.

Mr. Kitsz states that a possible problem could arise if the joystick was pushed to zero. Again he is in error. Had he read the article carefully he would have seen that 241 is the lowest data number the circuit develops.

Dennis seems to be under the illusion that I threw a few integrated circuits into a brown paper bag, shook it up, and by some incredible stroke of luck, as he puts it, came forth with a working device. (Witchcraft?)

Just for the record, Dennis, this was a carefully thought-out and tested project. All the joystick units that I have built since the first one (21 months ago) still work perfectly. Have you built even one and tested it?

May I remind the readers about the scientist who claimed that, theoretically, a Bumble Bee cannot fly because its wings are too small for its body. Well folks, it does fly and so do my joysticks.

Frank DiNunzio
Bristol, PA

Semiliterate

Mr. Zeppa's letter concerning Harv Pennington and his *Disk and Other Mysteries* is a real jewel (June 1981). Literacy is the ability to communicate ideas and thoughts. Harv did a masterpiece of communicating a technical and complicated subject in a refreshing and informative way. I suppose Mr. Zeppa likes dry and methodical technical writing. I'd bet Mr. Zak's work on the Z-80 would give Mr. Zeppa goose bumps! I have enjoyed reading Pennington's writing for some time. Writers such as Pennington and Dr. Lien have done a service to laymen in the microcomputing field.

I quote from Mr. Zeppa, "I would still be embarrassed by the semiliterate style, or lack thereof, of *TRS-80 Disk and Other Mysteries."* Now, does Mr. Zeppa mean the work was lacking semiliteracy? I'm confused. Surely he meant the work was not up to his literary standards. Why in the hell didn't he say that? I rest my case.

Steve Wright
Dayton, TX

Praise for Pennington

I can't believe what people are saying about H. C. Pennington's *TRS-80 Disk and Other Mysteries.* Just to learn about restoring killed files is well worth the price of a hundred books. I never thought that so much information could be stuffed into a book that size. Mr. Pennington turns Superzap from a thing used to put in Apparat zaps to a very useful utility. I have read the book from cover to cover and have learned many useful things. Mr. Pennington explains in English how to recover every type of file that I could dream of! His program Search has saved me many hours of work. I can now restore files as easily as I can kill them. I no longer have to remember passwords because I can easily change them to whatever I want whenever I want. I now know how to speak binary, thanks to Mr. Pennington.

Robert Smicinski
Amsterdam, NY

Shack Selling Bootleg

I was surprised by the comments attributed to Mr. Ed Juge on page 58 of the June 1981 issue of *80 Microcomputing* concerning protection of Radio Shack software. It surprised me because if anyone is guilty of giving away other company's proprietary software without permission, it is surely Radio Shack. How often have Radio Shack dealers or computer stores given away bootleg copies of other companies' software to their customers? It would appear that Mr. Juge wants to apply a double standard to the computer community. Radio Shack software is sacred but that of other companies can be freely "used."

If Radio Shack is going to abuse the rest of the industry by their own actions, then it is unreasonable for them to expect that they too should not share in such treatment.

John Paul Kapp, president
JK Consulting, Inc.
Baton Rouge, LA

Isolated Incidents

I would feel it was reasonable and proper for all authors to be able to protect their software from unauthorized copying. Mr. Kapp seems to assume that I am advocating that only Radio Shack's software should be protected.

With over 6,000 outlets in the United States, I'm sure there are several people in our employ who do not follow the rules.

META TECHNOLOGIES

26111 Brush Avenue, Euclid Ohio 44132
CALL TOLL FREE 1-800-321-3552 TO ORDER
IN OHIO, call (216) 289-7500 (COLLECT)

1001 THINGS TO DO WITH YOUR PERSONAL COMPUTER

BY MARK SAWUSCH

333 pages $7.95

333 pages, written in simple terms, of "what-to-do" and "how-to-do-it". Suitable not only for microcomputers, but for programmable calculators as well. Includes program listings, formulas, a glossary of computer terms and more! Definitely a MUST BUY!

A PARTIAL LIST OF APPLICATIONS

- Real Estate Evaluation
- Astrology
- Income Tax
- Speed Reading
- Personality Test
- Statistical Analysis
- Logic Circuit Analysis
- Carpenter and Mechanic's Helper
- General Purpose Clock Timer
- Test Your Typing Speed
- Finances & Investments
- Biorythm
- Energy Efficiency
- Antenna Design
- Letter Writing
- Recipe Index/Calculator

"OTHER MYSTERIES" VOLUME III

by Dennis Kitsz

Call now and place your order for this new book, "THE CUSTOM TRS-80™ & OTHER MYSTERIES", from IJG, Inc. More than 300 pages, with over 60 photographs, of projects for the hardware hobbyist. Includes schematics, PC layouts, software driver code, etc. for such do-it-yourself undertakings as high resolution graphics, reverse video, real-time clock/calender, music synthesis, ROM/RAM additions and more!

THE CUSTOM TRS-80™ $29.00
CALL FOR AVAILABILITY

MICROPARAPHERNALIA

NEWDOS by APPARAT

NEWDOS/80 by Apparat $149.95
NEWDOS + to
NEWDOS/80 UPGRADE ..... CALL
NEWDOS + with ALL UTILITIES
35-track $69.95
40-track $79.95

BOOKS

TRS-80™ DISK
AND OTHER MYSTERIES .. $19.95
MICROSOFT™ BASIC DECODED $29.95

Let your TRS-80™ Teach You

ASSEMBLY LANGUAGE

REMSOFT's unique package, "INTRODUCTION TO TRS-80® ASSEMBLY PROGRAMMING" includes ten 45-minute lessons on audio cassettes, a display program for each lesson providing illustration & reinforcement, and a text book on TRS-80® Assembly Language Programming. Includes useful routines to access keyboard, video, printer and ROM. Requires 16K - Level II, Model I.

REMASSEM-1 .... $69.95
FOR DISK SYSTEMS $74.95

Let Your TRS-80™ Teach You

ASSEMBLY LANGUAGE DISK I/O TECHNIQUES

REMSOFT does it again! REMDISK-1 is a concise, capsulated supplement to REMASSEM-1. Package consists of two 45-minute lessons on audio cassettes, and display programs providing illustration and reinforcement. Provides specific track and sector I/O techniques, and sequential and random file access methods and routines.

REMDISK-1 .... $29.95

Let Your TRS-80™ Test Itself With

THE FLOPPY DOCTOR & MEMORY DIAGNOSTIC

by THE MICRO CLINIC

A complete checkup for your Model I. THE FLOPPY DOCTOR completely checks every sector of 35- or 40-track disk drives. Tests motor speed, head positioning, controller functions, status bits and provides complete error logging. THE MEMORY DIAGNOSTIC checks for proper write/read, refresh, executability and exclusivity of all address locations. Includes both diagnostics and complete instruction manual.

SYSTEM DIAGNOSTICS $19.95

An improved version of the SYSTEM DIAGNOSTICS above. Designed for single or double density, 35-, 40-, 77-, or 80-track disk drives. Includes new and modified tests. Features THE FLOPPY DOCTOR, Version 3.0.

SYSTEM DIAGNOSTICS-V3 .. $24.95

Single Sided, Soft-Sectored 5¼-inch, (for TRS-80™) Mini-floppy

DISKETTES $21.95 box of 10

PLAIN JANE™

These are factory fresh, absolutely first quality (no seconds!) mini-floppies. They are complete with envelopes, labels and write-protect tabs in a shrink-wrapped box.

PLAIN JANE™ Diskettes $21.95
10 boxes of 10 .... (each box)$21.50

PLAIN JANE™ *Gold*

Introducing MTC's premium generic diskette. Single-Sided, Soft-Sectored, DOUBLE-DENSITY, 5¼-inch diskettes with reinforcing HUB-RINGS. Individually 100% ERROR-FREE certified. Invest in GOLD!

PLAIN JANE™ *Gold* $25.95

VERBATIM'S PREMIUM DISKETTES

DATALIFE™

Seven data-shielding improvements mean greater durability and longer data life. These individually, 100% error-free certified diskettes feature thicker oxide coating, longer-lasting lubricant, improved liner, superior polishing and more! Meets or exceeds IBM, Shugart, ANSI, ECMA and ISO standards.

VERBATIM DATALIFE™ DISKETTES
5¼-inch (box of 10)
MD525-01 .... $26.95
10 boxes of 10 .... (each box)$25.95
8-inch FLOPPIES
Double-Density, FD34-8000 . $43.95

'RINGS' & THINGS

HUB RING KIT for 5¼" disks..... $10.95
HUB RING KIT for 8" disks....... $12.95
REFILLS (50 Hub Rings) .... $ 5.95
CLEANING KIT for 5¼" drives ... $24.95
5¼-inch diskette case $3.50
8-inch diskette case $3.95
5 1/4-inch File Box for
50 diskettes $24.95
8 inch File Box for
50 diskettes $29.95

MOST ORDERS SHIPPED WITHIN ONE BUSINESS DAY

Products damaged in transit will be exchanged.

PRICES IN EFFECT Sept. 1, 1981 THRU September 30, 1981.
Prices, Specifications, and Offerings subject to change without notice.
8109

WE ACCEPT
- VISA
- MASTER CHARGE
- CHECKS
- MONEY ORDERS
- C.O.D.

- Add $3.00 for shipping & handling
- $3.00 EXTRA for C.O.D.
- Ohio residents add 5½% sales tax.

Store managers are prohibited from carrying or even giving away non-Radio Shack software.

This occasional breaking of the rules that we certainly try to control and eliminate is all the more reason why software protection in the industry would be good for everyone.

Ed Juge, director
Computer Merchandising
Tandy-Radio Shack
Fort Worth, TX

Heavens Tibetsy, is Kapp right... or Juge? How about it readers, have you come across freebies from Shackers, or is Kapp exaggerating. Tell me... Wayne

Fix Needs a Fix

Oops! Someone goofed in the *80 Input* section (April, 1981, p. 14) in the letter entitled "Shack Statistical Fix."

Actually the last statement in line 280 of the Radio Shack Advanced Statistical Analysis (26-1705) program for the computation of Chi Square should read: CS = CS + (ABS(O(I,J) – E(I,J)) – CC)↑2/E (I,J).

Only two changes are necessary to the original program—the addition of ABS for the absolute value portion of the computation, and the addition of a parenthesis before the subtraction sign preceding CC in the formula. CC is Yates' 0.5 correction for continuity which is to be used when the df = 1.

Statistically speaking, you should get it printed right this next time.

Edith Cracchiolo
Cerritos Community College District
Norwalk, CA

Model III Bugs

I recently purchased a Model III 48K with one disk drive after selling my Model I. Radio Shack did a good job of cleaning up many Model I problems. The hardware for the Model III is very impressive; the package is one of the best I have seen in the micro price range. The keyboard allows me to whiz along at 70 to 80 words per minute. The new disk controller allows a physical sector to be read or written much faster than on Model I.

However, the people in the software department merely rehashed TRSDOS 2.3 and added a few more bugs and annoyances. I will list just a few:

- You can't kill Basic/CMD since they didn't give you the password and there are three systems on the disk that aren't in the directory that are part of Basic. So assuming you don't want Basic on a disk, tough luck (and 12 wasted grans).
- The LOC function still does not work in Basic. There was one small patch to be made on Model I which they didn't make. To make it work on Model III, patch address 5E0CH from CD 75 60 to CD 27 5A and LOC(n) will now return the current record number for file n.
- The concept of a logical record seems to be completely lost on Radio Shack. Basic now supports logical record lengths other than 256 (as did some Model I Basics), but every time you read or write a record TRSDOS reads or writes a sector, then does a block move. If you write one byte, TRSDOS has to write 256 because it doesn't know how to block records (TRSDOS 2.3 does block records).

The worst case is one byte logical records (stream I/O). Many programs like to think this way, for example Electric Pen. I wrote the following program with one byte logical records to test the problem:

```
10 OPEN"R",1,"TEMP",1 ' REMEMBER TO
   ANSWER 1V TO How Many Files?
20 FIELD 1,1 AS A$
30 FOR N = 1 to 256
40   LSET A$ = "A"
50   PUT 1,N
60 NEXT N
70 CLOSE
```

It took 160 seconds—a baud rate of 12.8. only 12.8 baud for disk!! I'm going back to tape.

- Debug will not dump below 5600H, cannot be entered from Basic, has a worse display mode than Model I, and overlays all the programs it used to debug. For the first problem, zap the following addressed from 5600H to 0000H; track 0, sector 10H, relative byte DDH, sector 11H, RB 09H, sector 12H, RB 73H. I have also put Debug in relocatable source form to solve the last problem. Anyone interested in doing this can write to me.
- DOS overlays all the error messages every time it gets an error, then prints an obnoxious "* * ERROR xx * *". To correct this, zap track 10H, sector 10H, RB 2DH from 20H to 18H. You will now get a meaningful error message in the same amount of time as it took for "* * ERROR xx * *".

Despite some severe software goofs, I am satisfied with my Model III and believe it to be the best purchase on the micro market.

Ervan Darnell
Star Route
Palmyra, MO 63461

Where's Line 83?

I was more interested in a workable keyword indexing and retrieval program (April 1981, page 252) than in a tiff over an unappreciated but cute(?) title heading ("WHERZIT").

Program Listing 1 (Lines 73,75,77,79) calls for branches to Line 81, which in turn calls for a branch to Line 83. What Line 83? The following seemed to suffice for me and may help others who were miffed by the missing line(s):

```
83 IF Q = 0 THEN N = N + S:GOTO 63
84 IF Q$ = "C" THEN R$(N) = " ":GOTO 64
85 GOTO 64
```

Also, Clear 11500 (Line 1) caused an OM message on my Level II 16K. Clear 11000 corrected that.

Leslie J. Schnierer
Institute for Consciousness and Music
New Castle, DE

Color Computer Disappointment

I read with interest Wayne Green's editorial in the July issue, which dealt in part with the lack of response from your readers with Color Computers. I'd like to tell you of my experiences with the Color machine.

In late April, I bought one of the first 16K enhanced machines to come into this part of Ontario. I can say, without reservation, that I was really, really disappointed in the machine. What I didn't like: big things like 32 characters per line, that bloody flashing (or changing cursor), poor resolution on the TV screen (and I was using a *good* TV set). There was this sudden disappearance of 5 + K of memory, dedicated to graphics (granted, it can be reclaimed).

But most of all, it was the emphasis on graphics in the manual, to the chagrin of people like me who intend to use the machine for engineering applications.

I took the machine back under the 30-day guarantee, and bought a 16K Level II Model I. Much to their credit, the local Computer Store manager advised me that the machine was going out of production, as did the salesman who sold it. I didn't care; the price was right ($100 more than 16K color), it had a display, it had a cassette machine and it had the backup of all of the advertisers I saw in your magazine. I like it a lot. I'm doing engineering stuff related to my work on it, and I'm also writing a record index program for it, to help me in my preparations for an FM program.

Continued on page 18

META TECHNOLOGIES

26111 Brush Avenue, Euclid Ohio 44132
CALL TOLL FREE 1-800-321-3552 TO ORDER
IN OHIO, call (216) 289-7500 (COLLECT)

MTC AIDS-III™

MODEL I . . .$69.95 MODEL II . . .$99.95

Introducing the latest addition to MTC's family of data management systems, AIDS-III. NO PROGRAMMING, easy to use. COMPLETE PACKAGE including demonstration application, documentation and MAPS-III (see below).

- Up to 20 USER-DEFINED FIELDS of either numeric- or character-type.
- CHARACTER-type fields may be any length (total: up to 254 characters).
- NUMERIC-type fields feature automatic formatting, rounding, decimal alignment and validation.
- Full feature EDITING when adding or changing records:
  - ENTER FIELD (can't type-in more characters than specified).
  - BACKSPACE (delete last character typed).
  - DELETE FIELD contents.
  - RESTORE FIELD contents.
  - RIGHT-JUSTIFY FIELD contents.
  - SKIP FIELD (to next or previous field).
  - SKIP RECORD (to next or previous record).
- SORTING of records is MACHINE CODE assisted.
  - 200 RECORDS (40 characters) in about 5 SECONDS.
  - ANY COMBINATION of fields (including numerics) with each field in ascending or descending order.
- SELECTION of records for Loading, Updating, Deleting, Printing and Saving is MACHINE CODE assisted.
  - Specify up to 4 CRITERIA, each using one of 6 RELATIONAL COMPARISONS.
  - LOAD or SAVE selected records using MULTIPLE FILES.
  - Example: Select records representing those people who live in the state of Colorado, but not in the city of Denver, whose last names begin with "F" and whose incomes exceed $9000.00.
  - Example: Select records representing those sales made to XYZ COMPANY that exceed $25.00, between the dates 03/15 and 04/10.

MAPS-III (MTC AIDS PRINT SUBSYSTEM), included at no charge, has the following features:

- Full AIDS-III SELECTION capabilities.
- Prints user-specified fields DOWN THE PAGE.
- Prints user-specified fields in titled, columnar REPORT FORMAT, automatically generating column headings, paging and (optionally) indentation.
- Can create a single report from MULTIPLE FILES.
- Prints user-defined formats for CUSTOM LABELS, custom forms, etc.

BELOW ARE TESTIMONIALS from owners of AIDS systems. These are absolutely authentic statements and are typical of the comments we receive.

"This program will do more for my business than all the other programs I have, combined."
David Wareham, Vice President (EDP), National Hospital and Health Care Services Inc.

"We have 32 different Data Base Management packages for the TRS-80. AIDS-III is easily the best. It also makes it easier for us to step up to our Model II since the package is available for both computers."
Jack Bilinski, President, 80 Microcomputer Services

"Your AIDS program is far and away the finest information management system that I've ever seen. I am currently using it to maintain a clear picture of the demographic data on all the kids in our residential treatment program and it is working for me superbly."
Frank Boehm, Director, Front Door Residential Treatment Program

AIDS OWNERS! WE HAVE WHAT YOU'VE BEEN WAITING IV. . .

MTC CALCS-IV™, that is.

- More Computations
- Save Report Formats on Disk
- Faster, and more!

MTC CALCS-IV™. $39.95
For Model II. $59.95

CALL FOR CALCS-III UPGRADE PRICING

MTC AIDS MERGE-III™

This subsystem will combine up to 14 AIDS-created data files into a single, large file. An optional purge capability removes duplicate entries while performing the merge operation (can even be used to eliminate duplicates in a single file). Machine-code assisted for high-speed performance, MERGE-III™ properly handles files sorted by any combination of fields, including numerics, with each field in ascending or descending order.

MTC AIDS MERGE-III™. $19.95
For Model II $29.95

THE COMPLETE MTC AIDS-III™ PACKAGE

SAVE $$$$

Includes MTC AIDS-III™ CALCS-III™ and MERGE-III™

A comprehensive system at a competitive price!

MODEL I . . $99.95
MODEL II . $149.95

MTC AIDS CALCULATION SUBSYSTEM-III™

MODEL I . . .$24.95 MODEL II . . .$39.95

MTC's most popular AIDS subsystem. Use for report generation involving basic manipulation of numeric data. Features are:

- User-specified page title
- Columnar Headings
- Optional Indentation
- Use for accounting, inventory, financial and other numeric-based information systems.
- Columnar subtotals generated when there is a change in a user-specified column.
- User-specified Columnar Totals
- Columnar values computed using constants and/or column values
- Balance forward calculations (Ex: Gross sales equals previous gross sales + sale amount + sales tax).

Compare AIDS-III™/CALCS-III™ with any other data management package under $100!

CALCS-III™ REQUIRES THE PURCHASE OF AIDS-III™

MORE PRODUCTS ⟷

TRS-80 is a trademark of the Radio Shack Division of Tandy Corporation. DATALIFE is a trademark of VERBATIM. PLAIN JANE, AIDS-I, AIDS-III, CALCS-III, CALCS-IV, MERGE-III are trademarks of MTC.

MOST ORDERS SHIPPED WITHIN ONE BUSINESS DAY

Products damaged in transit will be exchanged.

PRICES IN EFFECT Sept. 1, 1981 THRU September 30, 1981. Prices, Specifications, and Offerings subject to change without notice.
8109

WE ACCEPT
- VISA
- MASTER CHARGE
- CHECKS
- MONEY ORDERS
- C.O.D.

- Add $3.00 for shipping & handling
- $3.00 EXTRA for C.O.D.
- Ohio residents add 5½% sales tax.

80 AID

Connecting Printers Without Expansion

I am just getting into computers and I own a TRS-80 Level II. I would like to connect an Epson MX-80 printer to my computer, but I do not have an expansion unit thus far. I would like to know if there is any way of connecting it to the keyboard unit without the expansion unit. I have called around and some places say yes and some say no. Could you please let me know if it can and how to do it, or if there is a company that I can get in touch with.

David D. Johnson
23720 Mary
Taylor, MI 48180

Supervisor Calls

Do you know where I might find information on running supervisor calls for my Model II 64K TRSDOS micro? Specifically the *sort* for *release 2.0.*

Darnell Lepre
2821 Lawnwood Drive
Ocean Springs, MS 39564

Using Scripsit

This is in reference to the letter from Petralia, Neild & Webb in *80 Aid* (May 1981) that stated they have a Daisy Wheel II printer that has special characters (such as section and paragraph signs) that are not accessible through their Model II Scripsit program.

There are two ways to fix this problem:

Boot up Scripsit disk, type in date and press Enter. When time question appears, press hold, then press Enter. You will be in TRSDOS. Now type the following and press Enter:

PATCH SCRIPSIT A = BE4F F = A3A5 C = COAF

This will replace the British pound sign with a section sign and the micro sign with a paragraph sign. (Please see Scripsit Reference Manual, p. 52 for the table of 10 special characters.)

Now the section sign can be obtained by typing CTRL 6, CTRL 9,6 and the paragrah sign by typing CTRL 6, CTRL 9,7.

The second method is to Superzap for Model II.

On the Scripsit disk on track 15 (OFH), sector 10 (OAH) you will see 10 bytes at location C6H that correspond to the 10 special characters in the table on p. 52 in the reference manual. The existing characters are 5E, 6O, 7E, A7, BE, 9C, A3, A5, A6 and BF.

Any of these can be changed to any of the codes available on the Daisy Wheel II as shown in the Printer Owner's Manual on page 22.

In the patch overleaf, I chose to replace the British pound sign and the micro sign with the section ad and paragraph signs as requested by Petralia, Neild & Webb.

Sayyed A. Bashir
Fort Worth, TX

Phantom Records

As you may well know, the TRS-80 Model III differs slightly from the Model I for which most of the listings in *80 Microcomputing* are written.

Having only a cassette-based Model III, 32K, I was delighted with the prospect of using the Sans Disk program (from the April issue) until disk drives are affordable. Much to my dismay, the *re-packing* function of the Sans Disk program will not execute on my Model III, leaving instead a numbered record with the first field name and equal sign followed by blanks! *This phantom record is impossible to delete.* Attempting to delete a record appears to work until the *select* function is executed. When the phantom record appears *save*ing to tape will not re-pack the file as I hoped it would.

I have written to the author of the program who is not familiar with the Model III. He was quite helpful providing other fixes to Sans Disk but couldn't give me any help on the malfunctioning *delete* command. I have carefully checked and re-checked the listing and must assume that the subtle differences of the Model III are to blame.

If anyone has corrected the program to run on the Model III, I'd sincerely appreciate hearing from them. I don't yet know enough about TRS-80 Basic or machine language to tackle the problem myself.

Colin Alexander
120 28th St.
San Francisco, CA 94131

A Fix and a Question

I've solved the problem mentioned by Joe Brandiner on page 24 of the April, 1981, *80*. The answer is to scrap the existing initialization routine and use a simpler one. This routine is modified from Dennis Kitsz's KBEEP program.

```
                  ORG 7DC1           ; Change to BDC1
                                       for 32k, FDC1 for 48
7DC1 21FC7D  LD HL,7DFCH             ; ULCBAS keyboard
                                       driver, change to
                                       BD or FD
7DC4 221640  LD (4016H), HL          ; Driver address
                                       location
7DC7 21527F  LD HL,7F52H             ; ULCBAS video
                                       driver, change to BF
                                       or FF
7DCA 221E40  LD (401EH), HL
7DCD C37200  JP 0072H                ; Change to C3191A
                                       for new Level II
                                       ROM
7DD0                                 ; Start of ULCBAS
                                       main body
```

Load ULCBAS in the normal manner, using the POKEs in the green sheet that came with your lowercase kit. When you get back to Ready after activating ULCBAS, load T-Bug or another monitor that doesn't load above 7D00. Use the monitor to enter the routine above, changing the addresses if necessary. Then punch a tape of both routines, with the entry point at the start of this patch.

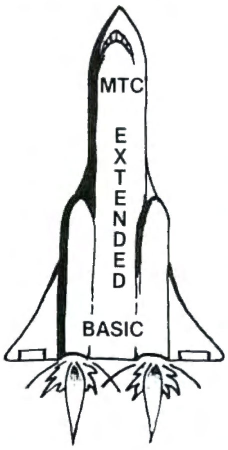
If you don't use any other routines, the memory size is 32192 for 16K, 48576 for 32K or 64960 for 48K. Joe's printer routine should be loaded 575 bytes lower than normal, changing the size accordingly. Just answer the

Continued on page 18

META TECHNOLOGIES

26111 Brush Avenue. Euclid Ohio 44132
CALL TOLL FREE 1-800-321-3552 TO ORDER
IN OHIO. call (216) 289-7500 (COLLECT)

TAKE YOUR MODEL II TO NEW HEIGHTS



MTC is proud to announce MTC EXTENDED BASIC for the Model II, by R. Ryen. Features include "fixes" to existing BASIC, multi-line functions, extending an existing sequential file, PEEK, POKE, greatly enhanced screen control and expanded editing capabilities. The contents of variables are NOT CHANGED when editing, deleting, inserting or merging lines, allowing continued program execution! All this and much more.

MTC EXTENDED BASIC................ $ 99.95

80 – 20 = 129⁹⁵

THE ORIGINAL NEWDOS/80

WITH MTC QUE CARD

★ SPECIAL PRICE ★

SAVE $20 **$129.95**

Apparat's long-awaited successor to NEWDOS+ is here! This is not an enhanced version of NEWDOS, but a completely new product. Simplified DOS commands can be instantly executed from BASIC, even within a program, without disturbing the resident code. System options, such as password protection, number and type of disk drives, BREAK key enable/disable and lowercase modification recognition, can be quickly and easily changed. Five new random-access file types allow record lengths of up to 4096 bytes, and no FIELDing! A powerful CHAIN facility allows keyboard INPUTs to be read from a disk file. An improved RENUMBER facility permits groups of statements to be relocated within program code. Diskettes may even be designated as RUN-ONLY! Features all NEWDOS+ utilities (SUPERZAP 3.0, etc.) and much more! One MTC technical staff member said having NEWDOS/80 is "better than sex" (you'll have to judge for yourself!). Includes 180-page instruction manual and MTC QUE card.

NEWDOS/80 # SPECIAL # $129.95
MTC QUE Card only .... $ 7.50

CALL REGARDING OUR NEWDOS+ UPGRADE PRICING

MODEL III VERSION

Has all the features of the Model I version plus enhancements. Allows any mix of single- or dual-sided 40- or 80-track disk drives. Most BASIC and many machine code programs written for the Model I will run without modification. Includes a utility for converting Model I single density to Model III double density.

NEWDOS/80 for Model III.. $149.95

CALL FOR AVAILABILITY

MICROPARAPHERNALIA

DISKETTES (box of ten)

5¼" PLAIN JANE™ $21.95
5¼" PLAIN JANE™ *Gold* $25.95
5¼" DATALIFE™ MD 525-01 .. $26.95
8" DATALIFE™ FD34-8000 .... $43.95

NEWDOS by APPARAT

NEWDOS/80 by Apparat $149.95
NEWDOS+ to
NEWDOS/80 UPGRADE ..... CALL
NEWDOS+ with ALL UTILITIES
35-track $69.95
40-track $79.95

BOOKS

TRS-80™ DISK
AND OTHER MYSTERIES .. $19.95
MICROSOFT™ BASIC DECODED $29.95
1001 THINGS TO DO WITH YOUR
PERSONAL COMPUTER .... $ 7.95

Michael Shrayer's

ELECTRIC PENCIL VERSION II

for
Model I and Model III

An expanded version of the critically acclaimed original word processing system! Includes all features of Version I plus many new extensions. Runs under most disk operating systems, has improved video text handling, loads any ASCII file for editing (including BASIC files), single sheet mode for printing on letterhead and more! Simple to use, features 2-key commands. An incredible package at an incredible price!

SPECIFY MODEL I OR III

Disk Version $79.95
Tape Version $69.95

CALL FOR AVAILABILITY

Complete for Model I with all utilities Plus exclusive MTC QUE card!

NEWDOS +

$69⁹⁵ by Apparat

Includes REF, RENUM, SUPERZAP, EDITOR/ASSEM., DISASSEM., DIRCHECK, and more! This is the original NEWDOS with all of Apparat's utility programs. Includes exclusive MTC QUE (Quick User Education) card.

40-Track Version $79.95
MTC QUE Card only .... $ 1.50

MORE PRODUCTS

MOST ORDERS SHIPPED WITHIN ONE BUSINESS DAY

Products damaged in transit will be exchanged.

PRICES IN EFFECT Sept. 1, 1981 THRU September 30, 1981, Prices, Specifications, and Offerings subject to change without notice.
8109

WE ACCEPT
- VISA
- MASTER CHARGE
- CHECKS
- MONEY ORDERS
- C.O.D.

- Add $3.00 for shipping & handling
- $3.00 EXTRA for C.O.D.
- Ohio residents add 5½% sales tax.

80 DEBUg

Super Debug

"Supermaze" (June 1981, page 160), on lines 430, 550 and 560, the IF statements are apparently written in the old Level I format with ANDs as *s and ORs as + s. Change line 430 to:

```
430IF (X+(E*Y)=A(100))AND D=4) OR
(X+(E*Y)=A(101) and D=2) THEN 500
```

and change the others in the same manner.

Narciso Jaramillo III
198 Belmont Ave.
Rialto, CA 92376

```
1CLS:INPUT"STARTING ADDRESS (XXXXXD)";S:INPUT"ENDING ADDRESS (XXXXXD)";E
5 D=255:FOR L=S TO E:POKE L,D:IF PEEK(L)<>D THEN 15 ELSE NEXT L
10 PRINT"TEST COMPLETED, NO ERRORS":STOP
15 PRINT  L;" SHOULD'VE BEEN 255, BUT IT CONTAINS";PEEK(L):STOP
```

Program Listing 1.

```
1 CLS:INPUT"ENTER RAM ADDRESS(XXXXXD)"; L :D=1
5 POKE L,O:IF PEEK(L)<>0 THEN PRINT"BYTE";L;" WON'T CLEAR TO ZERO, IT CONTAINS";
PEEK(L):STOP
10 FORX=1 TO 8:POKE L,D:IF PEEK(L)<>D THEN 20
15 D=D*2:IF D>128 THEN PRINT"TEST COMPLETED, NO ERRORS":STOP
17 NEXTX
20 PRINT"ERROR AT";L;" SHOULD'VE CONTAINED";D;" BUT IT CONTAINED";PEEK(L);
"INSTEAD":STOP
```

Program Listing 3

Missing Line

Program Listing 1 of my article in the April issue (page 259) omits line 83. Line 83 should read:

```
83 N=N+S:GOTO 63
```

Several readers have been kind enough to point this out to me.

James H. Fox
14601 55th St. South
Afton, MN 55001

Memory Bugs

I have found several misprints in my article "Memories are Made of This" published in the May edition. The corrections follow.

Note that if memory above location 32767 is to be tested, L in the programs will need to be equated to ' – 1 * (65536 – desired location to be tested). See the Level II Basic Reference Manual for more information about this.

One more error was found in the RAMTST source code listing at line number 320. The Op-code should have been 13 and not 35 as was printed.

Robert D. Randall
841 South 50th St.
Lincoln, NB 68510

Color Correction

We published a scrambled program listing in the June issue ("The Color Computer—An Inside Look," p.202). The program that should have appeared in Listing 1 follows:

```
1 REM MEMORY PEEK ROUTINE
2 REM ROBERT F. NICHOLAS
3 REM 9/28/80
4 REM
10 CLS
12 H$="0123456789ABCDEF"
15 FORX=1TO4:READH(X):NEXT
17 DATA 1,16,256,4096
20 PRINT"    THIS PROGRAM LOOKS AT YOUR"
30 PRINT"COMPUTER'S MEMORY AND RETURNS
40 PRINT"THE FOLLOWING INFORMATION:
50 PRINT
60 PRINT"        MEMORY LOCATION
80 PRINT"        VALUE THERE
90 PRINT"        CHR$(THERE)
95 PRINT"   (VALUES IN DECIMAL OR HEX)"
100 PRINT
110 HE$="D":INPUT"STARTING ADDRESS             (PREF
    ACE WITH 'H' IF IN HEX)";LN$:IFLN$=""THENCLS:GOTO1
    10
111 IFLEFT$(LN$,1)<>"H"THENLN=VAL(LN$):GOTO120ELSELN$=R
    IGHT$(LN$,LEN(LN$)-1)
112 LN=0:LE=LEN(LN$):FORX=1TOLE
113 L$=MID$(LN$,LE+1-X,1)
114 FORZ=0TO15:IFMID$(H$,Z+1,1)=L$THEN116
115 NEXT:GOTO100
116 LN=LN+Z*H(X)
117 NEXTX
118 HE$="H"
120 IFLN>65535THENCLS:GOTO110
```

Continued on page 19

DATA RESOURCES CORP.

QUALITY SOFTWARE FOR YOUR TRS-80® DISK SYSTEM

TO ORDER CALL TOLL FREE 800-525-7958

BIONIC BASIC

The **BIONIC BASIC** routines implement a number of useful commands and functions that are not available in regular BASIC. **BIONIC BASIC** appends these routines to the BASIC/CMD file so that they load automatically every time Disk BASIC is entered. The **BIONIC BASIC** modules will not require reserved memory, they are resident in memory whenever Disk BASIC is being used.

RESTORE data statements for reuse indivdually
MOVE a block of data in memory from place to place
SHIFT a string to all upper or lower case.
FLASH will turn a flashing cursor ON or OFF.
REPEAT function of the keyboard ON or OFF.
LOAD a BASIC program without losing variables.
GOSUB statements execute at computed lines.
SEARCH a program for occurrences of a string.
LINES of a program can be moved to new locations.
COMPRESS spaces, remarks from a program.
SORT any array in ascending or descending order.
REDIMension any array or list of arrays.
LOOKUP occurrences within arrays of special data.
STARTER PACKAGE....................... $49.95
COMPLETE BIONIC BASIC $74.95

NEWDOS/80 Version 2.0
for MODEL I and III $149.00

The hottest Disk Operating System is now available in its latest version. This is the ONE from Apparat, Inc., the people whose systems have made the TRS-80 a reliable computer. This system includes:

- Double Density support on the Model I
- New Superzap with expanded capabilities
- Diskette compatibility betweeen Model I and III
- Repeat Key and Repeat DOS 'Command' functions
- Single step execution for BASIC programs
- Enhanced Chaining with many new features
- Merging BASIC program overlays while running
- Disassembler dump of source code to disk files
- The famous NEWDOS/80 Utilities and Superzap
- New manual with examples and user explanations

MODEL I DOUBLE DENSITY PACKAGE
NEWDOS/80 Version 2.0 and the LNDOUBLER

Everything you need to convert your TRS-80 Model I to run double density. Complete with software, hardware, and instructions, — installs in minutes with **NO SOLDERING, WIRING OR CUTTING ... $229.95**
LNDOUBLER Alone $149.95

THE MICRO CLINIC DIAGNOSTICS

Computer "professionals" have long known the importance of regular use of diagnostic software in verifying the integrity of computer hardware. The TRS-80 is no exception; good diagnostics are a must in any situation where valuable data files are maintained. The **MICRO CLINIC** offers two programs designed to thoroughly check out the two most trouble prone section of the TRS-80 – the disk system (controller and drives) and the memory arrays.

The **FLOPPY DISK DIAGNOSTIC** is the most complete diagnostic program available for the TRS-80 disk system, and could be the most important program in your software library. This program provides a positive means for thoroughly checking out the disk controller circuits and disk drives. Included is a test of the accuracy of the drive motor speed, allowing adjustment. From one to four drives can be tested at the same time, 35, 40, 77 or 80 tracks, single or double density with the **LNDOUBLER** or **PERCOM DOUBLER.**

The **MEMORY DIAGNOSTIC** is broken down into two basic tests, the conventional write/read test and the M-1 Worm test. The **WRITE/READ TEST** portion of the diagnostic performs a comprehensive check on the ability of each memory location to store and retain data. The **M-1 WORM TEST** is the most critical in regard to memory timing and "worms" its way through memory.
MODEL I.......................... $24.95
MODEL III $29.95

M T C AIDS – III

The best of Meta Technologies family of data management systems. This easy to use system allows even a novice TRS-80 user to create data files custom configured for many applications, from lists for mailing and cataloging to financial reporting and time scheduling. Easy to customize without programming, it is complete with features for adding, deleting, sorting, updating and printing. The flexibility will make it one of your most useful software purchases. Included FREE is the MTC Aids Print Subsystem (MAPS), a powerful report generator.

MODEL I $69.95
MODEL II.................................. $99.95

M T C MERGE – III

Combines up to 14 AIDS data files into a single file. Duplicates may optionally be purged, and sorted order of records is maintained. Machine code for speed makes this useful if large data files are to be created from smaller ones.
Model I $19.95
Model II $29.95

M T C CALCS III

Handles numeric calculation of data contained in AIDS files. Ideal for reports on financial applications. Data from more than one file may be analyzed.
Model I $24.95
Model II $39.95

DATA RESOURCES SPECIAL AIDS – III SUPER SYSTEM

AIDS - III + CALCS - III + MERGE - III
SAVE * * buy all three at one low price
Model I .... $99.95 **Model II $149.95**

THE MICRO ACCOUNTING SYSTEM

This is the finest accounting system available for your TRS-80. Ideal for small business, this Integrated General Ledger, Accounts Receiveable, Payable System includes a Checkbook manager.
We offer something no other general business package can give you - a money back guarantee. If you are not totally satisfied return the package within 60 days and receive a full refund. Write for sample reports and full descriptions.
MAS/80 complete $489.00

DISKETTES

Soft-Sectored 5¼" 40 Track Double density
Replacement GUARANTEED for 1 year
We will replace at no charge ANY diskette returned to us

VERBATIM DATA LIFE.. $ 23.95
GENERIC in a plastic box .... $ 21.95
Box of 100 .... $200.00

FLIPSORT

The DISKETTE FILE BOX
that stores more than 50 diskettes
dust free and protected

5¼"...................... $19.95 **8"...................... $24.95**
purchase a 5¼" FLIPSORT with 50 GENERIC disks
SPECIAL DISCOUNT PRICE.................................. $119.95

MASTERCARD
AND VISA
WELCOME
We also accept
- COD • CHECKS
- MONEY ORDERS

✓540

ALL SOFTWARE SUPPLIED ON DISKETTE. ORDERS IN U.S.A. SHIPPED U.P.S. **FREE.** Colorado residents add 6½% sales tax.

ATTN: DEALERS - We offer attractive pricing and term to small computer retailers. Write for information on our dealer program and discounts in small quantities.

Data Resources Corp.
Business & Professional Center
8000 East Quincy Ave.
Denver, Colorado 80237 (303) 773-6665

Prices in effect Sept. 1, 1981 to Sept. 30, 1981 and subject to changes

✓See List of Advertisers on page 354

80 INPUT

Continued from page 12.

If you could take a survey, I think you might find that the majority of TRS-80Cs in the field are being used for game playing, probably with ROM-paks, and for this, they are very fast and very well adapted. Maybe that's why you're not getting feedback from the readers.

Bruce Dingwall
Mississauga, Ontario
Canada

Music Made Easy

The article "The Color Computer—An Inside Look" was certainly a boon to all Color Computer owners who, as I am, are going through the "almost no software" syndrome at this early stage in the machine's life.

Those few CC owners with whom I have talked await this sort of article, because it gives them something with which to play. Needless to say, Messrs. Martel and Nicholas gave many of us hours of enjoyment.

In particular, the "Music-Music-Music" program was great. However, it does not address the problem of the programmer who wants music within another program. Considering the overhead required to run "Music-Music-Music," it doesn't make sense to include it as a part of another program.

The answer is to try something else—a program that converts the notation in "Music-Music-Music" to sound commands, and then the writing of a simple subroutine to play the music.

To make this all work, simply load "Music-Music-Music" and enter the notes as instructed. Then, save the notes to tape using the program.

The next step is to add listing 1:

```
FOR X=1 TO N
PRINT#-2, N(P(X)),D(X)
NEXT
```

You can assign any line numbers you want above 5030. Next, load the notes back into the Color Computer using the "Music-Music-Music" program. Enter GOTOnnnn, where nnnn is the line number of the short program in listing 1.

Your printer will then print out two columns of numbers, the first being the note number, the second the duration of the note. Finally, to make the notes run in a program, use the format below and you will have music imbedded in your program. The example given might be for music in a horse race program. Run it and you'll see what I mean.

```
10 FOR X=1 TO 750
20 READ A,B
30 IF A= -1 THEN END
40 IF A=0 THEN FOR PU=1 TO B*15:NEXT:
   GOSUB200 :GOTO60
50 SOUND A,B
60 NEXT
200 CLS:RETURN
500 DATA 170,4,170,8,170,8,147,8,159,4,170,4,176,6,
   170,2,176,4,193,4,185,8,0,8,-1,0
```

Line 30 tests for the end of the music. Your last item pair in the data list should be " -1,0". Line 40 takes care of the rests. Using Sound does not allow use of rests, as does Extended Color Basic's Play.

Larwence C. Falk
Prospect, KY

T-Bug for Model III

I have a TRS-80 Model III and have wanted to program it in machine language. Radio Shack does not supply any software to aid me in doing this—until now that is. I have come up with patches for Radio Shack's T-Bug and Editor/Assembler so that they will work on the Model III. I was aided by a friend who has a Model I TRS-80 and supplied me with a copy of ZBug.

T-Bug can be used to patch itself. As it comes cassette I/O does not work. The fol-

80 AID

Continued from page 14.

memory size question with the proper value and load the new driver.

While we're on the subject, does anyone have a schematic of the 26-1104 lowercase modification? I long ago gave up trying to trace the new wiring.

Ed Ellers
3000 Beaumont Rd.
Louisville, KY 40205

Sound Program

I own a 16K Level II TRS-80, and I'm tired of seeing sound programs for assembly language users. Non-assembly language and non-disk users can have sound too! Try this program:

```
10 CLS : PRINT "PUSH ANY KEY FOR SOUND"
20 I$= INKEY$ : IF I$ = "" THEN 20 ELSE 30
30 FOR T = 1 TO 120 : OUT 255,0 : OUT 255,1
40 OUT 255,3 : OUT 255,4 : NEXT T : GOTO 10
```

I use this sound as a subroutine in many of my games.

I'm also tired of my friends who push the Break key and list my programs. So I hit them with this surprise:

```
10 POKE 16396, 23
```

The above line can be placed anywhere in a program. If you want your Break back try:

```
10 POKE 16396, 22.
```

Randy Long
4791 Torida Way
Yorba Linda, CA 92686

BBouncce Aidde

Regarding keyboard bounce there is a permanent cure without using software such as KBFIX. You must carefully remove all of the keys from their electrical contacts on the keyboard (I used paper clips bent into the shape of hooks). Then spray each contact with a small amount of WD-40. This method appears to be effective and safe.

James E. Richey, M.D.
Johnstown, PA

POKE 16445

Both in Benjamin Junge's letter in "80 inputs" and in Ian R. Sinclair's *Into the 80's* there is mention of POKEing 16445 with eight to get 32CPL and 0 to get 64CPL. When I tried to incorporate that idea into titles using "PRINT @" statements instead of "PRINT TAB (X)" as Mr. Sinclair does, I get an automatic 64CPL during execution of a loop. Why?

James L. Palmer
Rt. 1 Box 153
Bybee, TN 37713

lowing patches will fix this problem.

location	previous hex	new hex
47C5	2100FB18	CD F801C9
47CA	21	C9
4643	CD CA 47 CD 62 47 FE A5 20 F9	CD 9602 00 00 00 00 00 00 00
4782	C5 06 08 CD	CD 35 02 C9
46EF	CD 8C 47	00 00 00
46F8	CD 8C 47	CD 8702
478C	D9 OE 08 57	CD 64 02 C9

The patches are basically calls to the ROM routines. The new T-Bug can be punched out with the P command. Just enter P 4380 4824 43AO T-Bug. Low or high cassette baud rate can be selected by first changing location 4211 to zero or one as documented in the TRS-80 Model III manual.

The Editor/Assembler can be patched by T-Bug relocated to high memory or by first moving the Editor/Assembler to high memory. I took the second approach. Set memory size to 25856 (6500H). Key a block move program using T-Bug at location 32590 (7F50H).

Block move program

```
01 46 1A    LD BC, 1A46    ; length of program
11 00 65    LD DE, 6500    ; to address
21 00 43    LD HL, 4300    ; from address
ED BO       LD IR          ; move
C3 19 1A    JP 1A 19       ; jump to Basic ready
```

Load the Editor/Assembler but don't execute it. Instead enter /32590. This will move the editor/assembler from 4300H to 6500H. Load T-Bug and enter the following patches:

location	previous hex	new hex
667A	40	60
6537	E5 21 00 FB	CD F801C9
653D	E5	C9
655D	C5 E5 06 08	CD 35 02 C9
6589	E5 C5 D5 F5	CD 64 02 C9
65A9	CD 3D 43 06	CD 87 02 C9
65B8	E5 CD 3D 43	CD 96 02 C9
6818	05 46	49 00 keyboard I/O
6859	05 46	2B00 keyboard I/O
6959	0A 46	3300 Display I/O
6555	0F 46	3B00 Printer I/O
6928		C3 8D 43
658D		FE 43 2006 "C" Command Patches CD 42 30 C3 DA 46 21 A2 47 C3 2B 47

All the locations are for the relocated program. If you are using a monitor routine in high memory only the MSB changes. Location 6500H is 4300H, and so on. If the program were now punched it would be reloaded back into low memory. At location 7F50 key in the following block move program:

```
01 46 1A    LD BC, 1A46    ; length
11 00 43    DE, 4300       ; to address
21 00 65    HL, 6500       ; from address
ED BO       LDIR
C3 8A 46    JP 468A        ; Jump to Editor/
                             Assembler
```

Now punch the assembler using the T-Bug command P 6500 7F5F 7F50 EDTASM. The patches basically replace all of the I/O routines with the ROM routines. Since I don't have a printer the printer patch is untested. I added one new command to the assembler. When "C" Enter is typed the cassette baud rate prompt routine is called. Your program's source can be loaded or written at low or high cassette rates. However, low baud rate must be used for the object code (systems tape) produced by the assembler.

One additional note. ZBug (January, 1981, *80 Microcomputing*) is a great program. For Model III users I would suggest one change. On listing 1, line 70 change the instruction from "SUB 30" to "SUB 3C." The Model III has ROM routines that extend above 3000H.

Harold Zbiegien
Maple Hts., OH

80 DEBUg

Continued from page 16.

```
200 CLS
210 PRINT"   AFTER READING WHAT IS DIS-
220 PRINT"PLAYED ON THE SCREEN, PRESS
230 PRINT"THE ';' KEY TO SCROLL FORWARD"
232 PRINT"OR THE '-' KEY TO SCROLL BACK-
234 PRINT"WARDS."
236 PRINT"   PRESS 'H' TO CHANGE TO HEX  OR 'D' TO SWI
    TCH TO DECIMAL."
240 PRINT
242 PRINT"   TO CHANGE START ADDRESS,":PRINT"PRESS 'CLE
    AR'."
250 PRINT@483,"PRESS ';' TO PROCEED.";
260 GOSUB2000
1000 CLS
1010 P=LN
1012 FORP1=0TO1
1014 FORP2=0TO15
1016 Q=P+16*P1+P2
1018 IFQ>65535 THEN 1090
1019 PRINT@P1*16+P2*32,"";:IFHE$="H"THENJ=Q:GOSUB3000EL
    SEPRINTQ;
1021 PRINT@P1*16+7+P2*32,"";
1022 X=PEEK(Q)
1024 IFHE$="H"THENJ=X:GOSUB3050ELSEPRINTX;
1029 PRINT@P1*16+12+P2*32,"";
1030 IFX>32THENPRINTCHR$(X);
1050 NEXTP2,P1
1090 GOSUB2000:CLS
1100 IFIN$=";"THENP=P+32:GOTO1150
1105 IFIN$="H"ORIN$="D"THEN1150
1120 IFIN$="-"THENP=P-32ELSECLS:GOTO110
1150 IFP<0 THEN P=0
1160 IFP>65535THENP=65525
1170 GOTO1012
1200 END
2000 REM PAUSE ROUTINE
2010 IN$=""
2020 IN$=INKEY$:IFIN$=""THEN2020
2022 IFIN$="H"THENHE$="H":RETURN
2023 IFIN$="D"THENHE$="D":RETURN
2025 IFASC(IN$)=12 THENCLS:GOTO110
2030 IFIN$=";"ORIN$="-"THENRETURNELSE2010
3000 A=INT(J/4096)
3010 GOSUB3500
3020 J=J-4096*A
```

Continued on page 20

DeBug for Fast Clock

Thanks to Allan J. Domuret for his modifications to run DOS at a higher CPU speed ("Fast Clock DOS," April, 1981). As a user of NEWDOS I was quite pleased with the workings of the given patches.

Unfortunately Allan used a RAM area for his second SYSO/SYS patch that interferes with Debug usage. Since I am using Debug quite a lot, I had to find another hole in the operating system where I could place his patch. And—Eureka—I found one.

There is a seven-byte wide hole in disk memory starting at 4479H that exactly fits the patch. Using it I now have both the fast clock speed (by the way 3.54 MHz) and Debug in my system.

Make the following changes to SYSO/SYS (NEWDOS only):

0008A9—change to read C3 A2 46 E3 E3 E3 E3 3A...
000952—change C3 E3 45 to read C3 79 44
001759—change 02 02 00 4E to read 01 09 79 44 E3 E3 E3 E3 C3 E3 45 02 02 00 4E
011403—change from 5D to 68

Manfred Peters
D 55110 Alsdorf
West Germany

Shift Lock

Martin Hambel's "Shift Lock" utility in the May issue was just what I needed. I had previously installed the lowercase mod by Steven Wexler (*Kilobaud Microcomputing*, April, 1980) and after correcting a typo (the instruction at 7FFAH should be D2 7D 04) it worked fine, except for the usual problem—shift for lowercase—which is happily corrected by Shift Lock.

The lowercase mod requires a short software patch for the video driver which I merged with Shift Lock's keyboard driver patch. The only bug I found in Shift Lock was the use of 1AH as the control character. In the new Level II ROMs, shift—down arrow (1AH) is used to access control codes from the keyboard and will not work with this program. I substituted shift—up arrow (1BH) at location 7FE6 and have had no problems.

Burt Cohen
New York, NY

Satisfied Prosoft User

It is a cruel fact of life that there must be weeks to months of lead time in the periodical publishing business and sometimes a grievous injustice is done to an unwitting victim. A classic example of this was vividly visited on Prosoft's "Subedit/Subscript" by Chuck Teslar ("Word Processors Compared," June 1981) when in the interim along came his improved full screen editor version christened "Newscript." Newscript has it all in spades plus, and please take note, purchaser support!

After attempts to obtain technical assistance with myriad programs I very tentatively called Mr. Teslar at home as his documentation suggested. His enthusiastic response to my novice queries were not only courteous and patient, but he sincerely requested that I call him at any time and he wished more of his purchasers would! He explained that if he were ever not available to ask his 17-year-old son Glenn or his wife Debbie, both programmers of impressive credentials. (Apparently nobody ever told Glenn that programming was difficult.)

It is indeed fortunate for the micro word-processing public that Mr. Teslar abandoned a long career with IBM to favor us with a thoroughly professional and polished word processing system. One that totally supports all enhancements that all the major line/daisywheel printers have to offer. It produces a product that is camera ready of type-set quality and will compare with results achieved by systems costing as much as *eight times* my investment in a Model I/Centronics-737! It came as little surprise to me that the program was written full Model III compatible when I added that machine at my place of business!

I hope for an updated review now which will probably appear just as Prosoft and Chuck Teslar come out with an updated 50,000 word vocabulary "Newscript"!

A large bouquet of red roses are in order for the family *that cares* in this society of "sell 'em and forget 'em"!!

Anthony Toulis
Clewiston, FL

Spanky and the Gang

Your article on the computerized complaint letter in the May issue had interesting prospects so here I go.

My name is Spanky and I seem to be having trouble with the program listings in your magazine. Yours is not the only one to be sure, but it is entitled "The magazine for TRS-80 users," which is what I am (actually not a user, just a TRS-80). The typos are bad enough, but when it comes to the use of incorrect variables and DIM statements it requires hours of debugging by my owner before I can run the program correctly. In addition to the out-and-out errors, the programs are poorly written, requiring me to go through lines and lines of If statements where a simple looping sequence would do the job. Of course I realize that the listings are not written by professionals, but I would think small modifications by your staff would be beneficial to your readers.

I am not the only one experiencing this difficulty. My friends Snoopy, Lucy, Applesauce and Big Bird are also having the same problems.

Your magazine is good, but let's just have a little quality assurance in the programs.

Spanky
Virginia Beach, VA

Authors, Take Note

80 Microcomputing authors are asked to mail in corrections *with* their author's proofs, not before. Galley proofs are mailed of all accepted manuscripts.

Thank You.—The Editors.

80 DEBUg

Continued from page 19.

```
3030 A=INT(J/256):GOSUB3500
3040 J=J-256*A
3050 A=INT(J/16):GOSUB3500
3060 A=J-16*A:GOSUB3500
3070 RETURN
3500 PRINTMID$(H$,A+1,1);:RETURN
```

THE ALPHA I/O SYSTEM
a complete failure?

THE INSIDE STORY

It happened 3 years ago, when our President made a decision. At the time we specialized in custom analog and digital circuit design. The decision was to attempt to develop a line of standard interface hardware for the emerging microcomputers. At the time (1977) we had to decide which of the new machines could become the "industry standard" of the low cost micros.

Despite a few aggravating but minor deficiencies, the TRS-80 seemed tohave the most chance of success and it had the best price/performance ratio. Also, with some imagination, their large sales organization could become the largest service network in the world, a reassuring throught for the many novices in this new field.

It became clear that the TRS-80 could be used (with our then hypothetical system) to solve problems in many fields where computers were not yet used, mostly because of their high cost.

The IDEA was simple! ALPHA PRODUCT would supply the missing link between the TRS-80 and the "outside world", (more about this "outside world" later)

Early Survival

DANGER! If Radio-Shack entered the same market, we probably would not have survived, but the expectation was that they would be too busy developing their basic line (drives, printers, modem etc.). Thanks to our more specialized products, we would not be competing with them.

BAD START! We began with a failure. Our first product was supposed to be a simple, low cost, general purpose device. It would allow the TRS-80 to accept inputs other than the keyboard. Many kinds of external devices (the "outside world" mentioned before) like photocells, sensors, thermostats, switches, contacts, etc., could be connected easily. In addition, there were two relays to control (on or off) external loads such as motors, lamps, appliances, heaters, etc., etc.. In other words, it would allow the computer to interact or interface with external devices. We called it the INTERFACER 2. What a mistake! It sounded too much like "expansion interface". Many enthusiastic TRS-80 users called thinking that our "INTERFACER 2" was a low cost Expansion Interface (at $85 that would have been a real bargain!). We wanted to change the confusing name. That meant reprinting the manual, changing the ad, scrapping the flyers, discarding the silk screened cases. Well, "INTERFACER 2" it would stay.

TROUBLE! We also found that the majority of TRS-80 users were AFRAID of the hardware. They could be very comfortable with fancy programming but thought you had to be a computer specialist or technically inclined to put the INTERFACER 2 to work. In truth, some IMAGINATION and a SCREWDRIVER is all you really need. Anyone able to wire a switch could use this device.

WORSE! There was also the fear of plugging a "foreign device" into the precious computer. This notion has all but disappeared as there are now so many quality products designed for the TRS-80 that plugging in a non Radio-Shack device has become common.

Our ad in Creative Computing (80-Microcomputing did not yet exist) hardly paid for itself. We had a decision to make. Were we wrong or just too early? Our first INTERFACER 2 was sold to someone who wanted to, and succeeded in, controlling his fancy model railroad with his TRS-80. Interesting, but what made us stick with the concept was that some of our INTERFACERS began finding use in applications with fascinating possibilities. Space is lacking to describe them, but the most exciting was the successful use of the system in assisting a handicapped young boy. We were pleased to hear of such a meaningful application.

Today

Three years later, as you can see in our ads. The INTERFACER 2 is alive and well. The price went up a bit, and despite the introduction of the more powerful INTERFACER 80, the sales have been steady.

Then came the least understood product! the ANALOG 80. This $139, nicely designed module is an Analog to Digital converter with 8 input channels. Used with your TRS-80, it provides a powerful "data acquisition system". This jargon simply means that you can monitor, measure and record 8 independant varying voltages. Very few people realized its real power. Such a system would have cost over ten thousand dollars just a few years ago.

The possibilities in scientific and engineering environments are endless. This system could replace chart recorders, digital data recorders, programmable calculators, data analyzers and many other specialized and expensive pieces of equipment. Furthermore, up to 8 ANALOG 80's could be used simultaneously for a total of 64 channels of analog input! They simply plug into the TRS-80 using our "X" series of bus extenders (EXPANDABUS).

The idea was simple. We would supply the missing link between the TRS-80 and the "outside world". . . . .

Our next product was to be a second generation, Input/Output interface, with more flexibility than the INTERFACER 2. Careful design and refinement yielded the INTERFACER 80, the most powerful real world interface on the market today. It has 8 inputs, each optically-isolated and 8 outputs, each with a relay contact. The INTERFACER 80 is fully compatible with our ANALOG 80, allowing these to be used together in order to create systems that control external devices based on "sensed" input under control of the TRS-80.

A FAILURE! in spite of our extensive advertising, very few are aware of the existence of the powerful ALPHA I/O SYSTEM.

The Facts Are:

- The ALPHA SYSTEM/TRS-80 combination forms an incredibly versatile and powerful tool for acquisition/processing/control.
- In spite of its moderate cost, the system is sophisticated and reliable.
- The entire system can be easily programmed in BASIC using INP(X) and OUT X,Y commands.
- The modular approach and our EXPANDABUS allow for instant expansion as requirements demand.

The following pages contain more information about the devices mentioned here. We invite you to call or write to discuss your particular application.

Device descriptions; NEXT PAGE ➡

TIMEDATE 80

Neat, Compact Design 3 Years Battery Life — Slips Inside E/I (Y Option Shown) — Real Time Without Expansion Interface

WHY LOSE PRECIOUS TIME ?

- Complete, self contained "true" real time clock/calendar, TIMEDATE 80 continues to keep accurate time and date when the computer is turned off or experiences a power failure.
- **TIMEDATE 80** only needs to be set once, and it's two replaceable "AAA" batteries (not included) keep TIMEDATE 80 running in excess of 3 years. Costly Ni-Cad batteries and charging circuits are eliminated.
- The instant power is applied to the TRS-80, TIMEDATE 80 provides MO/DATE/YR, DAY of WEEK, HR:MIN:SEC and AM/PM information with quartz accuracy.
- **TIMEDATE 80** replaces the computer's internal clock. Extremely useful for automatic operation of remote systems with no operator in attendance. If the power fails and then is restored, only TIMEDATE 80 will update the system with current TIME and DATE information, an impossibility with the computer's internal clock.
- **TIMEDATE 80** is quartz crystal based with INTELLIGENT CALENDAR, including provisions for leap year! TIME display may be by 12 hour AM/PM or by 24 hour military and Eruopean format.
- **TIMEDATE 80** plugs directly into the rear of the TRS-80 keyboard and gives the "TIME$" function even without an Expansion Interface. For those with a disk system, it plugs into the left side panel of the Expansion Interface. An optional "Y" connector can provide for further expansion.
- **TIMEDATE 80's** small size keeps the computer table uncluttered. If you have an Expansion Interface, TIMEDATE 80 literally "DISAPPEARS" by slipping into the empty space in the bottom of the interface.
- Two sets of software, on cassette, come with TIMEDATE 80—"TIMESET" and "TIME$". "TIMESET" is a step by step set of simple instructions for setting TIMEDATE 80. "TIME$" is a set of poke routines which patch DOS and Level II TIME$ to read TIMEDATE 80 and is easily incorporated into any user software. "TIME$" will always print the time and date when LISTING a program—great for keeping track of revisions!
- Other valuable uses for TIMEDATE 80 are: accurate date and time information for business reports like payroll records, financial reports, etc., or to various I/O devices requiring 24 hour clock input, such as laboratory instrumentation, and to communication systems needing "Log In/Log Out" data (bulletin boards).
- **TIMEDATE 80,** fully assembled and tested, 90 day warranty, complete with instructions and software on cassette, $95.00. "Y" option, add $12.00

NEW: Computer to Computer ordering:(212) 441-3755 (24 hr. data line)

ALPHA Products

85-71 79th Street, Woodhaven, NY 11421

CALL
(212) 296-5916
FOR FAST SERVICE

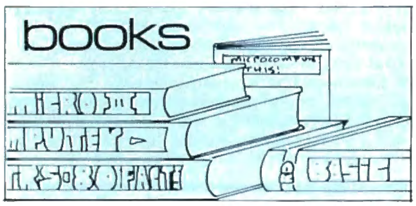
ADD $2.50 PER ORDER FOR SHIPPING AND HANDLING
ALL ORDERS SHIPPED FIRST CLASS MAIL
WE ACCEPT VISA, MASTER CHARGE, CHECKS, M.O.
COD ADD $2.00 EXTRA
QUANTITY DISCOUNTS AVAILABLE
N.Y. RESIDENTS ADD SALES TAX

✓210

80 REVIEWS

Edited by Michael E. Nadeau

"It seemed as if the author was sitting next to me as I read Inside Basic Games."



Inside Basic Games
Richard Mateosian
SYBEX, Inc.
Berkeley, CA
Software, 325 pp.
$13.95

by Bonnie Welsh

Inside Basic Games teaches interactive Basic programming through the use of games. The author strives for and succeeds in writing "not just a collection of games, but an educational book that will help the reader learn to design Basic programs." This book does, in fact, provide many examples of program design and development.

Richard Mateosian is to be commended on the clear and concise style in which he presents his material. Each chapter begins with the objective to be tackled, along with the steps of progression to be used in attaining this goal. If, perchance, you have forgotten what you were supposed to learn, he provides you with a summary at the end of each chapter.

Free Basic

One of his most interesting and useful concepts is the use of Free Basic, also known as GOTO-less programming. Its principal goal is to free the programmer from having to think about line numbers, although it does support structural program techniques. Actually, Free Basic is a form of pseudocode, but is much more precise in that it uses Basic instruction rather than verbal instruction. Once you have constructed your program using Free Basic, Mateosian explains the techniques used in translating it into Basic.

Each game is described per se and then in terms of their Basic and Free Basic instructions. The effects and advantages of certain routines in regard to their "cannibalization" and "generalization" abilities are discussed. Your main objective is to learn how to write routines that can easily be integrated into other programs.

General programming techniques such as the top-down development theory are explained, along with more specific techniques like space saving and file maintenance functions for a certain program.

The games are written in Microsoft Basic and can run on the TRS-80, Apple II, and PET/CBM. He identifies the small changes in each program that must be made when converting from one system to another.

It seemed as if the author was sitting next to me as I read *Inside Basic Games*, due to his easy-going personal style. I recommend this book to anyone interested in writing their own interactive programs. ■

Musical Applications of Microcomputing
Hal Chamberlin
Hayden Book Company
Hardcover, 650 pp.
$24.95

by Dennis Bathory Kitsz

The true musical application of microprocessors is not the topic of Hal Chamberlin's book. Instead, this is a compendium of the sonic rather than musical uses of these devices. In fact, only three measures of music are presented within its more than 650 pages, and even these contain errors in notation and transcription!

A Professional's Book

The true contents of this book cover the many methods of producing sound through electronic means. Both traditional analog and developmental digital concepts are presented; uses of microprocessors are introduced; and the analysis, development, and reproduction of natural and originally conceived sounds are discussed. In many ways, this is a professional's book; cursory attention is given to the basics of acoustics and electronics, but these seem included by obligation rather than enthusiasm. Furthermore, the occasional discussions of music itself are amateurish and extremely shallow.

Hal Chamberlin is recognized as one of the leading and most imaginative figures in the field of digital processing of sound, and one expects that this book would clearly demonstrate his wide-ranging skills. Perhaps it is an inevitable consequence that this wealth of knowledge cannot be satisfactorily compiled in one volume which has both popular and professional pretenses.

Nevertheless, *Musical Applications of Microprocessors* is destined to become one of the definitive books on digital creation and reproduction of sound, and digital interfacing of analog synthesizers. It contains sample circuitry, Assembly programs (mostly for the 6502), waveform diagrams, and drawings.

The book is divided into three sections: Background, Computer-Controlled Analog Synthesis, and Digital Synthesis and Sound Modification. Chamberlin reveals himself to be an expert engineer/programmer, but the sections on the goals of music synthesis and its history are superficial. By contrast, the section describing the parameters of sound is clear and thorough.

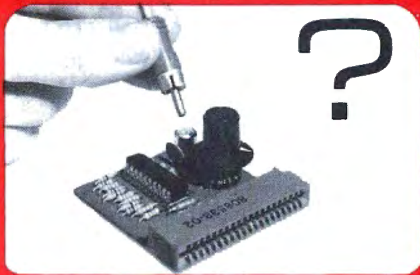
In fact, throughout the book Chamberlin is at his best when describing the details of circuitry or programming, and is weakest when talking about general principles and concepts. Sound Modification Methods (chapter 2) begins with a brief discussion of the aural effects of modifying known sounds, and moves directly to describing these modifications by means of waveform and frequency spectrum graphs. The following chapter explains the uses and application of voltage control in music synthesis.

The heart of the book begins with

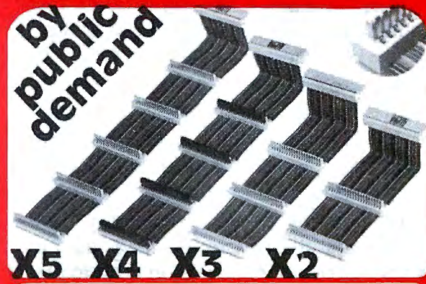
NOW MODEL I AND MODEL III!

Now Model III users can take advantage of the ALPHA I/O system too. Our new MOD III/I BUS CONVERTER allows most port based Model I accessories (such as our ANALOG-80, INTERFACER 2 and INTERFACER-80) to connect to the Model III bus. MOD III/I BUS CONVERTER, complete with all connectors, only $39.95.

STICK-80 MAKES KEYBOARD OBSOLETE.
Features the famous ATARI Joystick. 8 directions + fire control. Simple instructions to make joystick versions of most action games. Plugs into keyboard or expansion int... Price includes ATARI joystick with ALPHA interface and instructions. FREE "MAGIC ARTIST" program.....$39.95
Super Real Time Action Graphic Sound games for Stick-80 by Software Innovations: ALIEN INVASION, COSMIC INTRUDERS, BREAKOUT Each $9.95
STELLAR ADVENTURE super action with sound $14.95
Software authors and distributors contact us for joystick conversion package for your existing games



MUSIC-80 MUSIC-80 MUSIC-80 MUSIC-80 MUSIC-80
Use existing software or write your own. With this low cost 8 bit digital to analog converter you can synthesize up to 5 music voices. Built-in volume control handy when stereo not near TRS-80. Simply plug the "MUSIC-80" into the keyboard or the E/I screen printer port and connect the output (RCA jack) to any amplifier. The Radio-Shack $12 speaker/amplifier works fine.
Fully assembled and tested, 90 day warranty $39.95



YOU ASKED FOR IT: "EXPANDABUS" X1, X2, X3 AND X4.
CONNECT ALL YOUR TRS-80 DEVICES SIMULTANEOUSLY on the 40 pin TRS-80 bus. Any device that normally plugs into the keyboard edge connector will also plug into the "EXPANDABUS". The "X4" is shown with protective covers (included). The TRS-80 keyboard contains the bus drivers (74LS367) for up to 20 devices, more than you will ever need. Using the E/I, it plugs either between KB and E/I or in the Screen Printer port. Professional quality, gold plated contacts. Computer grade 40 conductor ribbon cable.
X2...$29. X3...$44. X4...$59. X5...$74.
Custom configurations are also available, call us.

ANALOG-80: A WORLD OF NEW APPLICATIONS POSSIBLE.
8 DIGITAL MULTIMETERS PLUGGED INTO YOUR TRS-80!!!
Measure Temperature, Voltage, Current, Light, Pressure, etc.
Very easy to use: for example, let's read input channel #4: 10 OUT 0,4 'Selects input #4 and also starts the conversion 20 A = INP(0) 'Puts the result in variable "A" Voila!
Specifications: Input range: 0-5V to 0-500V. Each channel can be set to a different scale.
Resolution: 20mV (on 5V range). Accuracy: 8 bits (.5%) Port Address: jumper selectable. Plugs into keyboard bus or E/I (screen printer port). Assembled and tested. 90 day warranty.
Complete with power supply, connector, manual .... $139.

INTERFACER 2: LOW COST INPUT/OUTPUT MODULE.
Still the best value in sense/control devices. Use it for energy control, burglar alarm, darkroom, selectric drive, model trains, robots, Skinner box...
—8 latched TTL outputs. 2 relays SPDT 2A, 125V. contacts
—8 TTL/CMOS inputs. Input 0 and 1 are optically isolated.
—Neat and compact design, very easy to use:
10 A = INP(0) 'Reads the 8 inputs (if A = 0: all inputs are low) 20 OUT 0,X 'Controls the outputs and the relays
Assembled & tested, 90 day warranty. Price includes power supply, cable to KB or E/I, superb user's manual, free phone dialer program: $95. Manual only: $5.

GREEN SCREEN WARNING

IBM and all the "biggies" are using green screen monitors. Its advantages are now widely advertised. We feel that every TRS-80 user should enjoy the benefits it provides. But **WARNING:** all Green Screens are not created equal. Here is what we found:

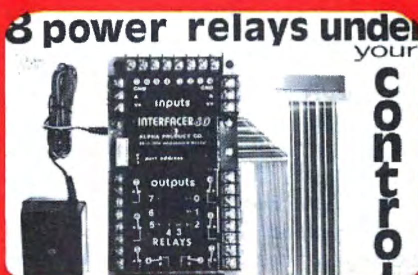
- Several are just a flat piece of standard colored Lucite. The green tint was not made for this purpose and is judged by many to be too dark. Increasing the brightness control will result in a fuzzy display.
- Some are simply a piece of thin plastic film taped onto a cardboard frame. The color is satisfactory but the wobbly film gives it a poor appearance.
- One "optical filter" is in fact plain acrylic sheeting.
- False claim: A few pretend to "reduce glare". In fact, their flat and shiny surfaces (both film and Lucite type) ADD their own reflections to the screen.
- A few laughs: One ad claims to "reduce screen contrast". Sorry gentleman but it's just the opposite. One of the Green Screen's major benefits is to increase the contrast between the text and the background.
- Drawbacks: Most are using adhesive strips to fasten their screen to the monitor. This method makes it awkward to remove for necessary periodical cleaning. All (except ours) are flat. Light pens will not work reliably because of the big gap between the screen and the tube.

Many companies have been manufacturing video filters for years. We are not the first (some think they are), but we have done our homework and we think we manufacture the best Green Screen. Here is why:

- It fits right onto the picture tube like a skin because it is the only **CURVED** screen **MOLDED** exactly to the picture tube curvature. It is Cut precisely to cover the exposed area of the picture tube. The fit is such that the static electricity is sufficient to keep it in place! We also include some invisible reusable tape for a more secure fastening.
- The filter material that we use is just right, not too dark nor too light. The result is a really eye pleasing display.

We are so sure that you will never take your Green screen off that we offer an unconditional money-back guaranty: try our Green Screen for 14 days. If for any reason you are not delighted with it, return it for a prompt refund.
A last word: We think that companies, like ours, who are selling mainly by mail should •list their street address•have a phone number (for questions and orders)•accept CODs, not every one likes to send checks to a PO box•offer the convenience of charging their purchase to major credit cards.
How come we are the only green screen people doing it?
Order your **ALPHA GREEN SCREEN** today ..$12.50

LET THE "CHAIN BREAKER" FREE YOUR MINI-DRIVES.
End the daisy-chain mess once and for all. Fits all mini-drives: Percom, Aerocomp, Shugart, Micropolis, MTI, Vista, Pertec, Siemens, BASF. Easy to install: just remove the drive cover, plug in the "CHAIN BREAKER" and replace the cover. Voila!!!
Now you can change and move your drives around without disassembly. Keep the cover on and keep the dust out. High reliability gold plated contacts, computer grade 34 conductor cable. Tested and guaranteed.
Get one for each drive.....only........ $13.95



INTERFACER-80: the most powerful Sense/Control module.
- 8 industrial grade relays, single pole double throw isolated contacts: 2 Amp @ 125 Volts. TTL latched outputs are also accessible to drive external solid state relays.
- 8 convenient LEDs constantly display the relay states. Simple "OUT" commands (in basic) control the 8 relays.
- 8 optically-isolated inputs for easy direct interfacing to external switches, photocells, keypads, sensors, etc. Simple "INP" commands read the status of the 8 inputs.

Selectable port address. Clean, compact enclosed design. Assembled, tested, 90 days warranty. Price includes power supply, cable, connector, superb user's manual........ $159.

✓210

ALPHA Product Co.

ADD $2.50 PER ORDER FOR SHIPPING AND HANDLING
ALL ORDERS SHIPPED FIRST CLASS MAIL.
WE ACCEPT VISA, MASTER CHARGE, CHECKS, M.O.
COD: ADD $2.00 EXTRA.
QUANTITY DISCOUNTS AVAILABLE.
N.Y. RESIDENTS ADD SALES TAX.

85-71 79 St. WOODHAVEN N.Y. 11421 ☎ Info and order: (212) 296•5916 chapter 4, Direct Computer Synthesis Methods. After touching on the limitations of traditional voltage control, Chamberlin describes the methods by which sound is digitized, including a set of new problems introduced by this form of storage and generation. He also mentions the mathematics involved and the languages in use. Chapter 5 completes this introduction with discussion of microprocessor terminology, history, interfacing, and instruction sets. This chapter, crucial to understanding the application of and reasoning behind microprocessors as applied to music, is very weak in both explanation and example.

Section II, Computer-Controlled Analog Synthesis, is a refreshingly complete look at using computer techniques for controlling the available body of synthesizers. Beyond standard keyboard and oscillator selection, Chamberlin also includes such ideas as using CMOS analog switches as an intermediary solution in the design of analog machines. Complete circuits are presented for many of the most popular synthesizer modules (oscillators, amplifiers, filters), with all parts specified; this represents a change from books whose examples consist of skeletal, theoretical circuits. Input and output methods are explained, digital-to-analog and analog-to-digital concepts (including time considerations) are detailed, and keyboard interfacing is shown in various options. The first control programming examples appear in this section. Graphic display techniques are also introduced, but Chamberlin becomes vague and circuit diagrams and other hard examples disappear; perhaps this area is not his strength, and he properly leaves it general.

Finally, Section III—Digital Synthesis and Sound Modifications—presents Chamberlin's major contributions to the field of digital creation and reproduction of audio. Tone generation, filtering, and analog input/output examples carefully detail the problems (accuracy, distortion, and cost) of digital techniques, and he forgives little of the reader's potential lack of background. No musician or amateur computer enthusiast without substantial engineering, electronic, or acoustic background will be able to finish this volume.

Fast-Fourier transforms allow the conversion of waveforms to their frequency spectra, and many program examples have been provided in microcomputer literature but with no idea of their use. Chamberlin presents examples of these in Basic and Assembly, and puts them to work analyzing sound for compiliation and storage in digital form. The significance of the mathematics involved and the difficulty of its implementation in real time become clear in this section.

Finally, Chamberlin presents module-by-module replacement of the analog components of a music synthesis system with digital varieties. His approach is complex and comprehensive, arriving eventually at the description of a complete multi-voiced music development unit.

It may seem from the above summary that *Musical Applications of Microprocessors* would be an ideal addition to the computer user's library. Certainly it is a valuable contribution to the field, and will be used by anyone seriously interested in the electronic production of sound who is also highly qualified in electronic theory and application. But, just as certainly, the obvious significance of its contents will be a frustration to musicians and experimenters who wish to put its ideas to use.

Like his colleague Wayne Bateman (though he does not say so directly), Chamberlin assumes that contemporary musicians must inevitably come to an understanding of the details of any natural sound's creation in order to reproduce it compellingly, and in order to invent new, viable music. But Chamberlin's work, impressive though it is, remains unbendingly technical, with neither recognition of nor concern about the musical art at even its rudimentary levels.

For engineers, here is a bonanza of exciting ideas; for musicians, here is another in a series of frustrations. ■

Basic Scientific Subroutines, Vol. I
by F.R. Ruckdeschel
Byte Publications
Peterborough NH
Hardcover, 316 pp.
$19.95

by Bruce Powell Douglass

Basic Scientific Subroutines, Vol. I is a new book covering algorithms and Basic routines for plotting, complex number arithmetic, vector and matrix manipulation, series approximations of functions, and more. Volume II will deal with approximation, regression, interpolation, integration, root-seeking, and optimization.

The book is well written; it is unusually clear and concise. The mathematical techniques presented in the book are usually taught with so much theory that it is hard for the student interested in applications to use the techniques in real-life situations. This book is particularly worthwhile because the author spends time describing the physical interpretation and meaning of the techniques, along with the algorithms. This book is useful for students well versed in science but not in programming, or programmers who lack some math or scientific background. As a supplementary text, the book is excellent.

The routines are listed in two appendices, with driver programs both in North Star and Microsoft Basic. The book is not meant as a Basic primer, yet it does present methods for changing scientific algorithms into Basic routines.

The first section (chapter 2) deals with plotting routines. These are not extremely useful for the TRS-80, as they plot using byte resolution with asterisks for data points. Their potential is in plotting on a printer. Since printers seldom know about Set and Reset, byte resolution is often as good as one can get.

Chapter 3 deals with complex variables and contains a routine to convert rectillinear complex numbers to polar complex numbers. This is followed by routines to add, subtract, multiply and divide complex numbers.

The next chapter deals with vectors and matrices, and is, in my opinion, one of the two most useful chapters in the book (the other being the one on basic series approximations). Routines for eigenvalues, matrix arithmetic and matrix inversion are all given.

A useful topic nicely dealt with is the use of matrices for the rotation, transposition, and scaling of vertices of figures. Another is the routines to determine characteristic polynomials and eigenvalues (characteristic values) of matrices. These routines are primarily useful for scientists and engineers.

Basic Scientific Subroutines, Vol. I is a useful book for scientists, engineers, and students. With a few minor exceptions, the algorithms are excellent and the Basic routines well designed. Since the book gives the routines in a commented form, as well as compacted form in North Star and Microsoft Basic, the book can serve not only as a supplement to a computer text, but also as a sourcebook of scientific subroutines.

The book is well worth the investment. ■

GAME BREAKTHROUGH:

TO TRS-80

THE STICK 80

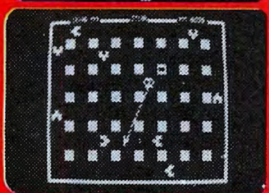
- ☑ FEATURES THE FAMOUS **ATARI JOYSTICK.**
- ☑ WORKS WITH ANY LEVEL 2 AND DISK SYSTEMS.
- ☑ PLUGS DIRECTLY INTO KB. OR E/I (EXPANSION PORT).
- ☑ INCLUDES SIMPLE, DETAILED INSTRUCTIONS TO MAKE JOYSTICK VERSIONS OF MOST ACTION GAMES.
- ☑ COMPATIBLE WITH **ANY** OTHER TRS-80 ACCESSORIES.
- ☑ FULLY ASSEMBLED AND TESTED. READY TO PLUG IN AND ENJOY.
- ☑ FUN TO EXPERIMENT WITH IN **BASIC:** SIMPLY USE INP(Ø).
- ☑ 90 DAY PART AND LABOR WARRANTY.

UNCONDITIONAL **MONEY BACK GUARANTEE.** Try your STICK-80 for 14 days. If for any reason you are not delighted with it, return it for a prompt and courteous refund (including shipping and handling). PRICE INCLUDES ATARI JOYSTICK + ALPHA INTERFACE + DETAILED INSTRUCTIONS + DEMO PROGRAM. ONLY $39.95

BIG FIVE SOFTWARE SUPER FAST MACHINE LANGUAGE ACTION GAMES

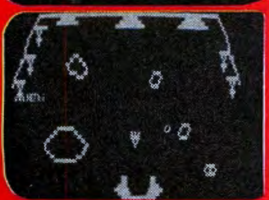
If you and your TRS-80 have longed for a fast-paced arcade-type game that is truly a challenge, then **SUPER NOVA** is what you've been waiting for. In this two player machine-language game, large asteroids float ominously around the screen. Suddenly your ship appears and you must destroy the asteroids before they destroy you! (But watch out because big asteroids break apart into little ones.) The controls that your ship will respond to are thrust, rotate, hyperspace, and fire. All right! You've done it! You've cleared away all the asteroids! But what is that saucer with the laser doing? Quick! You must destroy him fast because that guy's accurate! ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆ **SUPER NOVA®**

The sound of the klaxon is calling you! Cruel and crafty invaders have been spotted in battle formation warping toward Earth at an incredible speed. Suddenly, your ship materializes just below the huge flock of invaders. Quickly and skillfully you shift right and left as you carefully fire your lasers at them. But watch out! A few are breaking out of the convoy and flying straight at you! As the whine of their engines gets louder, you place your finger on the fire button knowing all too well that this shot must connect—or your mission will be permanently over! With sound effects! ◆◆◆◆◆◆◆◆◆◆◆◆ **GALAXY INVASION®**



Your TRS-80 screen has been transformed into a maze-like playfield for this game. As your ship appears on the bottom of the screen, eight alien ramships appear on the top. All of them are traveling at flank speed directly at you! Quickly and boldly you move toward them and fire missiles to destroy them. But the more aliens you destroy, the faster the remaining ones become. If you get too good you must endure the wrath of the keeper of the mazefield: the menacing "Flagship". You must destroy him fast because, as you will find out, that guy's accurate! With sound effects! ◆◆◆◆◆◆◆◆◆◆◆◆◆◆ **ATTACK FORCE®**

With thousands of stars whizzing by you, your **SPACE DESTROYER** ship comes out of hyperspace directly under a convoy of aliens. Almost effortlessly, you skillfully destroy every last one. But before you can congratulate yourself, another set appears. These seem to be slightly more intelligent than the first set. Quickly you eliminate all of them, too. But your fuel supply is rapidly diminishing. You must still destroy two more sets before you can dock with your space station. All right! The space station is now on your scanners! Oh no! Intruders have overtaken the station! You must skillfully fire your neutron lasers to eliminate the intruders from the station before your engines run out of fuel and explode! With sound! ◆◆◆◆◆◆◆◆◆◆◆◆◆ **COSMIC FIGHTER®**



NEW

The second **Big Bang** has occurred and the galaxy is full of stray asteroids and meteors. As you look through your space port you see a belt of asteroids drifting across the screen blocking your path to the safety of the space station above. But be careful because meteor showers, exploding suns and invading aliens may strike your ship and send it hurtling back to ground level. How many times can you and your opponent maneuver through those obstacles before time runs out? With sound effects! ◆◆◆◆◆ **METEOR MISSION II®**

SPECIAL

INTRODUCTORY OFFER:

Order the STICK-80 with one or more games and choose one:

•FREE ALPHA GREEN SCREEN
OR
•FREE MAGIC ARTIST PROGRAM

IN ADDITION YOU MAY:
ORDER 2 GAMES AND TAKE 10% OFF the games
ORDER 3 GAMES AND TAKE 15% OFF the games

All games are written in machine language and supplied on cassette.

EACH GAME IS AVAILABLE IN 2 VERSIONS

A (SIDE 1: LEVEL II 16K FOR MODEL 1; SIDE 2: LEVEL II 16K FOR MODEL 3) $15.95

B (SIDE 1: DISK 32K FOR MODEL 1; SIDE 2: LEVEL II 16K FOR MODEL 1) THE DISK VERSION SAVES THE HIGH SCORES ON DISK $17.95

• ALL THESE NEW JOYSTICK GAMES WILL ALSO WORK USING THE KEYBOARD WITHOUT ANY MODIFICATIONS.

GOOD NEWS: If you already have a non-joystick version of these BIG FIVE SOFTWARE games, send the original tape(s) with your STICK-80 order + $3 per game. We will send you a new Joystick version of your game(s).

ALSO AVAILABLE **DOUBLE-STICK 80:** 2 ATARI JOYSTICKS + ALPHA DUAL INTERFACE + INSTRUCTIONS + DEMO PROGRAM: $59.95

ALPHA Product Co.

✓210

ADD $2.50 PER ORDER FOR SHIPPING AND HANDLING
ALL ORDERS SHIPPED FIRST CLASS MAIL
WE ACCEPT VISA, MASTER CHARGE, CHECKS, M.O.
COD: ADD $2.00 EXTRA
QUANTITY DISCOUNTS AVAILABLE
N.Y. RESIDENTS ADD SALES TAX

85-71 79 St. WOODHAVEN N.Y. 11421 ☎ phone: (212) 296 5916

KGS-80 Keyboard Actuator
Kogyosha Company
NIK International Trading, Inc.
Tenafly, NJ
$599 with power supply, mounting hardware and manual.

by Ted I. Blumstein

Using your TRS-80 as a word processor can benefit your career, speed your professional growth, or improve your performance in school. However, the cost of buying a letter-quality printer has been the stumbling block that prevented me from word processing on my TRS-80 until recently.

I was pleased, therefore, to read about the KGS-80 Keyboard Actuator, a new peripheral from the Kogyosha Company, which turns an IBM Selectric or similar typewriter into a high-quality printer without modifying the typewriter.

The KGS-80 is impressive. It takes only minutes to position the unit on the keyboard, plug it into the expansion interface and make a few minor adjustments. The 46 plastic-tipped solenoids started printing all the Level II commands without error.

The keyboard actuator rests firmly on the typewriter. Two easily removed mounting pieces hook into the slit between the shift key and the typewriter frame. The two set screws on the KGS-80 then fit securely inside the lips on the mounting pieces.

The KGS-80 is designed to plug into the TRS-80 expansion interface, or directly into the CPU using the Radio Shack printer interface cable. Because the unit does not require software to operate, it was fully compatible with my Electric Pencil, and should work equally well with Scripsit or other text editing programs.

I have been using the KGS-80 for the past four months to prepare personal and business documents, and I am more than pleased with its performance.

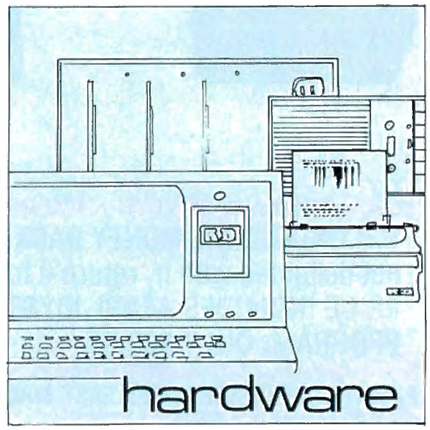
If a Selectric is not used, the KGS-80 needs to be realigned to the new keyboard, by removing the top cover and adjusting four set screws. The set screws position the tips of the solenoids to the proper height above the typewriter keys. A gap of about 0.02–0.08 inches seems to work best. At this setting, the solenoid is fully extended before striking the key and a snapping effect is achieved. The initial setting takes some time and experimentation, but once you get the technique down it is quite simple. After making adjustments you can lift the KGS-80 from the keyboard and replace it in a matter of seconds.

I have changed the speed of the original factory setting (10 characters per second, the reliable operating speed of most typewriters) by rotating controls VR2 and VR3 on the circuit board. VR2 sets the on time for the solenoid and VR3 affects the delay time. My experience shows the delay time setting is the most critical factor in determining operating speed. The speed can be adjusted to 20 characters per second, excluding the shift lock/release and carriage return. Control VR4 adjusts the delay time between the carriage return and the first letter of the next line, and is factory set for a standard 12-inch carriage. The delay time can be increased to accommodate the return of extra long carriages, however. All these adjustments can be made manually without the need for special software, which is a real plus, particularly if you have only 16K.

"All characters are printed in upper and lowercase."

All characters (including numbers and common symbols) are printed in upper and lowercase. With a special IBM typing element the ASCII signs can also be printed. Function keys include carriage return, space, case lock, shift lock and case shift release. The operative ASCII input code is 13, 32 through 92, 95, and 97 through 122. A line feed is automatically inserted with each carriage return.

"The 46 plastic-tipped solenoids started printing all the Level II commands without error."



The KGS-80 will not take the place of a heavy-duty line printer if continuous use is required. The solenoids are not designed to exceed 400 successive repetitions without other letters and symbols being activated. This is no problem with ordinary text and most program applications; however, if you wish to use it as a screen printer with very little on the video monitor, the successive activation of the space bar solenoid might cause it to overheat and lock up.

The lock-up characteristic is a safety feature of the Kogyosha solenoids to prevent burnout. In a matter of minutes, the cooled solenoid is again fully operational. Models that became available after September, 1980 have two solenoids instead of one which allows more continuous use of the space bar.

Another modification available is a buffer circuit which will allow the KGS-80 to print stored text while the computer is free to continue processing programs. This ought to be a good feature considering the comparatively slow rate of IBM Selectric printer mechanisms.

KGS-80 keyboard actuator models will soon be available for the Apple, Pet, Ohio Scientific and other microcomputers, with either parallel or serial ports. Service is available at Kogyosha's Tenafly address. Sales are handled by NIK International.

If you own a good electric typewriter, the KGS-80 is an excellent way to add a high-quality printer to your system at a very affordable price. ■

Reader Service for facing page ✓4.→

Regression II
Dynacomp
Pittsford, NY
$19.95 cassette
$23.95 disk

by Bruce Douglass

Dynacomp's Regression II is, as far as I know, the only non-linear regression program commercially presently available for any microcomputer. Non-linear regression is a complex, sophisticated technique. Probably because of this it has a smaller audience than it deserves. This program is one-dimensional, that is, it can handle one independent variable and one dependent variable.

There are three main approaches to non-linear regression: Taylor series approximation, steepest descent (gradient) method, and the Marquardt method which is an optimization of the previous two. The last is the most sophisticated algorithm and is used as the default algorithm in SAS (Statistical Analysis System) NLIN programs on mainframe computers.

What is done, essentially, is in a problem with N parameters. That is, when the fitting function to which you wish to regress the data has N unknown constants that you wish to determine, it finds the values of the parameters that gives the minimum sum of squares between the actual and predicted values of the dependent variable. That sounds worse than it actually is.

All that is meant is that we define a function $SS = \Sigma((Y'(i) - Y(i)) * (Y'(i) - Y(i)))$ called the Sum of Squares. Y' is the value predicted for the dependent variable for the i-th value of the independent variable by the fitting function. Y is the actual value that occurs with the i-th value of the independent variable (X(i)).

As you can see, the better the fit of the regression, the closer the predicted and actual values for Y will be. This will make SS smaller and smaller. There is a theoretical limit for how close you may come to predicting the actual Y value, and it is determined by the relationship of the form of the fitting function with what the real relationship is as well as the random chance variation (noise) that occurs in the data itself. In other words, some functions will fit the data better than others and, for any given function, how well it fits is dependent on the noise in the data.

Although there is a certain amount of evidence that for some classes of functions least squares methodology will result in non-homogenous error variance, least squares is most commonly used because of ease of both comprehension and implementation.

Although there is a certain amount of evidence that for some classes of functions, least-squares methodology will result in non-homogenous error variance, least squares is most commonly used because of ease of both comprehension and implementation.

The First Two Methods

The first two methods of non-linear regression have their problems. The first, Taylor series approximation, attempts to locate the optimal parameters in N-space by using a series expansion. This method has its downfalls. For example, if the initial estimates of the values of the parameters are not close enough to being correct the approximation may get lost and settle for a local minimum. However, its advantage is that when the initial estimates are close, it rapidly converges the optimal values.

"The program structure has some non-optimal characteristics."

The method of steepest descent looks around for the steepest slope (gradient) of the surface in the parameter space and follows it to the minimum values. This method is less likely to be fooled by local minimums, but, as it gets closer and closer to the actual values, the rate of convergence typically slows to a crawl.

The Dynacomp program uses the steepest-descent method, and, therefore, is reasonably stable in its convergence to the optimal parameters. It can take a long time to converge if you choose the wrong function, the data is noisy or there are a large number of parameters to be fitted.

Like most scientists, I have a streak of pedantry in my blood and would like to take this moment to point out to the people of Dynacomp that they are using the divergence of the vector in the parameter space, and not the gradient, when they add the partial derivatives of the parameters. The manual states otherwise.

The documentation that accompanies the program is mediocre. It does mention the method and algorithm used with a certain amount of completeness, but it is only moderately useful. In the space used it would have been even more educational (if that was their purpose) to delineate the routines used and their relationship to the algorithm.

The documentation mentions that there is one functional form that this program is not good at regressing, but it remains for the reader to discover what that is. It is only mentioned in one of the two examples given. It turns out that polynomial regression is not well regressed.

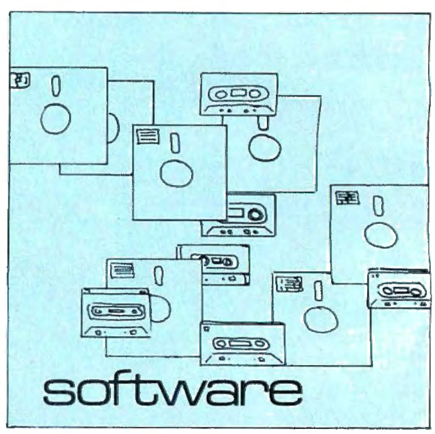
The examples are run with comments and cover everything that is in the program. Since the program asks the operator for all inputs, this merely serves to reinforce the user's understanding of operation. In this sense, it is useful.

The program structure has some non-optimal characteristics. It is available for several computers and looks as though it was written so that it could be easily used on a variety of computers rather than any one given computer. This is evident in the graphic routine, the lack of multi-statement lines and the filler lines that take up memory but don't contribute at all to the program.

Graphing Routines

Let's consider the graphing routines first. Data points are printed out on the screen using "*" for data. The graphs often scroll the screen, so the user does not get a good picture of the behavior of the function (which is one of the purposes of a graph.) The values of the data points may be printed on the screen at the user's option. On a TRS-80 with Set/Reset resolution, which is six times the resolution one gets with using "*" to represent data points, why use the latter? If you are outputting to a printer, it makes sense, but not to the screen.

True, you can control the height of the graphs but not directly, and often the graph scrolls to graph the data. I consider



that unacceptable.

Listing the program, you will notice an interesting program construction:

- No multi-statement lines.
- No Defint for looping variables.
- Up to six separate lines of Print statements followed by a CLS (clear screen command).

Since this is an iterative program, using multi-statement lines and Defint for loop variables would greatly reduce run-time.

Also, the Print statements (up to six separate lines) are followed by a CLS? If readability was the object, then one REM line is worth 10 times its weight in useless Print statements.

The program comes on both disk and tape. Being cheap, I bought the tape version and adapted it for sequential disk files. It runs nicely and even has the capability of editing the source file. Editing consists of changing, deleting or inserting data points. This routine is useful and works well. The program lists the data by page so that it will not scroll.

Before running the program, the function containing the parameters must be entered into the program. When running the program you will be asked the number of data points in the data set. This is not a good approach since it requires you to remember the number of data points in your data set. If it is a disk/tape file, all the user has to do is keep the number of data points as the first element of the file.

The user is asked whether the data is to be loaded from disk. If so, it will be loaded. If not, the user is expected to enter the data from keyboard.

When you get done with that, you will be asked if you would like to graph the data using the routine mentioned previously. Eventually you do get to the program. (There are a couple of useless timing loops that I deleted.)

At this point you will be asked for the number of parameters, the convergence criterion, maximum number of iterations, the step constant and the initial estimates of the parameters. This part is good since it is nice to be able to specify the number of iterations and the convergence criterion rather than depend on theirs.

Finally, results are displayed. The standard deviation of the regression and the number of iterations used are displayed along with the determined values of the parameters. You are asked if you would like to graph the function. It asks you if you would like to run the regression again, choosing different initial estimates of the parameters, in the event that the convergence was not to your liking.

This last option is useful, but, unfortunately, the jump is back farther than that, so you have to wade through reviewing the data set and other garbage before finally being asked for the new starting estimates of the parameters. This is easily corrected, but it really should be correct in the first place.

In conclusion, Dynacomp Regression II (or Parafit) is a useful, powerful, non-linear regression program, despite its less-than-optimal implementation. While correct in its mathematical usage, many small, minor adjustments need to be made to the program. Even so, it is the only program of its kind currently available, and is well worth the asking price. ■

C-BUG
Monitor Tape $29.95
Monitor ROM $39.95
Disassembler $49.95
The Micro Works
Del Mar, CA

by Howard Berenbon

The Micro Works is offering two outstanding utility programs for the TRS-80 Color Computer. They allow access to the machine at the Assembly-level.

C-Bug

The monitor is a powerful 6809 monitor, and is available on cassette. The two versions are virtually identical, except that the ROM version mounts into a Radio Shack ROM pack, allowing instant access (mounting instructions are included). The documentation is quite complete, including loading, running, and operating instructions. Also included is a documented Assembly listing of the C-Bug monitor. monitor.

Commands

The cassette version loads using the CLOADM command, and will run by typing EXEC. There are 19 commands, as follows:

G —Go back, Return to Basic
R —Display 6809 register list
M —Memory examine and change. Entering a hex address will display a line of eight bytes. The cursor may be moved up, down, left, or right to display more memory.
I —Insert hex to memory
T —Transfer block of memory
J —Jump to machine language subroutine
C —Change register list
S —Save to cassette
B —Set baud rate for comm link
L —Load hex
$ —Convert hex to decimal
. —Convert decimal to hex
P —Move display page; any 512 bytes of RAM may be displayed.
U —Upload and send data to comm link
D —Download, and receive data from comm link
! —Take over software interrupt
AU—Auto mode: To use your computer as an intelligent terminal connected to a host system. Commands are entered from the comm link.
X —Terminal mode: To use your computer as a terminal operating at 110 baud.
* —Reset causes the computer to be reset and returned to Basic.

These commands are completely documented in the manual, with examples of their use.

Disassembler

The disassembler is a program that will allow you to look into the Basic ROM, and is available on cassette requiring 16K or RAM to run.

The program is well documented, including a complete Assembly listing, instructions with sample outputs, and information on connecting your Color Computer to a printer. Also included is a complete list of useful ROM entry points, with their apparent functions, a list of interesting variable addresses that the ROM uses (like 0019 start of user RAM, etc.), and finally a memory map of the system.

Running the Disassembler

The program is loaded using the CLOADM command, and will run by typing EXEC. You will be prompted for a series of parameters, starting with the start address for disassembly. To disassemble the complete ROM, just type Enter to all prompted questions.

Features of the disassembler include: cross-referencing of variables and labels, output code that can be reassembled, output to an 80-column printer (highly recommended) or 32-column printer, output to your video display, and a data table area specification which defaults to the table boundaries in the interpreter ROM.

Both programs are highly recommended. ■

Cocobug Debugging Monitor
For the TRS-80 Color Computers
Allen Gelder Software
San Francisco, CA
$19.95

by Dennis Bathory Kitsz

By the time you read this, it's likely there will be a dozen machine-code monitors for the Radio Shack Color Computer. However, none will have a catchy name like Cocobug, nor its unique appeal. This program is a machine-code monitor provided on tape, complete with documentation, a quick-reference programming card, and a 32-page Motorola data sheet including timing diagrams and the full 6809 instruction set.

Typical of software by Allen Gelder, Cocobug is oriented towards machine-code programmers. This is an approach I find very appealing, because Assembly programs (which consist of mnemonic replacements for actual byte-for-byte binary coding) can easily obscure the real binary operations which the microprocessor is undertaking.

Part of Gelder's own description reads, "With Cocobug you may examine RAM and ROM in hexadecimal, ASCII or mixed hex and ASCII form...the powerful MC6809E CPU is made available in a pair of 6809 Programming Models that depict the CPU features at entry and exit (via a restorable Breakpoint) of your machine code string. Byte entry and Breakpointing, plus the ability to direct real-time program flow, are made easy and natural through line-entry of addresses, bytes and certain control characters."

The author's curiously formal description belies the ease of using the program. Cocobug itself consists of two parts, a Basic menu/prompting program, and a machine-code subroutine for display of memory and execution and breakpointing of the user's program. After a very brief CLOAD and CLOADM (the Color Computer equivalent of the Model I's System command), the program is run. A cursor appears at the top of the screen waiting for a user directive.

Memory display options are M and N, followed by a hexadecimal address. The M command displays the contents of 112 bytes, 14 lines of eight rows of hex bytes. Fifty-six bytes in hex and ASCII are displayed under the N command. The small number of bytes displayable on the screen is a function of the Color Computer's 32-character lines. However, the screen may be scrolled up or down eight bytes, or moved a single byte ahead or back.

Most interesting are the programming models included in Cocobug. All the registers (X and Y, plus the two accumulators) are displayed, as well as the top of the two stacks and the condition code register (known to Model I programmers as the flags). Under Cocobug's R command, two programming models are displayed: The first is the entry condition (before execution of the user's code), and the second is the exit condition (after a breakpoint). This display is disabled with the T command.

Line entry is a machine programmer's dream. The L command begins the process. Here's an example from the documentation: L(0D00)BF043039(0D10*8E4849 BD0D00<. This short line commands Cocobug to begin loading memory at 0D00 hex with BF 04 30 39, then move ahead to 0D10 hex and load 8E 48 49 BD 0D 00. The execution address (marked by the asterisk) begins at 0D10 hex, and a breakpoint has been placed after the return from the subroutine beginning at 0D00 hex. It's quite a bit of work for only 35 keystrokes. An Assembly equivalent would involve more typing:

```
0D00            00100   ORG   0D00
0D00  BF0430    00110   STX   0430
0D03  39        00120   RET
0D10            00130   ORG   0D10
0D10  8E4849    00140   LDX   #4849
0D13  BD0D00    00150   JSR   0D00
```

Gelder does allow Assembly enthusiasts some comfort, however. Comments may be inserted between bytes or addresses by using single quotation marks, such as L(0D00) 'STX 0430' BF 04 30. Incidentally, use of the ">" character from the Cocobug command mode restores the conditions before any breakpoints created during line entry.

Rumors of monitor programs as good or better than Cocobug are already being heard, but software from Allen Gelder (including Basic semi-compiler for the Model I, Accel) is unique. Like Accel, for which no royalties are demanded when used to compile and distribute programs, Cocobug is provided with complete instructions on where it resides in memory, what RAM patch points it uses, and how to make backup copies. Furthermore, Cocobug is fully supported by the author, and is the first in a group of modules (Gelder calls them a "suite of 6809 debugging and programming aids") that will include an assembler and other tools. Cocobug alone, however, is a capable and attractive program at a very reasonable price. ■

Disk Storage: It's No Place to Compromise.

A dependable disk system means everything to the high-performance operation of your computer. At Percom, we know this. And we do something about it.

For example, the connectors on our TFD drive systems are gold plated.

Why? Because solder-plated contacts oxidize, forming an insidious insulation that can cause erratic operation. And then total failure.

Gold-plated connectors are just one way we make better drive systems than the competition.

Now #1 for the Model III

Quality Percom TFD drives provide more features, cost less. Add-on drives start at only $439.00. Complete First-Drive Systems start at only $749.95. Features: **"Flippy" Capability** — Record your programs and data on either side of a diskette. **Greater Storage Capacity** — Store 180 Kbytes — formatted — on one side of a 40-track TFD drive. Store over twice as much on a TFD 80-track drive. **Think Megabytes** — Your TFD drive controller will handle up to four drives. You can access almost 1.5 million bytes of on-line program and data files. **Either Add-in or Add-On** — Your TFD First Drive System can be **either** internal (add-in) or external (add-on). In either case, you get a complete system including the disk controller, drive, power supplies and disk-operating software. **BASIC-Language DOS** — Included on diskette with each First Drive System is Percom's OS-80/III™. This easy-to-use disk-operating system loads into 7 Kbytes of RAM and then frees the drive for other use. With OS-80™ programs, you can have full, read-write interchangeability between Model I and Model III diskettes. **Or Model III TRSDOS*** — TFD 40-track drives work with Tandy's Model III TRSDOS, without modification. For 80-track operation, TRSDOS can be easily modified with diskette patches supplied.

Still #1 for the Model I

Besides greater storage capacities, more quality control measures and lower prices, all Percom Model I drives are rated for double-density operation.

Plug a DOUBLER™ in your Expansion Interface and enjoy the same double-density disk storge capacity as Model III owners. Included with each DOUBLER is a TRSDOS compatible double-density disk-operating system. DOUBLEZAP programs are available for upgrading other popular DOSs for DOUBLER operation. And our double-density version of OS-80™ costs just $49.95.

Of course you don't **have** to upgrade your Model I for double-density operation to use Percom disk drives. But it's nice to know you can.

Percom TFD drives for the TRS-80* Model I are available in 40-, 77- and 80-track versions, in 1-, 2- and 3-drive configurations. Prices start at $399.00.

System requirements: Model III TFD drives work with a 16-Kbyte system (min) and Model III BASIC. The initial drive must be a first-drive system. An optional interconnecting cable is available for expanding with external drives #3 and #4. Model I TFD drives work with a 16-Kbyte system (min) equipped with an Expansion Interface, Level II BASIC and DOS software, and an interconnecting cable. Two- and four-drive interconnecting cables are available from Percom.

To order products, or for the name of your authorized Percom retailer, call toll free:
1-800-527-1592.

PERCOM

PERCOM DATA COMPANY, INC.
211 N. KIRBY GARLAND, TEXAS 75042
(214) 272-3421

Yes . . . I'd like to know more about the best for my TRS-80 computer. Send me free literature about quality Percom products.

☐ Model I ☐ Model III

Send to
PERCOM DATA COMPANY, Inc., Dept. 8
211 N. Kirby St. Garland TX 75042

name ____________________

street ____________________

city ____________ state ________

zip ____________ phone number ____________

MAIL TODAY!

M01

™trademark of Percom Data Company, Inc.
*trademark of Tandy Radio Shack Corporation which has no relationship to Percom Data Company.

PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE.

Astro-Scope
Sex-O-Scope
AGS Software
Orleans, MA
$30 each, disk

by G. Michael Vose

Astrology is a big business, as seen by the number of newspaper columns and books devoted to the subject. It is also a serious business based on scientifically observable phenomena—the movement of planets, stars, moons, asteroids and other heavenly bodies. The positions of the stars and planets at the time of a person's birth are used by astrologers to make predictions and judgements about an individual's future. The logical connection between the position of the planets and the characteristics of a human being is a bit hazy in my mind, but millions of people believe that a valid connection exists.

Astrology and Computers

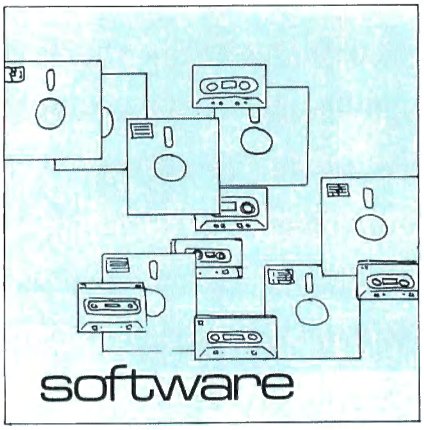
Professional astrologers are using microcomputers at an increasing rate to perform the calculations needed to cast horoscopes. Casting horoscopes by hand can take hours and hand-held electronic calculators simplify the process only a little. The use of microcomputers helps provide the information fast and can also provide greater quantities of it.

One of several companies that has been producing astrological software is Astro-Graphics Services/Software of Orleans, Massachusetts. They recently released two programs which add a new dimension to microcomputer astrology. The programs are called Astro-Scope and Sex-O-Scope. They both plot and display excellent natal event sky charts (using the pinwheel format) plus all the planetary aspects. More importantly, they also provide 1500 plus word descriptions of the person whose horoscope has been cast. This is a major advancement in astrological software.

The Sex-O-Scope program gives a reading of the subject's sexual proclivities and attitudes and is based on texts written by John Townley, author of *Planets In Love* and the editor of *Sexology Today*. The readings provided by the Astro-Scope program were written by Steve Blake and Robert Hand, authors and lecturers on the subject of astrology.

These programs can be used by anyone, which is what makes them unique. I've run other astrology programs that produced sky maps and aspect charts, but I didn't know how to interpret the information. These kinds of programs are designed for professional and serious amateur astrologers. The new offerings from AGS Software, however, can be great fun, even for someone who has no knowledge of astrology.

These programs, though only for the beginning astrologer, are complete and fun.



Once you enter the basic information—date, time and location of a person's birth—the computer will calculate and then display horoscope data and character descriptions. You can cast horoscopes for yourself, friends, family members or famous people.

Here are some excerpts from Astro-Scope and Sex-O-Scope.

Astro-Scope for Howard Cosell

Howard Cosell's ascendant is in Leo. If your ascendant is in Leo also some characteristics you may have are: You are proud, and like being the center of attention. You are strong, dominant, dignified and honorable. You like the power and privlege that comes with leadership. Others must have integrity to impress you. You like sumptuous surroundings, and have a regal bearing!

His sun is in Aries. If you're an Aries, too, you are: very energetic and high-spirited, independent and a risk taker. You are also an enthusiastic leader, but a reluctant follower. So blunt, direct and honest that at times you hurt other's feelings. Your strong competitive urge may alienate others. You are good natured and bold!

Sex-O-Scope for Linda Lovelace

Linda's ascendant is in Taurus which makes her earthy, slow at moving, but persistent once aroused, warm, friendly, loving, steady and strong.

I told you this could be fun! These are only excerpts—each of these readings took less than a half hour to peruse.

These programs are a mite expensive if entertainment is your aim (they don't have print routines although there are routines you can use to obtain printouts from Basic programs). AGS Software does offer versions that do provide printouts plus the license to reprint the material. These versions have to be special ordered and, at $200, are aimed at the professional astrologer.

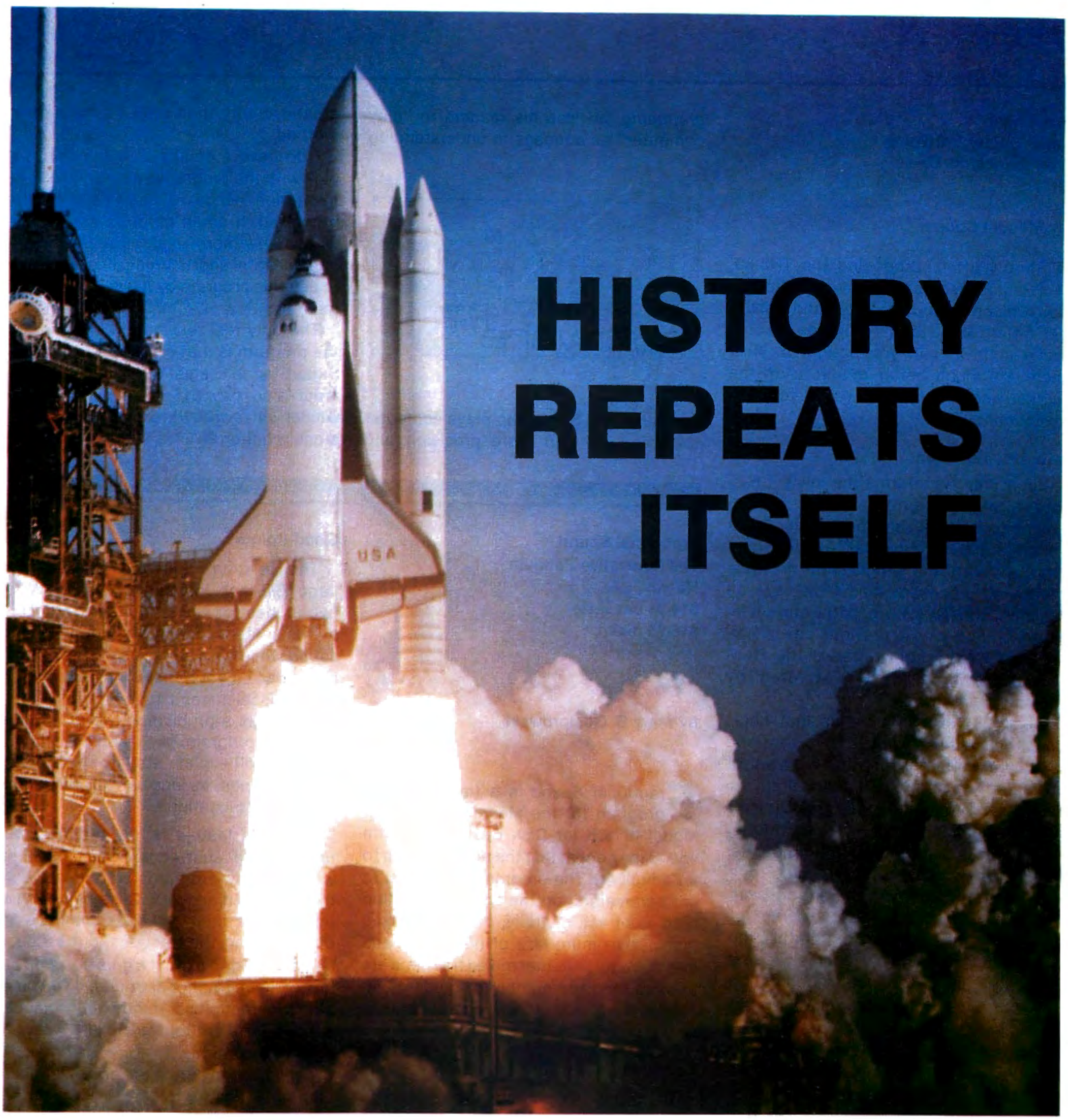
From Technocrat to Mystic

The text files that accompany the main program are quite large, so you'll need at least 32K and two disk drives to run these programs. The documentation booklets are some of the best I've seen, providing latitude and longitude figures for at least one city in every state of the U.S., blank charts that you can photocopy for your own use, and data on famous people that you can use to run their charts.

These programs might change your attitude toward astrology. You may see the economic potential of running horoscopes for your star-struck friends. Or you might want to start an astrology column in a local newspaper or magazine (once you've purchased the reprint rights from AGS Software.)

At the very least, you could use these programs to have some fun, and maybe even change your image from technocrat to mystic! ■

> *"The Sex-O-Scope program gives a reading of the subject's sexual proclivities and attitudes."*


HISTORY
REPEATS
ITSELF

SuperStep
Allen Gelder Software
San Francisco, CA
$19.95

by Mendel Cooper

Shortly after introducing the TRS-80, Radio Shack brought out a stripped-down machine-code monitor and debugger, T-Bug. At the time, T-Bug was the only game in town and you bought it. Now there are much better monitors on the market, and you may well regret your $15 investment.

Into a Heavyweight

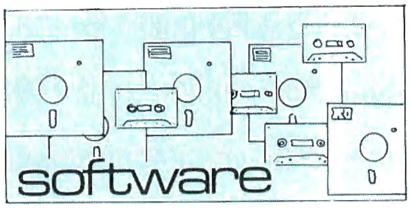
SuperStep transforms the lowly T-Bug into a heavyweight. It offers single-stepping, a two-speed trace mode, and a versatile disassembler that can run in conjunction with the single-step or trace modes. A processor model displays all registers and flags, both for the current instruction and for the previous one. An intelligent RAM window shows the RAM locations and contents affected by memory accesses.

SuperStep has commands that backspace in memory, clear the screen, and display memory contents in rows of 16—in hex or in ASCII. It gives direct access to the registers (each one labeled) and to the flags (each one individually set). There is also a relative-jump pointer that shows where you are jumping to—and can even take you there, and back. There is a high-speed save/load command included in the package. Regrettably, this one will not work on my machine because it has the XRX cassette load modification.

SuperStep's string editor allows inserting or deleting bytes in a sequence of code. The bytes on either side of the change move aside or close up, as necessary. Useful for editing ASCII strings or tables, this feature also allows creating or deleting space within a program. A patch can now have enough space to reside entirely within the program it modifies; it no longer need jump elsewhere to do its work (you would still need to fix up the Jumps and Calls, of course).

An Invaluable Aid

I rate SuperStep as the best single program I have in my modest collection. If the program is useful as a debugger, it is invaluable as an aid to learning Assembly language programming, and the Z-80 instruction set in particular. As the program's author states, SuperStep cuts down on the "imaginative overhead" of the programmer. William Barden, in his book *TRS-80 Assembly Language Programming,* advises his readers to "play computer" as a means to understanding program flow. SuperStep plays computer for you, and with more precision and greater clarity than the paper-and-pencil method.



I do have a couple of minor bones to pick with the program's author. The documentation needs to be revised and reorganized. It could be greatly expanded, with many more examples. But this is hardly a fatal shortcoming since you learn to use this program by using it and experimenting.

The only real reservation I have about the program is that it needs T-Bug to run. SuperStep is a steal at its price, but you must also pay $15 for T-Bug. Perhaps Gelder will rewrite the program as a stand-alone monitor.■

Fractional Sound
The Innovative Penguin
Harvey, LA
$14.96 cassette
$18.95 disk

by Mary S. Gasiorowski

Fractional Sound is a fractional drill program for the Model I Level II, or Model III 16K (or Disk Basic, 32K), with sound (by connecting the auxiliary plug to an amplifier). The program allows you to set parameters for: the type of operation (add, subtract, multiply, divide, or mixed types), maximum size of numerator and denominator, whether mixed fractions are allowed, and the number of students involved; it will also allow you to put in your own problems, if you want.

Bad Point

The format for putting in the answer is extremely awkward. You have to identify which of the eight positions each number of your answer goes into (two for the whole number, three for the numerator and three for the denominator). If you (and your kids) can get used to that, then the rest is easy.

Good Points

The program generates easy-to-read large (5/8 inch for numbers in fractions, two inches for whole numbers) numbers on the screen, and a random tone ticking sound while it's waiting, as well as a bugle charge when your answer is correct, and other miscellaneous sounds.

If you get a problem wrong (it informs you), the program goes into its waiting mode. It will explain step-by-step how to do the problem as long as you press a key each time to continue.

One nice feature is that a correct but unsimplified answer is not counted as wrong. Many programs will accept only the simplified answer, which can be frustrating for a child who has done the work of combining the fractions, but because he neglected to simplify, gets no credit for his work. In Fractional Sound, if you combine the fractions correctly but neglect to simplify, you are told to simplify your answer and continue.

On occasion during the drill, a graphics character (Marley Mole) appears to encourage you. He wants you to come out and play a game with him.

At the end of the drill is a little game—fence in the moving dot (Marley Mole) without getting trapped first. It's a nice little game, appropriate for the intended age level of Fractional Sound.■

"In Fractional Sound, if you combine the fractions correctly but neglect to simplify, you are told to simplify your answer and continue."

Put Your Fingers On The Pulse Of Wall Street With Your Radio Shack® Computer

If you invest in the market, operate your own business or require updated information quickly to manage your affairs, Dow Jones Information Services provides the authoritative business and financial information you need. . .from Wall Street, Washington and around the world. Now Radio Shack owners can receive this valuable service through TRS-80® Videotex® software.

Get Information For Business And Investment Control. . .On Command.

Discover how quickly and easily your personal computer turns into a specialized, sophisticated library of information with access to the Dow Jones database. It's considered the world's most reliable and comprehensive source of business and financial data providing you with news up-to-the-minute as well as from the previous 90 days.

Now you can review thousands of items from The Wall Street Journal, Barron's and the worldwide Dow Jones News Service without having to maintain piles of newspaper clippings. Information about companies, stocks, labor, government, international market and monetary developments, and so much more, is yours on command.

Quotes From Four Exchanges

Now, in just seconds, you can obtain current prices and the volume traded for stock quotes (15 minutes delayed) on over 6,000 companies. . .plus 50 key financial indicators on 3,200 selected firms and 180 industries. You'll also have access to trading information on mutual funds, treasury notes and bonds.

Imagine being able to spot trends, research a company, obtain data that could affect your business, your investments, your investment decision-making.

But you don't have to imagine anything. . .Dow Jones Information Services provides it all right now.

Low Start-Up Cost. . .Free Password.

Stop by your nearest Radio Shack store and find out how easy and inexpensive it is to get Dow Jones investment data through TRS-80 Videotex. Videotex software for Radio Shack computers is only $29.95. What's more, you get a Dow Jones password absolutely free. . .along with one hour of free introductory time. And there are no start-up or minimum usage charges. In addition, software and usage costs will be tax deductible in many cases (consult your tax advisor for further information).

Customer Support. . . Just A Phone Call Away.

Dow Jones' professional customer service staff is ready to provide complete information about this important home information service now. . . and full customer support after you've made your purchase. Just call toll free. . .

800-257-5114

In New Jersey call: (609) 452-1511.

TRS-80, Radio Shack, and Videotex are registered trademarks of Tandy Corp.

DOW JONES INFORMATION SERVICES

Providing Information Through Innovation

Dow Jones & Company, Inc. 425

Space Shuttle
Instant Software
Peterborough, NH
$14.95

by Jon Lindsay

When the space shuttle Columbia hurtled toward Earth, I experienced its shock wave as it descended over California's Big Sur to land at Edwards Air Force Base. So I waited for the arrival of a program named Space Shuttle, which simulates the flight of the Columbia from Earth orbit to a landing at Cape Canaveral. Why the landing was not simulated for Edwards Air Force Base is a mystery whose answer is known only to the program author.

To measure how well the operator performs during the simulation, score is kept of the pilot's performance during several phases of the flight and the score totaled after the simulation is over. I was impressed by the lack of emphasis on the game aspect. I would rather spend my time mastering the task being simulated than compete against the computer. This chance to assume the role of an astronaut will highlight some of the problems, monotony and excitement of guiding this revolutionary craft to a safe landing.

About the Program

The viewer is initially faced with three selections on a menu. The first selection is a flight from space to a landing at a spaceport. The second is called "long approach." You descend from about 100,000 feet to the spaceport, approximately 220 miles away. The final selection is a straight-in landing from about 28 miles out. The second and third options are really just portions of the full flight, isolated for the ease of adjusting the pilot to flying this "brick with a engine." True to life, you must "dead-stick" the Columbia—bring it in with almost no power. Not being an astrophysicist, I have no way of knowing how accurate the flight path algorithm is. But it appears to conform to some obvious rules. For example, the atmosphere extends to an altitude of about 58 miles. Since Columbia initiates its flight several hundred miles up, it will behave like a spacecraft above 58 miles and like an airplane below that figure. Such seems to be the case in the program. In space, only thrusters can be used to guide the craft; in the atmosphere, more conventional controls can be used.

There are four distinct phases to the simulation:

- In space—thrusters must be used to control the reentry configuration and temperature.
- After reentry but above 58 miles—loss of some thruster effectiveness, though they are still the only means of control.
- Below 58 miles—complete thruster ineffectiveness but craft can now pitch and roll as needed: Altitude is in miles, until 100,000 feet.
- Final approach—landing gear and speedbrakes can be used: Also, there is a yaw control for fine-tuning the approach.

Inside Columbia

You are presented on your flight screen with a primitive graphic view of the horizon, a set of controls and some sensing instruments. There is little animation through the split window of the shuttle, except for the horizon, which moves only up and down in a horizontal plane. Rolling the shuttle does not affect this plane.

The sensing instruments feed the pilot pertinent data:

- Altitude/entry vector;
- Airspeed/velocity;
- Heading;
- Glide (rate of descent);
- Time;
- Temperature;
- Pitch and roll values;
- Pitch and roll indicators;
- Latitude and longitude;
- Artificial horizon;
- Distance to spaceport;
- Localizer indicating distance off centerline of runway;
- Stall indicator;
- Temperature warning;
- Status of landing gear; and
- Status of speedbrakes.

Command line single key entries are:

- (P)itch—raise and lower nose;
- (R)oll—left or right bank;
- (T)hrust—control pitch and heading above 58 miles;
- (Y)aw—minor heading corrections without rolling;
- Trac(K)—check in to tracking stations; and
- (G)o—accelerate the flight-phase update at will.

At the bottom of the screen is the command line that allows the pilot to control the flight attitude of the shuttle and report at the appropriate times to the tracking stations. Except for the lack of feel of the shuttle, everything is there on the flight panel to permit the astronaut to guide the shuttle to a safe landing. It is more difficult to perform this flight than one would imagine.

Flying the Columbia

There are several aspects that differ from the airplane. The most striking difference is the surfboarding effect of the shuttle as it approaches reentry. If the entry phase is not carried out with precision and diligence, Columbia will skip off the atmosphere into space never to return again. On reentry the craft's hull shields it from heat.

Orientation and reality can be lost during the reentry process. I imagine this feeling might be similar to the experience of Columbia's pilots. This is because the scope of reentry is larger than your perceptions. This is not a fly-it-by-the-seat-of-your-pants operation, so it becomes necessary to fly the shuttle "by the book"—flying the shuttle to conform to a set of computer-generated coordinates that appear to make no sense. A sort of blind trust seems to be involved, a faith in computation.

Guiding an aircraft using heading, latitude and longitude bearings requires you to think of the Earth as a globe striped with markings bearing latitude and longitude. For the purposes of this simulation, the space shuttle is required to fly over a series of tracking stations along the reentry path. Not all of the check points are straight ahead. The pilot must manipulate the shuttle to guide it over these points. Once over the tracking station, he checks in via the command line. The computer tallies his performance throughout the reentry process and assigns performance points for how close the shuttle comes to overflying various check points.

Your performance score is based on a possible 1,000 points: Reentry is allotted 100; general flight, 100; navigation, 400; and landing, 400. Points are deducted as you commit astronautical malpractice such as stalling or overheating the craft. The flight conduct after reentry and the final approach and landing are also evaluated.

The spaceport runway is located on longitude 80.65 degrees west. The shuttle lands to the north, or on runway 36. Your approach on final is made at a heading of zero degrees, using the yaw control frequently to fine-tune any course correction.

It is here something should be said about navigation. The secret to flying the Columbia is understanding your spatial orientation. When heading 90 degrees (due east), the shuttle parallels latitude. When heading zero degrees (due north), it parallels longitude. This is great to know since any course in between simultaneously affects latitude and longitude.

Reader Service for facing page ✓97.→


StarFighter
The PENULTIMATE Space War Game is HERE!
TRS-80*
Model 1 & 3
16K and up — Tape
010-0102 . . . . $24.95
32K and up — Disk
012-0102 . . . . $29.95
Adventure INTERNATIONAL
A DIVISION OF SCOTT ADAMS, INC.
BOX 3435 • LONGWOOD, FL 32750 • (305) 862-6917
ORDER FROM YOUR FAVORITE DEALER
or CALL TOLL FREE (800) 327-7172
SHIPPING & HANDLING CHARGES ARE EXTRA
PRICES SUBJECT TO CHANGE WITHOUT NOTICE
IOTO COPYRIGHT · DON DIXON
*TRS-80 is registered Trademark of the Tandy Corp.

The complicated nature of the navigation makes it necessary to prepare in advance where you want to be and when. I found it extremely helpful to create a flowchart of what I was supposed to do to be where I was supposed to be at the right time. Even then, it was difficult and takes some practice.

An inspection of the check points reveals the space shuttle is traveling west (where longitude is larger) to east, and south (where latitude is smaller) to north. If you visualize this flight path, you have made a significant step in solving the navigational problem.

Once proper entry has been made, the message "Orbit Achieved" is printed. There is no indication when you are going to hit the entry window, only that you are so many feet above or below the window. If you arrive at the window but are too high or low, you could lose control of the shuttle. This is the reason for getting down to business the moment you initiate the flight. Getting the "Orbit Achieved" message assures you that you have passed through the window and are within the grasp of the earth's gravity. The nose is manipulated to adjust the shuttle's speed as you approach the first tracking. This is the most southwestern tracking station you must report to. Immediately past this, you take a heading of 70–80 degrees to move your craft northeast toward the next station, South Texas Radio. Watch the nose temperature! If you overheat, you will be warned (and you will lose points in the scoring). Remember, this scoring is not a game; it is a measure of your performance.

The next tracking station, U.S.S. Taft, lies in the Gulf of Mexico. And the final check-in is made to the U.S.S. Fairchild off the coast of Florida. Once this is made, start a left turn for the final approach. This turn will take you off your easterly heading toward the north. Your final approach will be on a due north heading. A localizer data display will tell how far off to the left or right of the centerline of the runway the Columbia is.

Once on the final approach, you still have problems. Proper velocity and heading have to be maintained, while the altitude is taken into consideration. Remember, you are dead-sticking the shuttle in! The main chore here is to get on a proper glide path and hold it. Ideally, within 100 miles the shuttle's speed should be around 2,000 mph and its altitude at about two to three miles. Twenty-five miles out, the speed should be around 400 mph and the altitude at about one mile. Lower the gear and adjust the speedbrakes again. As you near the approach of the runway, if you are in the final glide pocket, control of the space shuttle is taken over by the auto pilot and it completes the landing for you.

If all has gone well and the auto pilot has landed the shuttle, the computer gives you a report (score) of your performance. If not, probably one of the following has taken place.

> *"Because you landed safely does not. . . mean that you will have a good score."*

What Can Go Wrong?

The first and most disquieting event occurs during reentry. As mentioned before, if the entry vector is made too high for orbital insertion, the shuttle may skip back into space. If too low, it may burn up in the atmosphere. In either case, the simulation ends.

While making a descent from about 200 hundred miles up, I allowed the nose to pitch up too high. I compounded this adding even more down thrust, forcing the nose still higher. The stall warning went unnoticed and so did the stalled indication. When I finally noticed the pitch indicator, it was moving through 600 degrees, the airspeed was 160 mph and I was dropping at a rate of 35,000 feet per minute. I interpreted this interesting maneuver as tumbling.

The reason for arriving at this attitude was due to my forgetting how much thrust was given and in which direction. There is no indication that a thruster is firing once you have fired it. And a thruster fires continuously until you neutralize it with opposite thrust. So applying five units of down thrust, then giving the same command again results in 10 units of down thrust. You can see the result of this by watching the pitch data. As many as 100 units of thrust are allowed at any one input. But the units may be additive. This was another good reason for maintaining an accurate flight log.

The most obvious problem is navigation. You can deviate from course by a considerable distance. If you fly south of latitude 21.5 degrees north or north of latitude 28.7 degrees north, you will be given an Off Course warning. The effect of this wandering can make getting to the runway difficult. And speaking of off course, being on the final approach does not mean you will arrive at the runway. First, you have to move onto the centerline of the runway. A problem I had was losing altitude. Arriving at the approach end of the runway at 24,000 feet is a little embarrassing. Of course, you are permitted to circle and with that much altitude, there should be no problem. But there is. As the shuttle plummets towards earth, it seems to be very sluggish. It is big, heavy and fast. Making a circling approach over the runway was difficult for me. I had trouble realigning the shuttle back onto the centerline. There is a lot of inertia there.

The so-called straight-in landing is anything but easy. A common problem was overshooting the runway because of too much altitude. You certainly can do it while the instruments faithfully log your position. A message appears if you land off the runway. Landing short of the runway is considered a poor landing. Landing too hard could destroy the shuttle. In other words, just trying to get the space shuttle in a configuration allowing the auto pilot to take over becomes an exhilarating event. And if you have forgotten to put the landing gear down, it is a little messy. When the shuttle is on the runway you must endure the computer's measure of your performance—whether it was good or bad. Just because you landed safely does not necessarily mean that you will have a good score.

Conclusion

Space Shuttle is a different sort of simulation. There are no sophisticated graphics. One nice feature of the copy I received was it was written in Basic, making fine-tuning easy. On the other hand, being written in Basic possibly was a restrictive influence on the design of the program presentation.

Since it is in Basic, it might be interesting to see how the program would work if one is fortunate to have a Basic compiler available, modifying the appropriate timing loops where indicated. I am not that eager to try to rewrite what is already a well thought-out algorithm, nor to try to make any particular adjustment in the program. If one was inclined to do so, it should not be an impossible task.

As for the program's space algorithm, I thought it was terrific! It seemed plausibly accurate. All in all, left exactly like it is now, it is quite entertaining. For those of you with an itch to pilot the Columbia, this is the program for you. ■

ADVENTURE INTERNATIONAL

NEW

THE Adventure Continues

The Scott Adams' Adventure series announces the release of Adventure #11 *"Savage Island,* part 2" and Adventure #12 *"Golden Voyage"*.

For The • TRS-80 • APPLE • ATARI • $19.95 tape

Also available on Bonus Triple Pack Disk Adventures #10, #11, and #12. For the TRS-80 and Apple $39.95 ✓222

2 maces & magic SERIES **NEW**

STONE of Sisyphus

The Stone of Sisyphus carries you to the "thinking man's" dungeon. A subterranean world of monsters, magic, traps and treasures demands brains rather than luck to survive (of course, you might take along your mace just in case...).

MACES & MAGIC are fantasy adventures involving you and your computer. Armed only with your wits, a microcomputer, and the software provided, you can become the hero or the meal your destiny dictates. You create a character, equip him (or her) with suitable weapons and armor, and enter the dungeon in search of fame and fortune. Neither is particularly easy to obtain.

If you are successful in avoiding or conquering the various monsters, traps, enchantments and illusions set by our nefarious dungeonmasters, you may escape with riches and glory. Your name and deeds will be recorded for posterity in the records of the dungeon. More importantly, you'll be alive. You may then use the same character in his more experienced and wealthy form when you enter dungeons on later occasions.

The object of the whole exercise is not just to fight the monsters and collect treasure. You have to get out alive to enjoy it. In every dungeon there is at least one exit. It is possible to escape from each and every dungeon with a whole skin. We state that fact here because players often believe this not to be true. We really aren't out to get you. Not really...

FOR ALL OF THE BELOW
Apple 2 Disk with Applesoft
or Apple 2 Plus 48K
TRS-80 32K Model 1 or
TRS-80 48K Model 3

032-0100
$29.95

TWO FULL DISKS WORTH OF DATA!!

A compacted data structure makes it possible to run this fantasy on a single disk drive, though 2 full diskettes worth of data are supplied. Expanded use of free-form input puts your ingenuity to an even greater test. The responsiveness of the programs to the individual qualities of the character make this adventure frustratingly enjoyable for hundreds of hours before its secrets can be unlocked.

SCOTT ADAMS SEAL of APPROVAL

A Division of Scott Adams, Inc.
Adventure International • Box 3435 • Longwood, FL 32750
TOLL-FREE ORDER NUMBER: (800) 327-7172
IN FLORIDA (305) 862-6917
SHIPPING AND HANDLING CHARGES ARE EXTRA
PRICES ARE SUBJECT TO CHANGE WITHOUT NOTICE

✓224

ai

80 APPLICATIONS

by Dennis Kitsz

Autumn will be upon us soon, but the nights are already longer than the days in Roxbury, since it nestles between two ranges of the Green Mountains at the headwaters of three rivers. Across the street, the mechanic is putting the clapboards on his solar-powered, microprocessor-controlled garage. As if to compete with that technological neighbor, the sunflowers are reaching to top last year's highest bloom, more than 14 feet. All these simultaneous occurrences could mean just one thing—it's time for the second annual Autumnal Equinox Hardware Festival!

You can keep the soldering irons cool, though, because this month's column is (by popular demand, no less) an illustrated introduction to electronic hardware—resistors, capacitors, transistors, diodes, integrated circuits, and displays. There won't be a heavy dose of theory, just a simple rundown on identification of parts, how to read their values, and what they look like.

Your basic TRS-80 Model I keyboard unit consists of 59 capacitors, 11 diodes, 6 transistors, 76 resistors, 82 integrated circuits, switches, sockets, a crystal, and a fistful of connection hardware. If this quick list of contents doesn't conjure up any mental images of particular parts, then this column is for you.

What's an IC?

An integrated circuit (IC) is a black box, figuratively and most of the time literally. Once upon a time, there was a great, ambitious space program, one that required a lot of electronia in a very small capsule that also contained a fairly bulky human. Electronic engineers set about to miniaturize as much of the electronia as possible, whittling the metal cans off individual transistors and stuffing several of the transistors together wherever these individual parts had some logical relationship. Gradually the engineers came to understand that general-purpose circuit building blocks could be built inside a single metal or plastic package, saving both valuable space and a lot of piecemeal wiring. They concluded that only those circuit connections which had to be mated to other circuits needed to be brought to the outside of these combinatorial black boxes.

The technical prowess to achieve this integration of parts came slowly at first, especially since the industry had only very recently learned how to produce individual transistors reliably. To produce 10 or 15 or 20 transistors joined together, all perfect, was a major achievement as significant as a man first stepping on the surface of the moon; the whole story can be read in a special issue of *Scientific American* (September 1977), but for the moment consider this remarkable fact: The 16K bytes of memory inside your run-of-the-mill TRS-80 contain the equivalent of over 140,000 transistors. A complete TRS-80 system with disk drives and printer would boast a miniaturized body of nearly a million individual transistors—and every one must be perfect for your low-cost personal computer to operate.

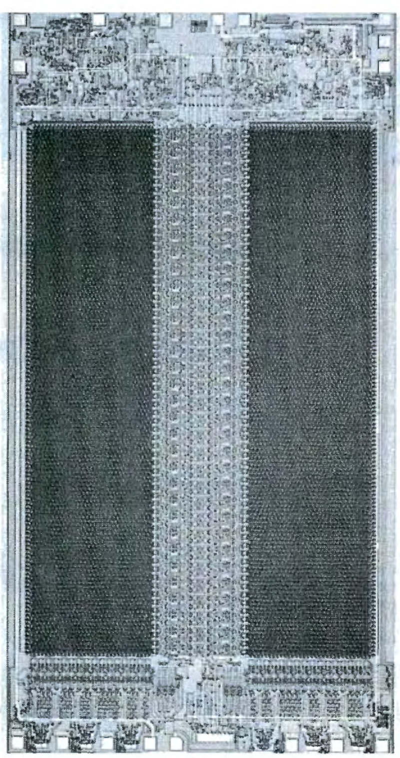


Fig. 1. Mostek Co. Memory Circuit Chip (Scientific American, 1977).

If you are not in awe of this achievement, give up one of those 16K RAMs which now cost less than $2 each in sets of eight; carefully crack open its cover just above the external pin connections. Get a jeweler friend to help if you haven't got steady hands. Now take a look at Fig. 1, which shows what you've got inside that memory circuit on a three by five-millimeter chip of little more than refined sand. That's a remarkable technical achievement, I think you'll agree.

Integrated circuits come in a variety of purposes, voltages, interconnection characteristics, and shapes and sizes. With a few exceptions, the purpose of those inside your TRS-80 is as digital building blocks; they operate from a five-volt power supply; they directly interconnect with each other; and they are produced in industry-standard, dual-inline packages (DIPs). The TRS-80 ICs that don't fit this mold are its voltage regulators, which are not digital circuits; its program memories, which need 12 volts, 5 volts, and −5 volts; and the cassette and video input/output control ICs, which do not connect directly to the others.

Photo 1 shows a group of integrated circuits. The first row across the top is the bank of eight 16K dynamic RAMs (read/write memories) used for program memory inside the TRS-80. The next row contains eight 1K static RAMs, used for the video memory. You can't tell much about a chip by looking at its shape, as these two similar rows demonstrate; shapes are part of an industry-wide standard which specifies that the external pin connections are spaced 1/10 inch apart. Three different widths of the dual-inline packages (DIPs) are used, depending on the number of pins.

The remainder of the integrated circuits in the picture have been chosen to give a general view of the variety of functions similar-looking ICs may have. The two white integrated circuits are in gold-topped ceramic packages. The first is a serial keyboard encoder (it scans a keyboard and outputs a stream of bits representing the key pressed) and the second is

TRS-80™ PROGRAMS
SUPER SOFTWARE AT SUPER PRICES

THE ULTIMATE DISK-OPERATING SYSTEM

LDOS™

A completely documented new generation of operating system for the TRS-80. Far superior to any on the market. It is a totally independent device system, capable of device linking, routing, setting, and filtering. LDOS will support 5" and 8" floppies, single/double density, single/double sided, and up to 80 tracks. Full customer service provided with a toll free 800 number. Excellent documentation in a tab indexed manual (over 250 pages). Call or send for more information today.

Available for Model I or III . . . . only $149.00

LDOS™ is a product of Logical Systems Inc.

ALSO AVAILABLE

EDAS 3.5 Model I and III. . . $79.00
EDAS 5.0 Model II. $179.00
Host Model II was. $199.00
now. $99.00

BUSINESS SYSTEMS

INVENTORY MASTER SYSTEM

With today's market, keeping on hand only what your demand calls for is reason enough to let your TRS-80 accurately and effectively take care of your Inventory. Always know what to stock and when to stock it. This system has many features which were modeled after a main frame system of large capacity. More information available upon request.

Model I version just. $159.00
Model III version just. $259.00

ALSO AVAILABLE . . . For the MODEL I or III STOCK MARKET MONITOR

Cassette version. $89.00
Disk version. $99.00

galactic software ltd.
11520 North Port Washington Road
Mequon, Wisconsin 53092

MAILING SYSTEMS

MAIL/FILE - SERIES II

NEW Series II Mail/File has all of the outstanding features of the original Series I Mail/File, PLUS many additional qualities which again will set standards for this type of system. "Word processor" type input editor, fast sorting, and fabulous editing capabilities are a few of its features. Name and ZIP code are under constant sort. Series II is compatable with Series I data files.

NEW RELEASE

Model I system will handle up to 600 names.
Model III system will handle up to 1200 names.
Either system available for only. $159.00
Also available for the Model II. $199.00

THE ULTIMATE IN MAILING SYSTEMS MASS/MAIL SUBSCRIPTION SYSTEM AVAILABLE FOR MODEL II.

"I think the system will stand the test of time . . . it is probably the most sophisticated system around for the micros today"
Mike Schmidt (EDITOR) 80-US Jan/Feb pg. 94

Galactic's Mass/Mail System was designed with the needs of the large mailing list in mind. This system has capabilities of handling up to 10,500 records, with access times of less than 15 seconds and usual access of about one second. All adds, deletes and edits are instant for the operator and

MAILING SYSTEMS

are then completed later in a "batch mode". The System comes complete with extensive documentation and on-going support. . . . .

Contact Galactic Software for detailed specifications for your exact needs. Available only for the TRS-80 Model II.

ULTRA-TREK GAME

Can you master this trip through the galaxies?

Climb aboard the Star-Ship Enterprise and head for a universe of unknown encounters, enemies and stellar battles. As commander, will you be able to keep the Enterprise and its crew en route and on patrol? Can you destroy the hostile base? Order today.

Model I or III version only $19.95

All of Galactic's products have a one year limited warranty with customer support included with each purchase. Warranty information is available upon request in writing.

Money orders, credit cards, & cod's shipped within 24 hours.

Can't wait? call

414/241-8030

VISA® master charge

TRS-80 A Trademark of Tandy Corp.

Photo 1. Integrated Circuits

an industrial type 8060 microprocessor. Next in the photo is the heart of the TRS-80—a Z-80 microprocessor in a black plastic package. All three are in 40-pin arrangements.

Following are two chips used in 8080-based computer systems: a 28-pin 8251 serial interface, and a 24-pin 8212 bidirectional buffer. These chips permit an 8080 microprocessor to talk to the outside world in different ways. The third row shows a pair of 24-pin integrated circuits containing Level I Basic. Next is a 2716 erasable, programmable, read-only-memory (EPROM) with 2048 bytes of information in a 24-pin grey ceramic case; the EPROM may be programmed and erased by a user. The last three ICs are a three-state buffer 81LS95, which is an ideal tool for interfacing the TRS-80 edge card to the real world; an ordinary transistor-to-transistor logic (TTL) circuit, from which family over 60 of the ICs inside the TRS-80 are drawn; and a 1024-bit shift register, which sends bits along in a bucket-brigade-like stream. Finally, turned sideways in the photo are two IC-like parts. These are DIP shunts, groups of metal bars which may be broken by puncturing. These are inexpensive programmable switches used in the computer to connect together circuitry which Level I or Level II Basic, 4K or 16K memory.

Reading ICs

Occasionally you may see a magazine article which describes a modification you'd like to make to your TRS-80. Don't be afraid to tackle it (once you've checked the Debug column for the next two months). For hardware newcomers, the most fearsome part of going inside a computer seems to be that anonymous, threatening, indistiguishable mob of integrated circuit black boxes. Actually, they're really quite tame. Every integrated circuit inside the TRS-80 is marked in two ways. The first marking is a "Z" number silkscreened on the circuit board, which is a simple way of referencing its board position; this number can also be found on the schematic diagram. The second marking is the combined industry-standard part number, manufacturer code, and dating code. The code for Z44 stamped on the IC may look something like this:

SN74LS367N
7914

By cross-checking the Z number in the TRS-80 parts list, you will find this is a 74LS367 tri-state hex buffer (whatever that means). The number itself (74367) is an industry-standard, generic type number in the 74xx series of logic circuits; the LS is an abbreviation for low-power Schottky, a specific family within this generic circuit tape (you may find LS, C, H, L, S, or no letters at all). The SN prefix in the code is the manufacturer's identification (Texas Instruments uses SN, Signetics uses NE, etc.); the suffix letter is the package style (N means dual-inline plastic). Finally, 7914 is a date code in year/week format (the part was manufactured during the 14th week of 1979). So the part number SN74LS367N 7914 tells us that this circuit is a low-power Schottky-type tri-state buffer in a plastic dual-inline package, manufactured by Texas Instruments during the 14th week of 1979 (whew!).

There's an easier way to discover the part type. Look through the Z numbers, examine the integrated circuits on the TRS-80 circuit board, and reference the parts list (pages 93–98 in the *Technical Reference Handbook*). You should get a feel for how part numbers are printed and how to read them. Occasionally you will find a part number with no relationship whatsoever to the generic number in the parts list; the schematic also does not match the part on the board. In this case you are probably looking at a "house-numbered" IC; house numbering is done when a customer (in this case, Tandy Corporation) orders a large quantity of parts, and the manufacturer's production run is dedicated exclusively to that order. House numbering is often found on the TRS-80's video memories.

"Reading" an integrated circuit is easy, since they're all numbered the same way; see Fig. 2. The IC is positioned by finding a notch or a dot conspicuously placed toward one end; look down from the top (some ICs have both a notch and a dot). This is the keyed end of the IC. With the notch or dot pointing north, pin 1 is to the west. Pin 2 is to the south of pin 1; pin 3 is next to the south of pin 2; etc.

When you get to the bottom of the IC (say at pin 8 on a 16-pin DIP), the next pin is immediately to the east (say 9). Pin 10 is to the north, 11 north of that. The highest numbered pin always ends up opposite the lowest numbered pin. It's done in a "square circle".

If you read an article that says, "solder a wire to pin 12 of Z44", you would start at the top left and count 'round the bottom until you reached 12. That's the place.

The Logic Blocks Inside

Digital logic is a subject I've covered before (see "80 Applications," May and June, 1980, March, 1981, and accompanying hardware construction projects), but this time I'll describe the hardware from—as much as possible—a software approach, and with lots of analogies to everyday situations. Let's start with some simple logic functions: AND, OR, Exclusive OR, and NOT (complement).

Four simple AND, OR, Exclusive OR, and CPL (complement, or NOT) calculations using only two binary digits are shown in Example 1.

These are the truth tables for single bits, something with which machine-code programmers will be very familiar. Basic users, too, have the AND, OR and NOT functions available. For each truth table, there is an equivalent, commercially available, logical integrated circuit (Example 2).

Some integrated circuits contain multiple inputs—the 7411 three-input AND gate, the four-input 7420 NAND gate, the three-

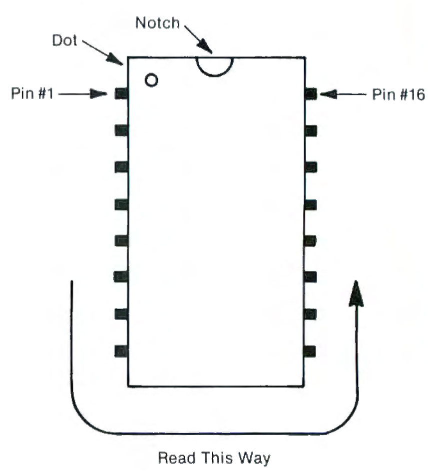
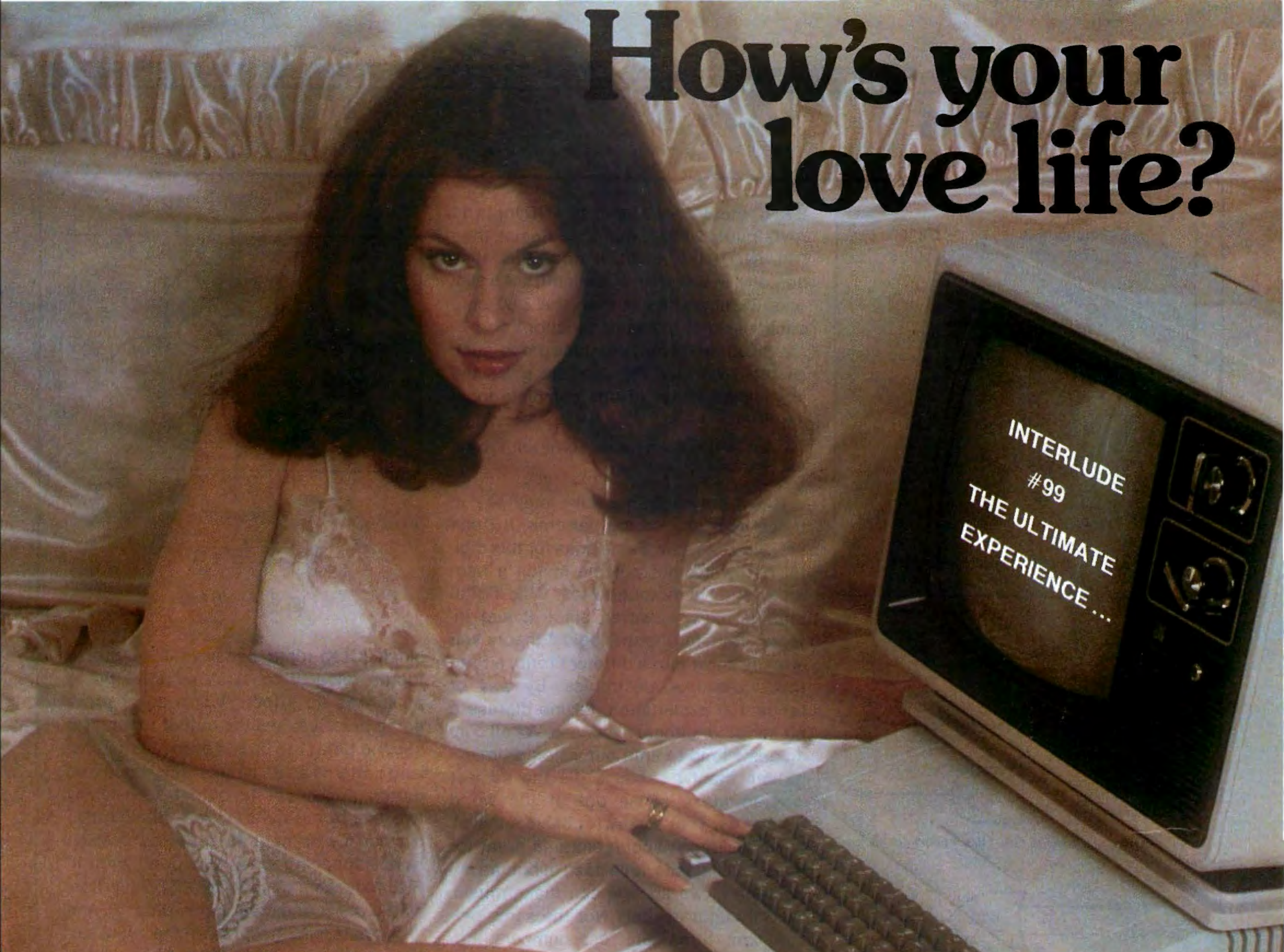
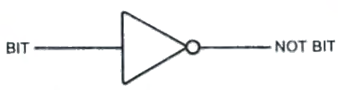


Fig. 2. How to read integrated-circuit pin numbers.

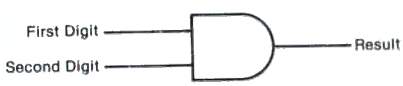

How's your love life?
INTERLUDE
#99
THE ULTIMATE
EXPERIENCE...

Bit X	0	0	1	1
AND Bit Y	0	1	0	1
	--	--	--	--
Result	0	0	0	1
CPL (NOT)	1	1	1	0
Bit X	0	0	1	1
OR Bit Y	0	1	0	1
	--	--	--	--
Result	0	1	1	1
CPL (NOT)	1	0	0	0
Bit X	0	0	1	1
XOR Bit Y	0	1	0	1
	--	--	--	--
Result	0	1	1	0
CPL (NOT)	1	0	0	1

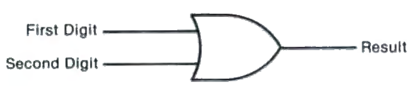
Example 1. Binary Digit Calculations



Not (CPL) Gate, Generic type 7404, 6 gates per IC.

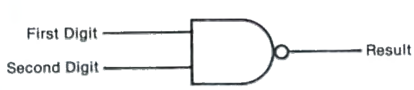


AND Gate, Generic type 7408, 4 gates per IC.

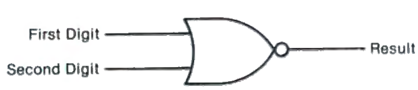


OR Gate, Generic type 7432, 4 gates per IC.

XOR Gate, Generic type 7486, 4 gates per IC.



NAND Gate, Generic type 7400, 4 gates per IC.



NOR Gate, Generic type 7402, 4 gates per IC.

XNOR Gate, Generic type 74266, 4 gates per IC.

Example 2. Equivalent Integrated Circuits

input 7427 NOR gate, etc.—which have the effect of applying the logic function to all inputs. The 13-input 74133 NAND gate would perform this function: Input 1 AND Input 2 AND Input 3 AND Input 4 AND Input 5 AND Input 6 AND Input 7 AND Input 8 AND Input 9 AND Input 10 AND Input 11 AND Input 12 AND Input 13 = Result, then CPL (NOT) the result. It does this, incidentally, in less than 15 billionths of a second, 30 times faster than the Z-80 chip could perform the same function!

Still other ICs may combine gates, such as the 7464 (see Fig. 3). This gives the single-digit result of this complete logic equation:

Y = NOT ((A AND B AND C AND D) OR (E AND F) OR (G AND H AND I) OR (J AND K))

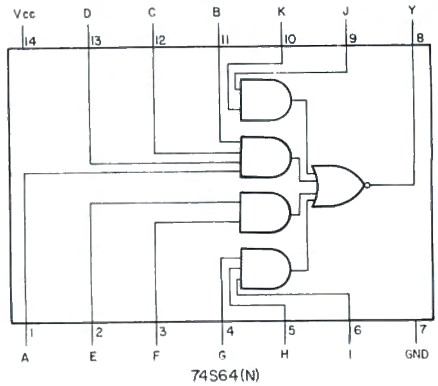
Okay, stay with me. Like they say in those ads for clothing patterns, if I can sew, you can sew. Same goes for this digital hardware; but put me inside a ham radio, and I'm thoroughly lost. This digital stuff is easy, however, since it just involves transferring the logical theory you already know into a different form, a form that looks like electronics. By way of tangential explanation, let me digress a bit about functional fixedness. We all consciously tend to limit our vision or imagination, mostly because it's the easiest way to make sense out of the world around us. That's okay for basic survival, but consider: A spoon alone is great for soup, but if used with a dash of human imagination, two spoons can make music. Who thought of that? Not the same person who turns junk car bumpers into sculpture, but the principle is about the same. Here's another thought: Until a new timing belt arrives, my printer's carriage return functions because I have nylon string attached to the print head, looped over a pulley, and weighted down with a paper punch. It works, with no fancy electronia. Think about this: If thumbs were the key factor in the evolution of humans from lower animals, then binary is a truly unnatural number system. If it's unnatural, then no matter how it's transformed, it's legitimate if we need it that way. Now back to digital logic.

Consider this listing:

```
LD      A,VALUE
RRCA
OUT     PORT,A
```

What is being output through port A? Assuming A is greater than two (hex 10), then RRCA (rotate accumulator to the right) performs a binary division by two. (For those unsure about this, consider that the number 57440 decimal when rotated to the right would be 05744—division by 10. It works the same in base two.) There's a digital circuit that can perform this same division-by-two function, and it's called a flip-flop. It's a permanently hard-wired RRCA function, and it's shown as a box.

A positive-going pulse going into its clock (CLK) input causes the output to toggle once (0 to 1). A negative going pulse has no effect. A second positive-going pulse causes the output to toggle a second time (1 to 0). For every pair of one-zero patterns as its input, a single one-zero pattern appears at the output—division by two, electronic style. Oh yes—there's also a clear (CLR) input to keep the division from happening when you don't want it. Fig. 4 shows a generic type 7473 flip-flop, with a clock and clear input, plus "J" and "K" inputs, which represent the initial setting of 1 and 0. Q and Q-bar (the Q with a line over it) represent the output in its "true" form, plus in complement form—a bonus provided by many integrated circuits. Eight flip-flops are like having a processor with both an accumulator and a complement accumulator available at the same time.



$$Y = \overline{ABCD + EF + GHI + JK}$$

Fig. 3. A 7464 IC

Last among logic circuit considerations for this brief introduction is the concept of a three-state device. The first and second logic states are 1 and 0 as you know, but the third state is a kind of electronic invisibility. In a computer, many electronic circuits must share the same group of address and data lines—the address bus and the data bus—but only one may actually use the bus at a given time. With more than a dozen devices connected to the data bus in the TRS-80, a terrible electronic confusion would result if they all tried to submit their information to the central processing unit (CPU) simultaneously.

To avoid this conflict, electronic designers devised an invisible logic state. Liken the computer control circuitry to the traffic light at a complex intersection, where only one line of traffic receives a green light, while all the others receive a red light. Assuming no suicidal or reckless drivers and no right turn on red, then a driver with a green light may continue

TRUTH TABLE

73, H73, L73

INPUTS				OUTPUTS
CLR	CLK	J	K	Q $\overline{QO}$
L	X	X	X	L H
H	Ω	L	L	QO $\overline{QO}$
H	Ω	H	L	H L
H	Ω	L	H	L H
H	Ω	H	H	TOGGLE

TRUTH TABLE
LS73

INPUTS				OUTPUTS
CLR	CLK	J	K	Q $\overline{QO}$
L	X	X	X	L H
H	↓	L	L	QO $\overline{QO}$
H	↓	H	L	H L
H	↓	L	H	L H
H	↓	H	H	TOGGLE
H	H	X	X	QO $\overline{QO}$

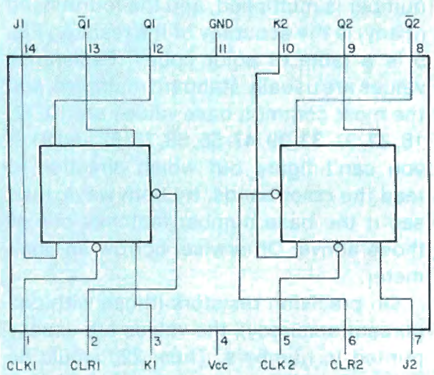


Fig. 4. 7473 Flip-Flop

through the intersection as if there were *no other cars present at all* (that's not exactly true according to my drivers manual, but please accept the analogy). When the light changes, traffic stops in one direction and starts in another. In the electronic intersections of the computer, signals may be output by just one device when it receives the proper signal.

To carry the traffic analogy further, recall that most traffic signals have an amber warning light. In real-life situations, cars slow down (or speed up to get through the light before it turns red). Cars on the other side inch forward upon seeing a hint of that amber signal. This is analogous to the rise and fall time of data signals running through the computer, that safe period during which signals may change logic state from 1 to 0 or from 0 to 1 with no conflict.

One final analogy: Those of you who have driven in New York City know that the traffic signals are timed so well that a driver can start in the financial district and drive clear through to Columbus Circle. Under these conditions the car maintains a constant speed akin to that of the changing traffic signals; all other vehicles can keep the same speed (or at least out of your way); and no emergencies—police cars, accidents, or breakdowns—impede the car's progress. In the computer, this is known as "access time." All electronic parts must be able to complete their responses throughout the system in at least as fast as the master traffic signal, the "system clock." Any part whose access time is too slow (like some RAMs) will have an adverse effect on the operation of the computer.

Narrow Streets and Garages

Resistors and capacitors are abundant in the TRS-80, but I'll keep the theory at a low level for this section. Resistors limit the voltage or current received by some part of a circuit. They can be found providing limited current to light-emitting diode displays (LEDs), holding an IC input to a one or zero level, or as part of the audio circuitry in the video output. Variable resistors serve as adjustments to the video display or to the power supply voltage being sent to the computer as a whole. They act like narrow city cross-streets, where only a limited number of cars can move across town from the large potential auto-streams flowing along main avenues.

Capacitors store an electrical charge, releasing it as needed. Capacitors inside the TRS-80 store power supply irregularities, effectively smoothing them out; they delay the transfer of current from one logic IC to the next (to adjust the horizontal or vertical video image, for example); or they store and release energy at regular intervals to provide clocking signals (what keeps your disk drive on for an extra few seconds, or times the characters being output by your printer).

A collection of resistors is shown in Photo 2. At the bottom right is a group of simple resistors, varying in size according to the amount of current they are capable of handling without overheating. Small resistors are most commonly available as carbon composition (inexpensive, packed carbon) and carbon film (resistive film plated on a non-conducting base). Small resistors make up the bulk of those in the TRS-80.

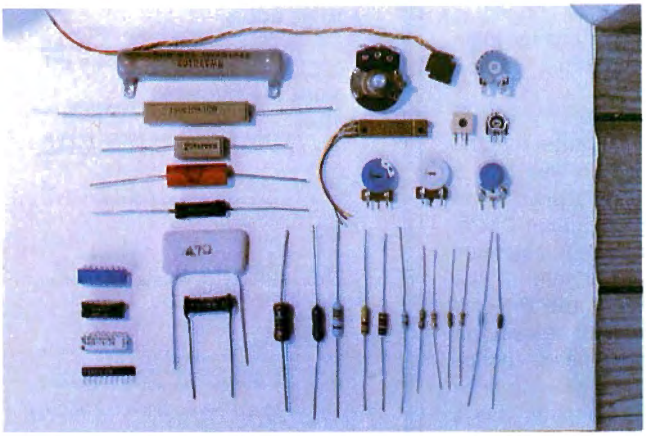


Photo 2. Resistors

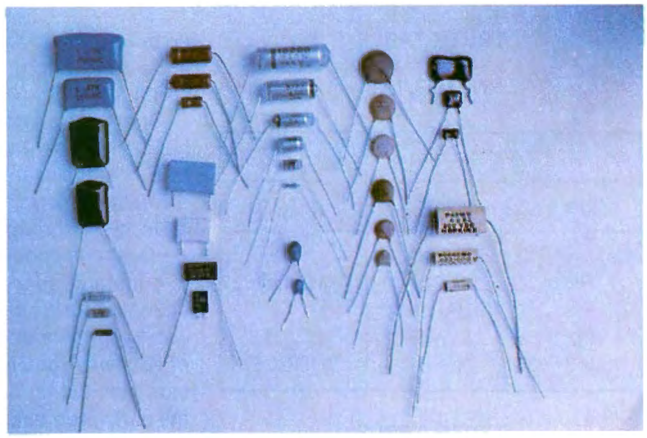


Photo 3. Capacitors

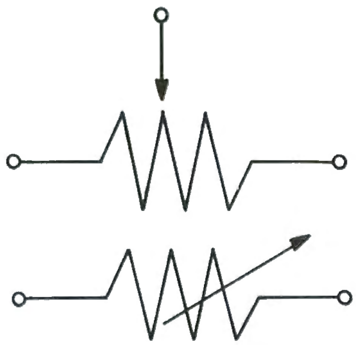
The large vertical row of resistors the photo depicts are heavy-duty wire-wound, porcelain, ceramic, and "sand" resistors. These types are used wherever heavier currents are required; a few ceramic ones are found in the TRS-80 power supply.

At the bottom left are three dual-inline (DIP) resistor packs that look like ICs, and one single-inline (SIP) resistor network. Often, digital circuits will require a string of "pull-up" resistors, needed to activate or keep stable some part of the circuitry. Pull-up resistor values are often the same for all eight data lines, so groups of identical resistors are combined in industry-standard packs such as these. In a TRS-80 system, you may be familiar with the so-called "terminating resistors" required in disk drive 0. In the photo, this is the blue DIP resistor pack, which is made up of eight 150-ohm resistors.

Finally, variable resistors (also called potentiometers) are shown in the top right block of the photo. A standard radio volume control is followed by eight different styles of "trimmer" potentiometers. The white one at the top right is similar to those in the TRS-80 for power supply and video positioning adjustments.

The schematic symbol for a resistor is a crunched-up wire, representing its resistance to the flow of electricity (Example 3). Variable resistors can be drawn either with an arrow showing they are "center-tapped" (a connection to either end of a fixed resistor, with a wiper moving in between) or merely variable (a single connection to the fixed resistor, and one to the wiper) (Example 4).

Example 3. Resistor



Example 4. Variable Resistors

STANDARD COLOR CODE
RESISTORS AND CAPACITORS

Color	First Figure	Second Figure	Multiplier
Black	0	0	None
Brown	1	1	10
Red	2	2	100
Orange	3	3	1,000
Yellow	4	4	10,000
Green	5	5	100,000
Blue	6	6	1,000,000
Violet	7	7	10,000,000
Gray	8	8	100,000,000
White	9	9	1,000,000,000

RESISTOR

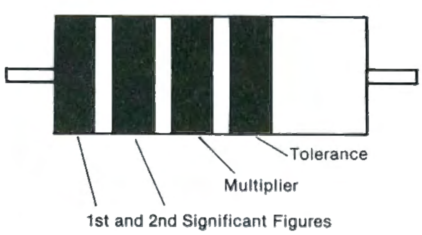


Fig. 5. Resistor Chart

Reading resistance values is quite simple, and has been an industry standard for several decades. Resistor values are measured in ohms, and these are painted on the body of the resistor with color-coded bands: The first and second bands are the resistance value base number, the third band is the power of 10 by which the base number is multiplied, and the fourth band (if any) is the accuracy of the resistor. Fig. 5 is a table of color codes. Resistance values are usually standard multiples, and the most common base values are 10, 12, 18, 22, 27, 33, 39, 47, 56, 68, 75, 82 and 91. If you can't figure out which direction to read the color bands, try both ways, then see if the base number matches one of those above. Otherwise, borrow an ohmmeter.

On precision resistors (those with one percent accuracy), the values are usually printed in numbers. Thus, 220 would be 220 ohms, 473 would be 47,000 ohms, etc. Variable resistors always have their numerical values printed or embossed on the body of the part. In schematics, the values are written with standard abbreviations K and M (kilo = 1,000; mega = 1,000,000). 15,000 ohms is written "15K ohms" or simply "15K." 1,200,000 ohms is written "1.2M ohms" or simply "1.2M." Occasionally you may come across British schematics, which do not follow the American standard. 4K7 is a mildly condensed way of saying 4.7K, or 4700 ohms, 3M9 is 3.9M ohms, etc.

For a while, capacitors were also color coded, but this practice has not caught on in the computer electronics industry. Photo 3 shows a small selection of the many shapes and styles in which capacitors are manufactured, including (top to bottom by columns) dipped paper, tanta-

YOUR WAIT IS OVER!

The fast, simple, affordable way to step up your printing . . .

FAST. The PBI™ (Programmable Buffer Interface) is a unique new solution to an age-old problem. If you've had it with waiting for your printer to complete a run so that you could continue with your work, the PBI is for you. Let our PBI wait for your print-out so neither you nor your computer has to do any waiting. The computer doesn't wait on the printer and the printer doesn't wait on the computer. The PBI increases throughput and saves time by entering data while the printer is printing. The PBI uses none of the host computer's memory and allows linking one or more host computers and one or more printers.

Micro Compatible reserves the right to change any engineering designs and specifications at any time

SIMPLE. The PBI is easy to use. Simply plug the computer cable to the PBI and the PBI to the printer cable.

AFFORDABLE. The PBI will help you avoid the expensive purchase of a high speed printer. We've priced the PBI with the user in mind.

Choose From These PBI Models:

- Single Parallel*
- Dual Parallel**
- Single Parallel and Single Serial (RS-232)
- Dual Serial
- Dual Parallel and Dual Serial

* All Single Ports are 16K Buffers
** All Dual Ports are 32K Buffers

MicroCompatible™

P. O. BOX 7624 • ATLANTA, GA. 30357 • PHONE (404) 874-8366

✓252

✓See List of Advertisers on page 354

Photo 4. Electrolytic Capacitors

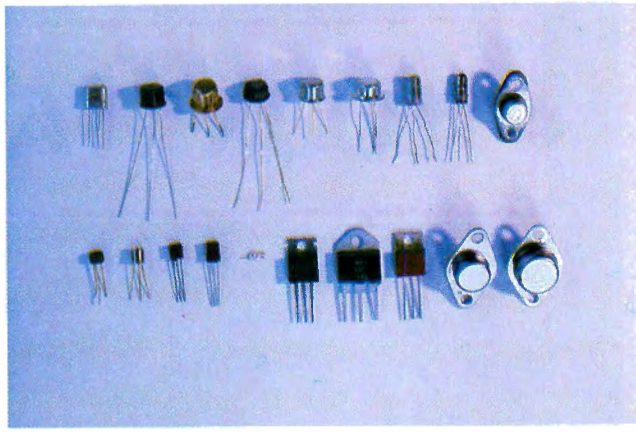


Photo 6. Diodes

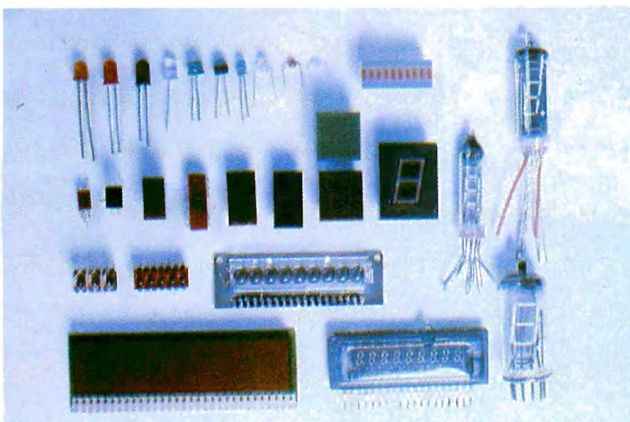


Photo 5. Light Emitting Diodes

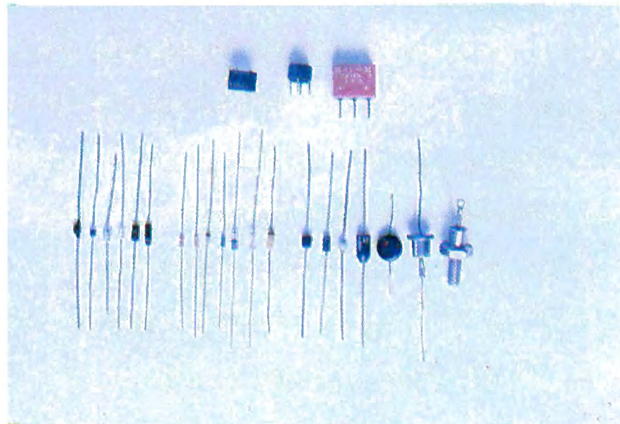


Photo 7. Transistors

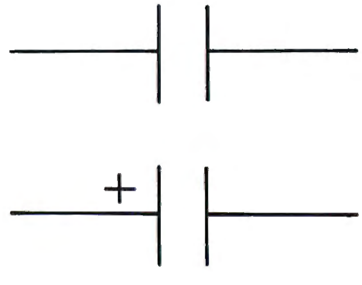
lum, ceramic, polycarbonate, polystyrene, bead tantalum, ceramic disk, silver mica and paper. The general physical construction is similar—layers of foil are separated by non-conducting material (dielectric). The material used, its thickness, and the number of layers determine the amount of current the capacitor can store, and the amount of voltage it can stand before the dielectric shorts across.

In general, ceramic disk capacitors are used wherever only ball-park capacitance is needed. These caps are temperature sensitive and have a tolerance of ±20 percent or worse. Paper capacitors are good for higher voltages, and dipped paper caps are fairly stable. Polycarbonates and polystyrenes are used in audio circuits because of their cleanness and low noise. Silver mica is the choice for close tolerances (within five percent or better), and bead tantalums work well wherever precision, durable high-capacity parts are needed. The TRS-80 uses ceramic disk capacitors throughout, except in its power supply section.

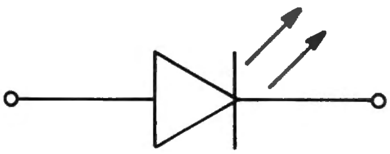
Capacitor values are marked in fractions of a farad. Now, a farad is one heck of a lot of capacitance; look at Photo 4. These are electrolytic capacitors, where the current capacity is increased by using a chemical dielectric. The one on the far right weighs in at half a pound and is less than one-twelfth of a farad! The smallest capactor in Photo 4 has a value of five microfarads, and the smallest capacitor in Photo 3 has a value of 10 picofarads. Capacitor values are abbreviated mf, uf, or μf for microfarads, and pf for picofarads. Sometimes the f is capitalized (mF, pF), and older texts occasionally use the term micromicrofarad in reference to picofarad.

Numerical abbreviations in picofarads are found on capacitors like those for ohms on resistors—474 will mean 470,000 pico farads (0.47 microfarads), and so forth. Again, British standards are somewhat different (and even more sane): Since the most-used capacitor values are midway between the picofarad range and the microfarad range, they will write these values in nanofarads. So don't be thrown if some peripheral devices specifies a "22 nF" capacitor—it means .022 microfarads.

The symbol for a capacitor shows its function well. Two current-carrying plates are separated by a non-conducting material. Electrolytic capacitors are marked by a positive (+) sign toward one end to indicate their placement in a circuit (Example 5).



Example 5. Capacitors and Electrolytic Capacitors.



Example 6. LED

Choosing Replacements

There may come a time when you have to replace a resistor or capacitor in your computer. To do this you must obtain a part of equal or better quality than the one you are replacing. For resistors, ask for carbon film, 1/4 watt, five percent tolerance, which can be found as part of the Radio Shack 271-1300 series. If the *Technical Reference Handbook* specifies wattage higher than 1/4 watt, make sure you get the higher wattage or else you'll watch the resistor darken, smoke, and snap in half. You may, on the other hand, substitute a higher-wattage resistor for a lower-wattage one if it physically fits in place. Make sure that the resistance you choose is within five percent of that specified, unless you are familiar enough with the circuit that you know you can substitute another value safely.

When you replace capacitors, 50-volt ceramic disks are fine for the small ones scattered throughout the board. Radio Shack sells these in its 272-120 series. The *Technical Reference Handbook* parts list specifies 12-volt types for about 30 capacitors, but erring in the direction of a higher voltage is wise. Where 16-volt electrolytic capacitors are specified, 25 or 35-volt replacements will do fine. Also, since electrolytic capacitors have a percentage of error of +100, −50 percent at best, don't hesitate to replace a 220 mf capacitor with a 270 mf capacitor.

Displays

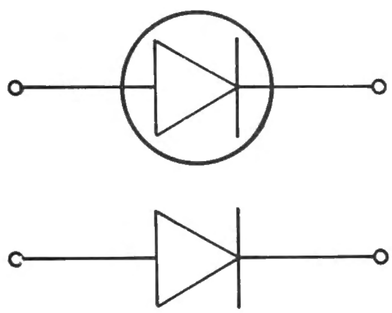
One of the prettiest parts of electronics is its human interface—light-emitting diode (LED) and neon displays. Photo 5 is a collection of these displays, including (at the top left) 10 red, orange, and green single LEDs, from jumbo to micro sizes; (second row) nine seven-segment numeric LEDs; (third row) three seven-segment, multi-character, "bubble" type displays, popular in calculators a few years back; (fourth row) a jumbo seven-segement LED and a multi-character neon display; (vertical row at right) three larger neon displays from desk calculators; (top center) a bar-graph LED used for hi-fi level controls in place of meters.

The schematic symbol for a light-emitting diode is the same as that for a diode (see Example 6), with a few arrows representing light emission.

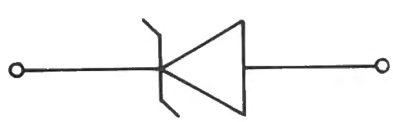
Diodes and Transistors

Diodes and transistors are the building blocks of logic gates and other integrated circuits, but I've left them till last because you will find so few of them in your computer. Photo 6 shows (at top) three bridge rectifiers, which contain four diodes each; (left group) zener diodes, used for controlling voltages; (middle group) small-signal switching diodes, found in certain interfacing and logic circuits and for switching unwanted signals out of a circuit; (right group) rectifier diodes, for transforming alternating current top pulsing direct current. The larger diodes at the right can handle higher power.

The schematic symbol for an ordinary diode consists of an arrow pointing at a bar, showing that electricity may only flow in one direction through these device. You may see either of the two symbols in Example 7; the surrounding circle is more common in older schematics. There are variants on these diode symbols, the most common of which is that for the zener diode. A zener diode conducts electricity in one direction until the voltage rises over a critical point. Then it undergoes a temporary breakdown, letting the voltage conduct backwards through the device. It is an electronic safety value. The broken lines at the edges indicate a zener diode (Example 8).



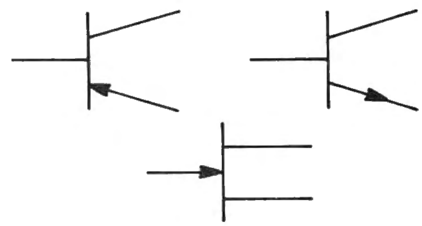
Example 7. Diodes



Example 8. Zener Diode

Transistors were the breakthrough over 30 years ago that made your TRS-80 possible. Although you will find few individual transistors inside present-day computers, there are still some used for non-logic functions. In Photo 7 you can see three general types: (top row) audio transistors; (second row) four small-signal switching transistors; a tiny radar transistor; and five high-power transistors.

Transistors usually have three leads, because they are switching or amplifying devices. Two of the leads represent entry and exit points of some type of circuit; the third lead controls the flow of current between the entry and exit points. Switching transistors flip on and off almost full; audio transistors follow the waveform present on the third lead and mimic its level. I won't go further with the theory because of the many types of transistors, and the numerous considerations needed to design circuits using them. Among the schematic diagrams, see Example 9.



Example 9. Transistors

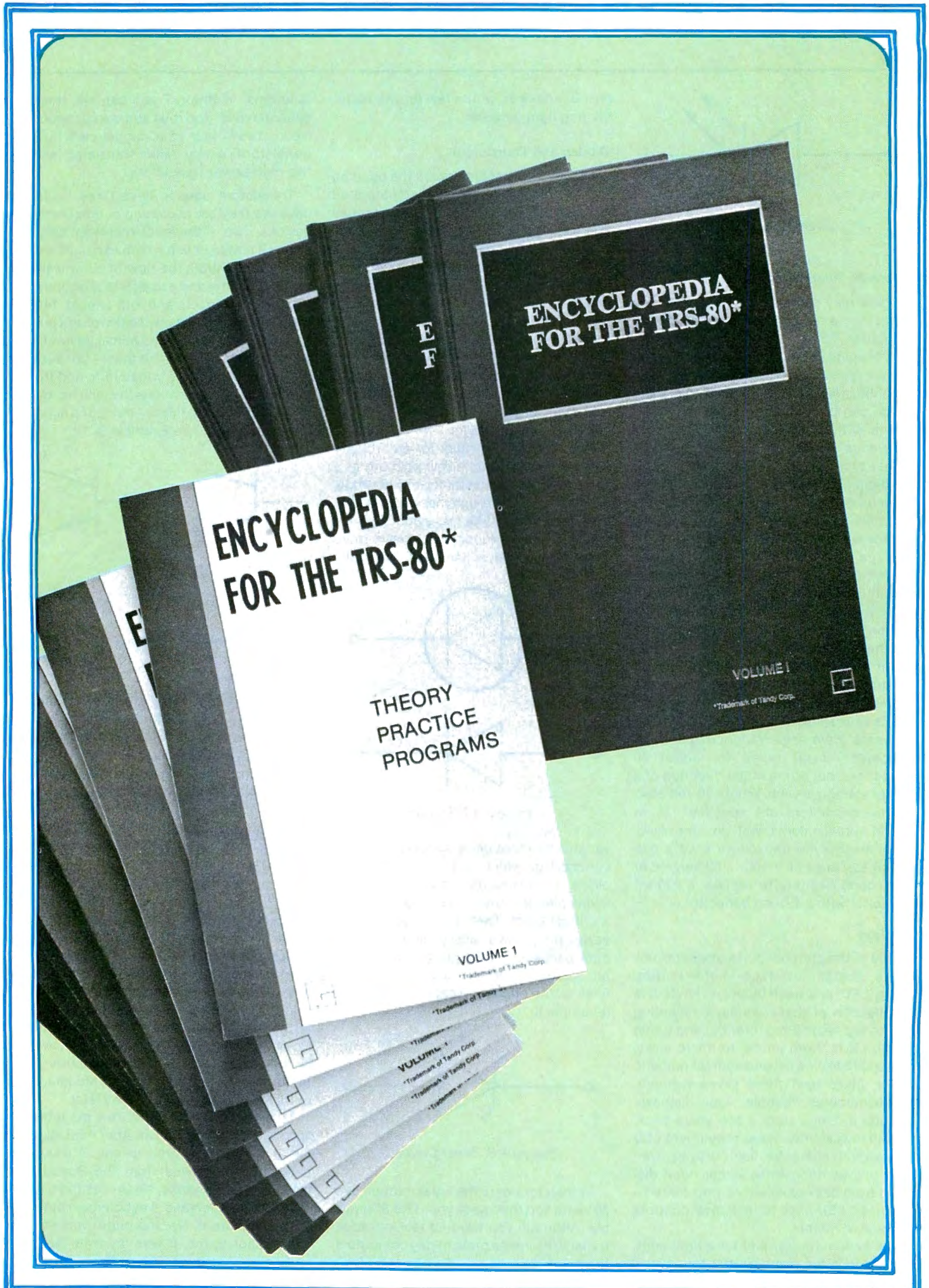
More to Come

In a future column (I'm not promising next month anymore!) I'll present a rundown on what to do when your computer breaks down, some precautions to prevent it, and, when things get desperate, what replacements you can make that will get you up and running. Included will be how even a rank amateur with 10 thumbs (except my friend Paul) can remove and replace a bad IC.

Updates.

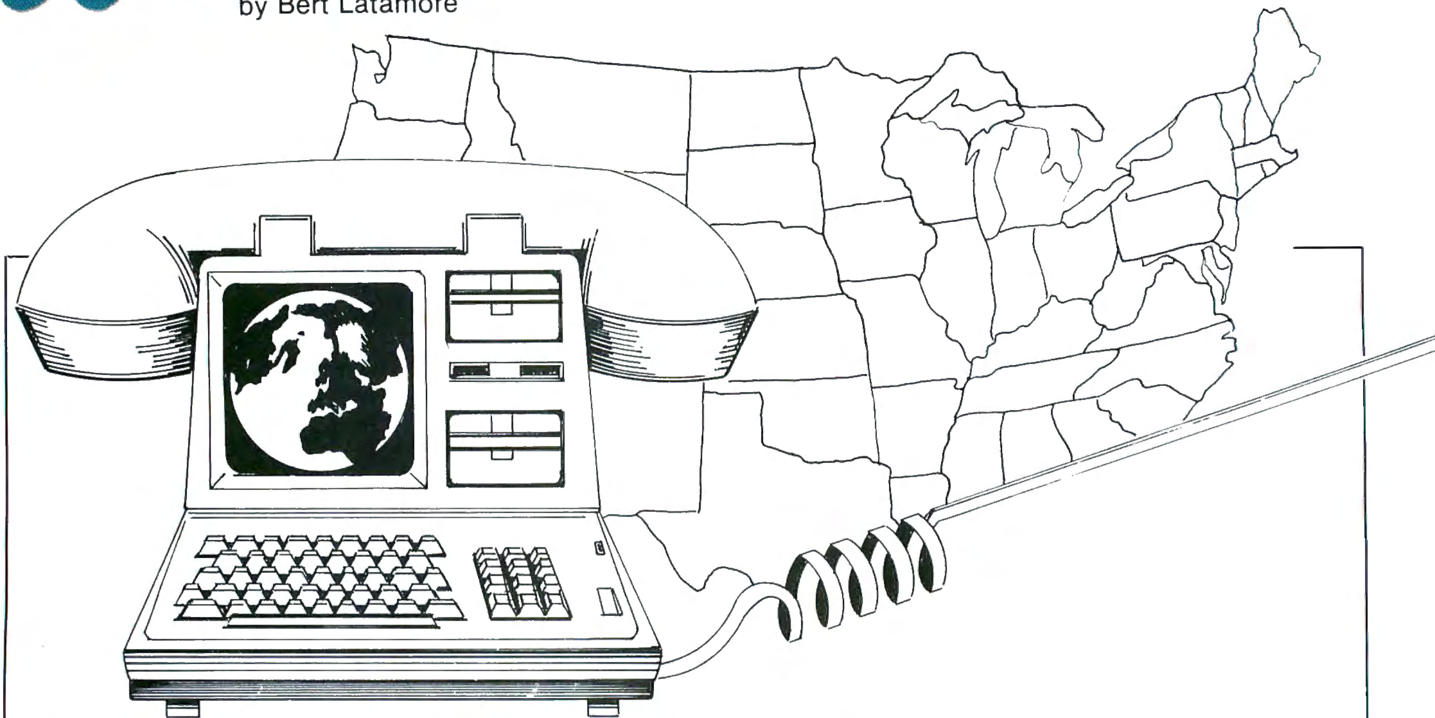
I'd like to thank the hundred or so readers who wrote after reading my report of the minimal response to February's reader survey, and to those who reacted with dismay at my suggestion (in July) that I might be moving away from writing about the TRS-80 Model I in favor of the Color Computer. Yes, I probably will, and here's why: I feel the Color Computer is faster and more powerful than Model I, II or III. More on that sometimes later.

To those who wondered: Yes, the interview with Dr. Lirpa *was* an April Fool joke. No, it wasn't a five-volt salami, it was a daisy ham borrowed from the Roxbury Country Store (thanks, Neal). Yes the program does increase keybounce rather than eliminate it. No, the photo wasn't a secret shot to me, it was my wife Claire (put on your glasses, folks!).■


ENCYCLOPEDIA FOR THE TRS-80*
VOLUME I
*Trademark of Tandy Corp.
ENCYCLOPEDIA FOR THE TRS-80*
THEORY
PRACTICE
PROGRAMS
VOLUME 1
*Trademark of Tandy Corp.

80 NEWS

by Bert Latamore



Networks Wire You to the World

Strung out between the terminals and microcomputers of America and the computer banks of The Source, Compuserve and several hundred other timesharing services worldwide, are thousands of miles of leased telephone lines.

Originally designed for use by larger computers, this international system of linked common-carrier networks and the vast majority of the databanks they serve are compatible with TRS-80s and Tandy's Videotex terminals.

While most services are specialized, their large number ensures that they have something to offer everyone, whether it's timely legal advice, medical information or a handy way to communicate with someone overseas.

The networks cut the cost of reaching the data banks. Some of them can handle certain compatibility problems between different computers and protect data from transmission problems ranging from line noise to physical breaks in the cables.

Most users, however, are only aware of the service they are subscribing to.

Early Experiments

Common carrier data transmission networks like Tymnet and Telenet grew out of experiments in the late 1960s and 1970s. Telenet, Vienna, VA, was one of the first experiments which went public after several years of successful private operation.

Lawrence G. Roberts, one of the networking pioneers involved in Telenet from the start, writes in his article "The Evolution of Packet Switching" (*IEEE Proceedings*, June 1978) that the networks are based on a very different approach to communications than that used for most electronic communication methods.

Telephones, Telex and most radio transmissions, for instance, are preallocation methods. That is, a transmission channel is allocated exclusively to each user, either on a permanent basis as with commercial radio frequencies or on a per-use basis as with the telephone. Even if the assigned user is not communicating at a given moment, no one else may use any part of the channel allocated to him.

An alternative to this method is dynamic allocation which incorporates a time-sharing concept. This is the system used by telegraph and postal systems.

Rather than giving each user a direct link to the person he wishes to communicate with, these systems allocate channels only after a message is received. The messages are moved from point to point (for example, post office to post office) within the system until they reach their destination. The system does not provide a direct, complete link between message sender and receiver.

While both kinds were tried for computer networking, a form of dynamic allocation called packet switching proved to be more effective. This method was developed simultaneously in the mid-1960's in Britain by Donald Davies of the National Physical Laboratories, who named the system, and in the US by Lawrence G. Roberts at the Advanced Research Projects Agency (ARPA), Cambridge, MA.

Both these networks consisted of communications cables connecting minicomputer packet switches. To use the system you call the nearest packet switch, tell it who you want to contact and give it the message you want to send. It chops your message up into discreet packets, each a standardized size. Davies' network, for instance, used 128-byte packets.

The network determines the optimal path for the message, and the computer feeds the packets out to the next switch in that particular path. This method allows each packet to be checked to make sure it has not been garbled and determines the route for the packet to the next switch.

While Davies wrote the original design for the network, European communications people were slow to recognize its value, and Davies was only able to set up a

Reader Service for facing page ✓357.→

The Newest & Most Astounding Arcade Game that TALKS has Just Reached Planet Earth:

BIG FIVE SOFTWARE

P.O. Box 9078–185 · Van Nuys, CA 91409 · (213) 782-6861

Ordering info for all games:

Super Nova, Galaxy Invasion, Attack Force, Cosmic Fighter, Meteor Mission 2, and Robot Attack:

Prices per game: 16K Level 2, Mod 1/Mod 3 Cassette · $15.95
32K Level 2, Mod 1/Mod 3 Diskette · $19.95
Optional Joystick for Model 1 · $39.95

10% discount for 2 items, 15% for 3 or more.

Please add $1.50 per order for postage & handling, Calif. residents add 6% tax.

Outside USA (except Canada) please add $3.00 per order for postage & handling.

We accept checks, money orders, and MC/Visa orders ($2.00 extra for COD).

All games © 1980 by Bill Hogue & Jeff Konyu.

Programs are written in machine language for high quality graphics & sound effects.

Voice & other sounds are playable through the cassette AUX plug.

High scores are automatically saved after each game on the disk versions.

TRS-80 is a trademark of Tandy Corp.

Write for more detailed info about all of our products.

Dealers: All games now available in full color packaging like above, please inquire.

one-switch network to connect the computers at the National Physical Laboratories.

Launching ARPANET

The story at ARPA was different. In 1968 work began on the development of a packet switching network to connect computers involved in ARPA-sponsored research at several major universities. The result was ARPANET which used 50 kilobit per second (kbps) leased lines and minicomputer packet switches. It went into operation in December 1969 with just four client computers. By March 1977 it had 111 connections.

ARPANET proved dependable and demonstrated several advantages over pre-allocation networks. It made more efficient use of the leased lines which, in a period of increasing line costs and decreasing computer costs, means savings.

Since each switch or node checks each packet and can get a retransmission from the sending node if it detects an error, it guards against transmission garbles. It routes each packet separately, so it reacts instantly to overloads and even physical breaks in the transmission lines.

The ARPANET and National Physical Laboratory experiences encouraged others, particularly in France, Germany, Canada and Japan, to start networks. By 1974 five were being planned, and those involved decided it was time to agree on standards so they would be able to interconnect.

By March 1976 they had formed the CCITT, an agency of the United Nations based in Geneva, Switzerland, and agreed to Recommendation X.25, which established basic rules for making networks compatible. Because of this foresight, today there are 35 countries worldwide connected by networks.

At this point the groundwork was finished, and public packet-switching networks began to develop. In most countries these were sponsored either by the national government or the national telephone monopoly. The Trans-Canada Telephone System, for instance, started Datapac, the dominant network in Canada, in 1974.

Things developed differently in the US. According to Roberts, the developers of ARPANET offered to sell the system to American Telephone and Telegraph (AT&T) in the early 1970's in the hope it would become the basis of a public network. AT&T was uninterested.

At that time, however, the Federal Communications Commission (FCC) made some changes in its regulations designed to increase competition in the communications field. These allowed ARPANET to go public on its own in 1973 under the name Telenet.

Competition Emerges

Meanwhile, Tymeshare Corp., Cupertino, CA, a timesharing company, had founded its own private packet switching network to serve its customers in 1969. According to company spokesman Dick Jokinen, "One day someone came along who wanted to use our network to reach their own computers."

Tymshare was interested, but this would make it a common carrier under FCC regulations. In order to comply with these rules, they split Tymnet off as a wholely-owned subsidiary in 1977.

While Tymnet is a packet-switching network and is fully compatible with international standards, it is significantly different from Telenet in several ways. For one thing, Tymnet uses 66-byte packets.

Also, it is more centralized. While Telenet depends on its thousands of nodes for all routing and other computer functions, Tymnet uses a combination of packet

Dr. Lawrence Roberts, The Quiet Revolutionary

Dr. Lawrence Roberts is an easy-going man. To talk to him you would never think him a revolutionary. Yet, he is one of the original developers of the packet networking concept and the designer of the first operational packet system.

Today, as president of GTE's Subscriber Network Projects division at Vienna, VA, he remains intimately connected with the network he started at the Advanced Research Projects Agency (ARPA), Cambridge, MA, in 1967.

Roberts said the groundwork for Telenet was laid while he was with MIT's Lincoln Laboratory, Lincoln, MA. "We had pretty well developed time sharing in the research community in the early 1960s, so I felt that was going to happen in the business world, but I wanted to go further and access computers remotely."

Can Computers Communicate?

The first question to be answer was whether the computers would communicate at all. No one really knew. MIT had a mainframe computer in Lawrence and another in a laboratory in California, and Roberts hooked them together using a leased telephone wire, "to see how the machines would react."

"At that point I concluded it was feasible to link machines," he said. "Then I started worrying about the networking aspects of it."

Eventually he developed the concept of packet networks. He said it is natural for a computer person to think of digital communications in packets instead of in analog streams. Roberts wasn't the only one working on this idea, but he didn't discover that until after he had finished his design.

"There were other activities going on, particularly in England, and we corresponded, but they were not doing any building," Roberts said. "We had already finished our conceptual work when we became conscious of their conceptual work. It also turned out Paul Barron had written some things in the early 1960s for the Air Force, but no one had heard of that until after our network was designed."

Unlike the others, Roberts didn't just leave his network on paper. In 1967 he joined ARPA, a group which, among other things, sponsors research on several university campuses. He interested them in the idea of building an experimental network to link computers, and ARPANET was born.

Selling the System

It wasn't quite that simple, actually. While Roberts and, apparently, his bosses at ARPA believed in the concept, virtually no one else did.

"The communications community (which was telephone at that time) thought it would never work," Roberts said. "They have a certain model and 100 years of working with circuits of their kind, and they couldn't see it."

Roberts met with a different kind of resistance from the universities. "They were violently against it," he said. "They thought I was crazy."

Each university wanted to have complete control over its own system and did not want to share anything. At the same time, none of them believed that any of the others would have anything interesting to offer.

"That quickly changed when they found other people had exciting things and they could do research through the system."

ARPANET paid another dividend in

terms of savings. "We developed ARPA-switching minicomputers and a central bank of four mainframe "supervisors." These keep track of the condition of the system.

When you call Tymnet and tell your local node what timesharing service you want, it tells the supervisor. The supervisor establishes the route your data will follow. The supervisor also steps in to re-route data if a line is broken or some part of the network is blocked.

The nodes take care of establishing packets, checking them for transmission errors and retransmitting them whenever necessary. As with all other packet switch networks, its nodes handle several calls at once, operating like an electronic post office.

Today Tymnet uses more than 700 nodes, services more than 350 host computers and can carry more than 4,000 simultaneous users. It reaches about 250 US cities. Telenet, a division of GTE Corp., Stanford, CT, reaches about 300 cities and carriers 200 timesharing services.

Presently these networks are all using leased telephone lines to carry their transmissions. Roberts estimates that, depending on the kind of data being sent, packet switching makes from three to 100 times better use of its lines due to the efficiency of time sharing.

The extra computer needed for packet switching also costs money, but it is cheaper than the rising cost of transmission lines.

Because of this, packet switching became more cost effective than preallocation several years ago. Roberts predicts that as that gap widens virtually all electronic communications will be by packet switching.

2001

Roberts also predicts the eventual replacement of telephone leased lines with a variety of more efficient alternatives. He says as traffic grows, present trunklines, which work at 56 kbps, will be replaced with T1 digital trunks which operate at 1.544 megabits per second, allowing a great increase in the transmitting capacity of the system.

He also says satellite transmission will become economically feasible for long-

Continues to page 56

NET as a research tool and to access the research sites in the community," Roberts said. "One of the goals we had was to lower the cost of computer resources, and we did that by a factor of three by allowing universities to share resources instead of duplicating them."

ARPANET also quickly dispelled the worries of the communications engineers who had said it would not work. It did not lose packets; it did operate smoothly and efficiently.

Ma Bell's Reaction

ARPANET did not impress everyone, however. One group that was not excited at all was American Telephone and Telegraph, the parent company of Bell Telephone, as Roberts found out when ARPA tried to sell the network to them.

"The Bell lab's evaluation said it wasn't compatible with anything else in the Bell network," Roberts said. "Actually it wasn't incompatible with their network—we were using rented Bell lines—it was incompatible with their thinking."

Bell apparently had decided back in the early 1960s that the Sherman Anti-Trust legislation made it impossible for them to get involved in data networks, so they never did any research in them. They have lived to regret that decision.

Today they are struggling to develop their own packet network under the name Advanced Communications Systems (ACS). Roberts says they are still putting hundreds of man-hours into designing the system and are at least two years away from building it. Meanwhile both Telenet and Tymnet, Cupertino, CA, are operating smoothly.

One way or another digital data transfer will take over, probably in this decade.

"For the home market it will take longer," Roberts said. "In business it is happening very quickly, as businesses go to leased lines connecting buildings full of terminals to the network directly."

> *"Actually it wasn't incompatible with (Bell's) network—we were using rented Bell lines—it was incompatible with their thinking."*

The problem with the home market is the microcomputers are spread out. To solve this, GTE, the Japanese and several other groups are working on digital telephoning. In fact, the Japanese telephone company is presently installing a small operational digital telephone system.

GTE is working on a method to allow 64 kilobit (kb) data transfer through paired wire, an idea which Roberts terms quite feasible. He said the system would actually work at about 200 kb, but would be alternating so you could transmit in one millisecond, receive in the next.

Roberts envisions a telephone-computer terminal combination of the future which would handle both voice and digital data transfer at the same time, to and from different destinations.

It would, for instance, tell you while you are taking one call that someone else is calling. It would get the caller's name and number and tell him you are on a call and either put the caller on hold or call back. If he wanted to send a written message, he could enter it and your phone would store it for you.

If you had a list of people you needed to talk to, you could have your phone send a packet to their phones and find out who is free. Your phone would then make voice contact automatically.

Your phone would also be able to take and store messages and communicate with time-sharing computers over the network. It would be able to tell callers if you are unavailable and when you will become available. It will be able to direct them to another phone where you can be reached.

Roberts said the cost of lines makes it absolutely necessary the two systems, voice and data, be combined, although it is not impossible that voice transmission would remain analog for some time and the two methods would simply share the same cables.

Roberts is working in the forefront of this development with GTE. He is responsible for the development and marketing of hardware that will complete the digital communications revolution.

He remains optimistic about the future of packet switching. "It's my belief that the only way to handle data is to do the economics and the protocol conversions (to allow different machine to communicate with each other)," he said. "It will continue to be the major and probably the only way to handle data. There's really no other way to do it."■

by Bert Latamore
80 Microcomputing *staff*

Ethernet Alive at Stanford U.

Local networking may be the hottest thing in business communications since the telephone. A network changes the microcomputer on an executive's desk from an isolated business machine into an integral part of a company-wide computer system.

It allows an executive in his office in one building to instantly send written information, charts, diagrams, worksheets, etc., to another executive in another office in another building.

"Ethernet is one of . . . 20 to 30 . . . systems . . . ."

It allows him to compose a letter, correct it himself, then send it directly to a spinwriter in the mailroom for printing. It allows him to receive up-to-the-second information from the firm's central data banks on any aspect of his company's activities from parts in stock to the book balances whenever he needs them.

Such a tool creates a large business market, and many firms, from large computer manufacturers like Digital Electronics Corp (DEC) to tiny components makers, have developed networks and are supporting them with the necessary hardware and software.

Ethernet

One of the most interesting of these is Ethernet. Several manufacturers, including Xerox, DEC and Intel have promised to provide support to it.

This network, developed by Xerox, is barely out of the experimental stage, and most components are not yet available for it. An experimental version is running at Stanford University, Stanford, CA.

The Stanford Ethernet is a three megabit system, and the equipment and software used were part of the experimental system Xerox built when developing the 10-megabit Ethernet. Xerox donated the experimental equipment to Stanford, Carnegy Mellon University and MIT. A group at Stanford has been working with the experimental system. It is operating successfully, interconnecting 25 computers on campus. The object of the experiment is to determine whether Ethernet or Ethernet-type network technology has a place on Stanford's campus.

10,000 Computers Expected

The current population on campus of 250 computers and around 2,000 terminals is expected to grow to something like 10,000 in 5 to 10 years. They hope to develop a network or system of networks to allow most or all of these to interconnect. While a single Ethernet system would not be able to connect all the computers spread out around the several-acre Stanford campus, it could be used on a more limited scope in a single building or a large number of Ethernets could be linked together to do the job.

Simple Packet System

Ethernet is a packet system but with a much simpler and more limited method of operation than the intercontinental common-carrier systems. When you send a message through Ethernet, the system chops the message up into packets including addresses indicating what stop on the system the message is being sent to. Then it inserts your packets into the stream of packets running through the Ethernet cable.

So far its operation closely resembles the common network. However, the Ethernet arrangement does not include any packet-switching computers, and all packets follow the same path through a single cable.

Each client computer on the system reads the addresses of every packet going past. If the packet is for it, the computer captures and stores it. If not, it passes it back into the system.

Ethernet by itself is even more limited than that. Ethernet is not a complete system; it provides only the bottom two of the seven levels of "protocols" or software and hardware packages needed for a total network system. What it does do is give you the connection devices to attach your machines to a cable and the interfacing software to allow it to create and address the packets.

It does not include application programs, for instance, which would instruct machines on how to communicate with each other. Ethernet is meant to provide a common network base over which the networking of different firms' machines can be accomplished.

Networks

continued from page 55

distance transmissions when the excess data flow exceeds 100 packets per second between two cities. Because of the built-in 270 ms delay in one-way satellite transmissions this would not be acceptable for transmission of interactive traffic but could be used for batch data transmission.

Roberts also spoke of packet radio with local distribution systems. Local distribution is the most expensive part of a network. In this alternative to land lines, several clients would share a frequency. Each would broadcast to the node only when it actually has data to send, he says.

Tymnet recently announced it was trying a packet radio system in San Francisco and a satellite link between San Francisco and New York City. The experiments follow a January FCC ruling reallocating the 10.55 to 10.68 GHz RE spectrum to digital electronic message services, both for local distribution microwave transmissions and satellite communications.

As the cost of copper continues to increase and the cost of computing drops, it becomes easy to visualize a future in which dynamic allocation networks dominate telecommunications. Cable television (and radio), with its promise of unlimited channels, is growing rapidly. Packet switching could greatly improve its economics.

Even telephone service may go this way. However, Roberts says, voice transmissions cannot be condensed as much as digital transmissions can, so such a change may not be economical in the near future, especially as it would involve the complete rebuilding of a large in-place system. Because of this, electronic mail sent through packet networks may become serious competition for the telephone.

Whatever the future, the networks are here today which provide a doorway to a whole new world.■

by Bert Latamore
80 Microcomputing *staff*

This does not mean that DEC machines will be able to talk with Xerox computers—unlike the common carrier networks, Ethernet does not handle any interfacing problems. These are taken care of by the higher level protocols which the Ethernet package does not include.

Ethernet also does not include any provision for confirming that the receiving machine gets the message sent to it. If that machine is turned off or malfunctioning or if one of the packets are addressed incorrectly, the message will never arrive, and without some kind of confirmation the sender will not know this. This must also be included in the higher level protocols.

Physical Limits

Ethernet has physical limits as well. It is a baseband system as opposed to a broadband system. This is like a telephone line which is a baseband system able to handle comparatively few messages at once while a television cable is a broadband system able to carry hundreds of channels simultaneously.

Additionally a single system cannot be more than 2,500 meters long. You can connect several Ethernets together using "gateways," however. These are devices used to interface different networks. Alternatively, an Ethernet system can be connected to a different kind of network, including the common carrier networks, using gateways which would translate between Ethernet and the other network's protocols.

Not On Market

Part of the problem with determining the economics of the system is that the commercial hardware and software is not yet on the market and not expected to be for 18 to 24 months.

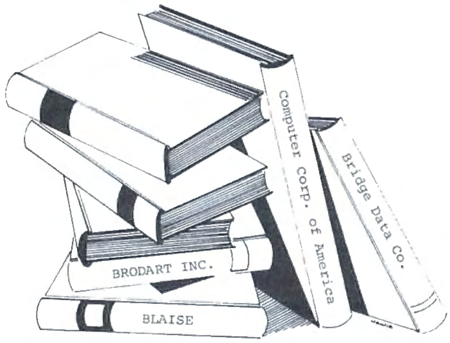
Meanwhile, Ethernet is only one of some 20 to 30 possible networking systems, each with their own advantages and disadvantages. Some, like the Tymnet local network which is really a smaller version of its common carrier network complete with packet switches, have much greater capability in terms of area covered and number of machines it can handle.

Others, like Omninet by Corvus Systems, which does not even use coaxial cable but runs on simple twisted pair wire, have less capability than Ethernet but are probably less expensive.

The question is what system fits the individual's needs and pocketbook. ■

by Bert Latamore
80 Microcomputing *staff*

Four Hundred Databases Await Your Microcomputer



Compuserve, The Source and, recently, the Dow Jones Electronic News, have received a great deal of press attention as interactive electronic information services for general microcomputer use.

They are, however, only the start of the list of available data bases. Several hundred of these exist in 35 or more countries.

Most were not originally designed for use by home microcomputers; however, the networks automatically handle most compatability problems, and as the microcomputer market grows, more data base publishers are becoming conscious of compatibility and are making efforts to take care of formatting and other residual problems.

It is impossible to list all of these services here. Both Telenet, Vienna, VA, and Tymnet, Cupertino, CA, publish directories showing most of the services they connect to. If you want to subscribe to a service, you must deal directly with that service, not with the network. All network charges are included in the fees you pay for using the service.

Here are a few of the data bases:

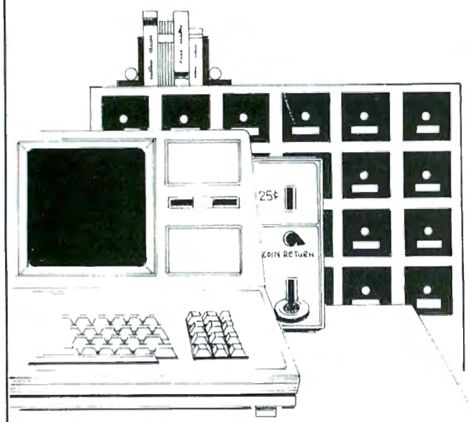
- Berkeley Solar Group, Berkeley, CA, offers solar energy analysis, weather data and other services for the solar energy industry.
- Blaise, British Library, London, England, includes medical data bases, a listing of all British-imprint books acquired by the Library of Congress and a document-ordering service.
- Bridge Data Co., St. Louis, MO, gives common stock quotes and other services to investors.
- Bodart, Inc., Williamsport, PA, offers 800,000 English-language monographs, including a descriptive list and an on-line ordering service.
- Comcap Computing Systems, Oakland, CA, provides technical time-sharing services for civil and structural engineers and land surveyors.
- Dr. Dvorkovitz & Assoc., Ormond Beach, FL, offers what is reputedly the largest data base of licensable technology in the world.
- E. H. Boeckh Co., Milwaukee, WI, provides building cost estimates for insurance appraisers, developers, contractors, etc.
- Florida Computer Inc. (FCI), North Miami, FL, has daily and historical data on the United States, Canadian and London commodities markets for speculators, hedgers and brokerage houses.
- Lockheed Information Systems, Palo Alto, CA, has more than 100 data bases holding more than 30 million records giving references to published articles, conference papers and technical reports on a wide range of subjects.
- National Library of Medicine, Bethesda, MD, provides medical information including a 500,000-citation file of abstracts and references and is one of several medical services available.
- *New York Times* Information Service, Parsippany, NJ, carries abstracts from *The New York Times* and more than 60 trade publications.
- QL Systems Limited, Ottawa, Canada, has 40 or more data bases on Canadian subjects, including Canadian law, aquatic environmental data, fisheries, mining technology, energy, business, metals and Parliament.

"Several hundred . . . databases exist in 35 . . . countries."

- Weather Systems International Corp., Bedford, MA, provides past, present and forecast weather data and past and present sports and business information.
- Westlaw, West Publishing Co., St. Paul, MN, gives full text and headnotes of decisions in all federal courts and in courts of all 50 states. ■

by Bert Latamore
80 Microcomputing staff

Micro for Hire—Four Bits or Eight



Ask her, and Head Librarian Marion Boben will tell you that she isn't at all threatened by the machine. The fact is, she's fascinated by it. It's been earning extra money for her library, and she hopes it stays right where it is—near the card catalog, just within sight of her checkout desk.

With the exception of its coin slot, the cassette-equipped Model I is not very exotic. There is no disk drive, no expansion interface, no printer. The '80 has generated a lot of interest among library regulars in Strafford, a suburban community of 23,000 just outside of Philadelphia. It even achieved date-line status for the town on the Associated Press wire when a story about the library's new addition appeared in several newspapers across the country.

There aren't many microcomputers in public access settings yet and even fewer for rent. In Strafford, however, if you've got the money, the Tredyffrin Public Library has the microcomputer: four bits for 15 minutes or $2 an hour.

Who is using the rent-a-brain? Marion Boben says it's hard to categorize the users. There are 15-year olds who plunk down 50 cents for 15 minutes of Space Taxi, local businessmen getting their feet wet in computerized Accounts Receivable and general ledger bookkeeping, middle-aged parents trying to become computer literate before their children, and students from the Junior and High Schools doing their computer programming class homework.

Marion Boben's machine has been swallowing change at a respectable rate since last December. Use has fallen off slightly during the summer, but she's sure the soothing silence of the library will once again be broken by the clink of dropping quarters come fall.

Straightforward Installation

The Model I installation in the Tredyffrin Public Library is straightforward. A TRS-80 is connected to a power-interrupt device controlled by a timer. Drop 50 cents into the slot and the '80 comes alive—for 15 minutes. Three minutes before shutdown the user is warned that a crash is imminent. At that point more money can be deposited or operations terminated. The microcomputer has not been specially modified: it has no kid-proof keyboard or protective enclosure. The only precaution taken is bomb-proofing the coin box.

The one-armed '80 is the brainchild of a Philadelphia firm called Compuvend, Inc. Compuvend, a division of Computer Systems, Inc., has placed the coin-operated micro in the Tredyffrin Library strictly on a trial basis. Compuvend's president, Will Zimmerman, is encouraged by the results. Zimmerman told *80 Microcomputing* his new company has been besieged by requests for information since the AP article about the mercenary microcomputer appeared. One group that has exhibited exceptional interest is the electronic arcade game industry.

Response has been so encouraging Zimmerman is planning to offer several versions of his coin unit. The Series 10 will accept only coins, the Series 20 will be a "dollar bill recognition unit" and, eventually, a version that will accept credit cards will be marketed.

"Zimmerman told 80 Microcomputing his new company has been besieged by requests for information since the AP article about the mercenary microcomputer appeared."

Coin-Op Printer

Zimmerman has other plans as well. He hopes to have a companion coin-operated printer available soon for those who need a printout and are willing to pay for it. In addition, he intends to make large data bases such as The Source available on an auto-dial, cost-per-minute basis from public access settings.

In the meantime he is dickering with Tandy, Apple, Atari and other microcomputer manufacturers for the right to sell their equipment on a non-exclusive dealer basis. Two of these deals have been secured already, but Zimmerman isn't willing to name them.

Zimmerman supplies a variety of software with his system, selected to be of interest to homemakers, businessmen and students. The machine Marion Boben is using came with a 20-program library that included games, Basic language tutorial tapes, statistical analysis programs, home-budgeting routines, vocabulary drill programs and a payroll program. Users are free to purchase software from other sources if they desire or write their own.

Though only one prototype is being field-tested by Compuvend at present, Will Zimmerman is confident that great potential exists in the public access market. He told *80 Microcomputing*, "My gut feeling is that this (the rent-a-computer) market could be as lucrative as the photo copier market has proven to be in public settings." He also said that since libraries are in the information business and the microcomputer is the tool of the information age, its placement in libraries is inevitable.

While the information age may have finally arrived at Marion Boben's library, its premier tool, the microcomputer, is being used mostly for game playing and teaching computer programming. Marion Boben is not disheartened, though. In her opinion, public access computing may take the microcomputer out of the realm of mystery and place it in the hands of the masses, providing, of course, that they have the ability to pay. ■

by Chis Brown
80 Microcomputing *staff*

✓See List of Advertisers on page 354

College Board Database Available on CompuServe

CompuServe now offers electronic help to college-bound microcomputer users trying to find and qualify for admission to the right school.

The databank, provided by the College Board, Princeton, NJ, contains information on choosing a school, finding financial aid, preparing for the Scholastic Aptitude Test and finding adult education.

Prospective additions to the service include excercises related to building decision-making skills and a schedule of the activities associated with college selection, application and admission.

The College Board is an association of more than 2,500 secondary schools, colleges and educational associations, which provides guidance and admissions information to college-bound students and adults nationwide and helps schools and colleges respond to the educational needs of students.

$5 An Hour

The College Board information will cost CompuServe subscribers $5 per hour of access time nights and weekends. Weekday business hour access is also available.

CompuServe, Columbus, OH, is a general database service offering the latest news from major newspapers and a national wire service, financial information on stocks and commodities, home banking, electronic mail and real-time communications, computer games, family information and computer time-sharing for programming activities.

Available by local phone call through the common-carrier packet networks and its own private network, CompuServe is a division of H&R Block.■

Wall $treet Week Transcipts On Line from Dow Jones News

Transcripts of the popular half-hour financial news program *Wall $treet Week* are now available electronically the Thursday after each show is aired from the Dow Jones News/Retrieval Service.

The show, hosted by Louis Rukeyser, is seen by about 10 million people in 250 markets in the United States, Canada, Guam and the Virgin Islands.

"The electronic news service will make transcripts available significantly sooner than printed versions sent by mail."

The electronic news service will make transcripts available significantly sooner than printed versions sent by mail. Transcripts of the last four programs aired will be available in the databank.

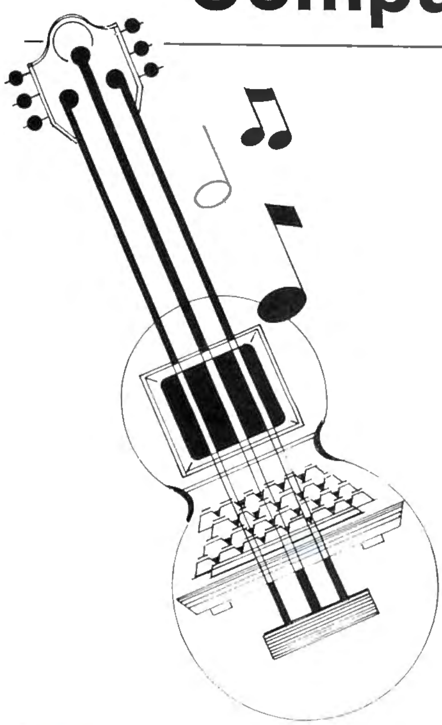
PBS Center Agrees

The service is being made available through an agreement between Dow Jones and the Maryland Center for Public Broadcasting. Access to the data bank will cost 50 cents per minute during business hours and 25 cents per minute nights and weekends.

The Dow Jones News/Retrieval Service provides business and financial news information on command through standard time-sharing terminals and personal computers, including TRS-80s. The Wall $treet Week data base is formatted to be compatible with TRS-80s.

Dow Jones is the publisher of The Wall Street Journal, Barron's, Book Digest magazines, the Dow Jones News Service wire, The Asian Wall Street Journal, and the 20 community newspapers of the Ottaway Newspapers, Inc. chain.■

More than Math—Computers Meet Curriculum



When the 4,000-strong student body returns to the Lyons Township, IL, High School this month, they will begin an experiment in computer literacy education that will involve every student from math whizzes to special education students and every class from physics to physical education.

Over the summer, teachers and programmers have been working together to implement a new school policy requiring 10 percent of all class work to be done on computer.

The Computer Department, meanwhile, has been installing 150 TRS-80 microcomputers, most of them 16K Model III machines.

An integral Part

The idea, according to Norman Weinberg, the school's head of computer services, is to make the computer "an integral part of much of the course work."

"This is as it should be," Weinberg said, "because computers are an integral part of much of what's going on in the world today."

The program is a major change of approach for the school, which, until this year, offered only a few courses in computers to selected students as electives.

"In the past, generally there would be too few machines available for the kids, they would have to queue up and would quickly lose interest," Weinberg said. Otherwise, "the students have been gung-ho on computers ever since they had anything available."

One school source said students have been "breaking down the doors to get at the computers" ever since a May seminar on the new program ran for the staff.

Weinberg said the old approach meant the computer course had to compete with foreign languages, art, history, and other elective courses for student registration. The new approach gives computer literacy the status of a basic requirement like English and math.

Few Problems

So far there have been few problems implementing the program. Staff involvement started in May when the computer department ran a four-day seminar on the TRS-80 for teachers and other education professionals. Weinberg said about 250 of the 300-person staff attended on a volunteer basis, and they were all enthusiastic at the end of the experience. Since then they have had no trouble with teacher acceptance of microcomputer use.

As the next step, teachers were asked to submit recommendations on how they could use the microcomputers with students. Ideas ranged from drilling students on spelling and punctuation in English to teaching basic guitar cord fingering in music.

One of the more imaginative uses came from the Physical Education Department, which is going to use its computer time to have students derive personal exercise programs based on computer analysis of their scores on several physical fitness tests, including the President's Test on Physical Fitness.

Continues to page 62

Color Computer Gets Educational Software

The TRS-80 Color Computer will be going to school this fall with a new learning program called the Talk/Tutor system.

The system, developed by Dorsett Educational Systems, Inc., Norman, OK, and licensed to Tandy/Radio Shack, Fort Worth, TX, presents numbers, upper and lowercase test, high-resolution graphics recorded by a special television camera and processor and high quality audio.

Only a standard TRS-80 Color Computer and television set and a single-track audio cassette player are needed to retrieve lessons prepared on this system. Students interact with lessons using the computer keyboard or optional joysticks.

Exciting Program

William T. Gattis, director of Radio Shack's Education Division, called it "One of the most exciting tutoring programs in a long time. We plan to offer a variety of instructional programs for both the home and classroom that run on the new system."

He said the first three of these will probably be released during the third quarter of this year.

Dorsett will supply the video camera control and image processing equipment to Radio Shack, which will use the equipment to develop teaching programs under the licensing agreement between the two firms. Dorsett will also supply Radio Shack with some finished tutorial programs in reading, mathematics and other subjects.

Radio Shack will not sell the video equipment, but Gattis said that equipment may be made available to educational publishers under a sublicense agreement allowing them to develop and sell educational programs for the Radio Shack Color Computer.

"We have a large-scale effort under way to develop instructional programs that support the use of our TRS-80 Model I and III computers in the classroom," Gattis said. "The Talk/Tutor system now extends this effort and this commitment to include our TRS-80 Color Computer." ■

Math

continued from page 61

During the summer the teachers are working with professional programmers the school has hired to program these ideas. Weinberger said they hope to attain the goal of 10 percent of all class material on computer within another year.

A Little Advice

Weinberg said the TRS-80s are permanently installed in special classrooms. Each room is networked to the others using two Tandy multiplexers connected to a 48K, twin-disk master machine (which the teacher can use to monitor and interact with student activity) and to a printer.

There are four-and-a-half classrooms of 16K machines and two-and-a-half of 48K twin-disk machines, intended for use in programming classes. Weinberg said they plan to teach Basic and may offer Cobol.

TRS-80s were selected because they seemed to do the best job for their price. "Some people wanted fancier machines," Weinberg said, "but that would also mean fewer machines."

He said he hopes other school systems watch the Lyons Township experience closely.

"They should take our approach and try to design systems so they reach the maximum number of students," he said.

To counter teacher resistance or concern, he said, the best approach is to involve the teachers at the start. "We got 250 of them to get their hands on the machines and scared away the boogeymen."■

Don't Pay High Prices for

Get top quality & fast service at the lowest prices,

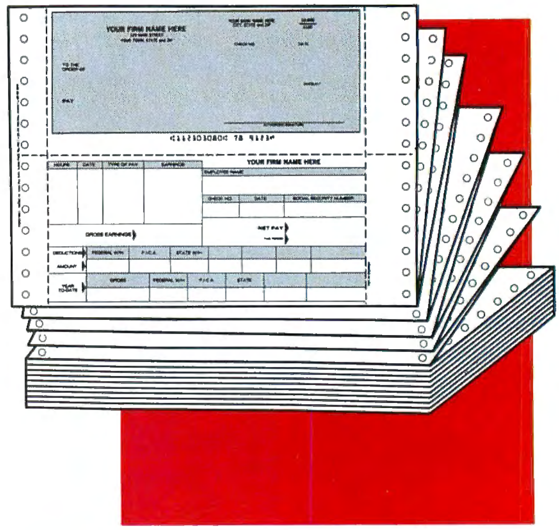
"carbonless" PAYROLL CHECK with PAY STATEMENT

Product 9002 PAYROLL CHECKS are designed to be used with the RADIO SHACK* payroll program #26-1556. Size 9½"x7" (7½"x7" detached.) Check size 7½"x3½". Prices include printing your heading, bank name and number, and MICR encoding. Attractive green fine line design on white MICR Bond stock. Available in single (white) or duplicate (white, canary) continuous sets.

QUANTITY	SINGLE Product 9002-1	DUPLICATE Product 9002-2
6,000	$149.00	$299.00
4,000	110.00	215.00
2,000	65.00	124.00
1,000	43.50	75.00
500	32.50	48.95
250	27.50	37.00

Versatile, easy to use
Special 9½" check width allows for use on adjustable and fixed pin printers. Duplicate checks use "carbonless" paper to produce clean copies automatically.

Maintains good employee relations
Pay Statement and Pay Check given to employee as convenient unit. Space to spell out exact earnings, standard and special deductions.



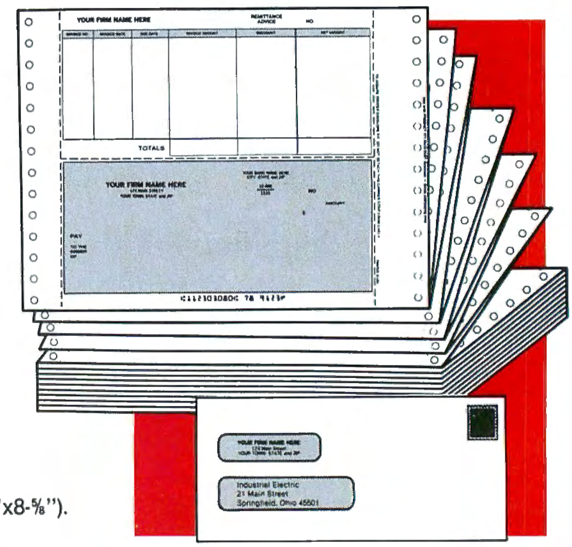
"carbonless" ACCOUNTS PAYABLE CHECK

Product 9023 ACCOUNTS PAYABLE CHECKS are designed to be used with RADIO SHACK* accounts payable programs #26-4505 and #26-1554. Size 9½"x7" (7-⅜"x7" detached.) Check size 7-⅜"x3½". Prices include printing your heading, bank name and number and MICR encoding. Available in green fine line design on white MICR Bond stock. Available in single (white), or duplicate (white, canary) continuous sets.

QUANTITY	SINGLE Product 9023-1	DUPLICATE Product 9023-2	Product 775 DU-O-VUE® Envelopes
6,000	$149.00	$299.00	$140.95
4,000	110.00	215.00	98.50
2,000	65.00	124.00	53.50
1,000	43.50	75.00	29.50
500	32.50	48.95	17.95

Versatile, easy to use
Special 9½" check width allows for use on adjustable and fixed pin printers. Duplicate checks use "carbonless" paper to produce clean copies automatically.

Eliminates Envelope Addressing
Assures privacy, checks are designed for use in Product 775 DU-O-VUE Envelope (3-⅝"x8-⅝"). Made of opaque gray stock.



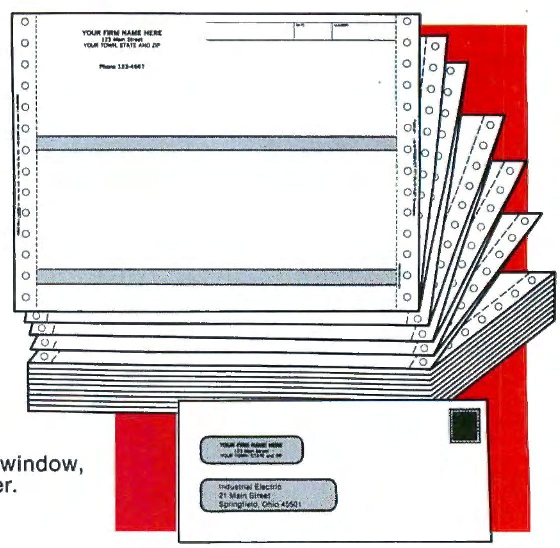
"carbonless" Multi-Use STATEMENT/INVOICE

Product 9050 STATEMENT/INVOICE is designed for use with RADIO SHACK* accounts receivable program #26-4504. Size 9½"x7" (8½"x7" detached.) Prices include printing your heading in black ink. Available in single (white) or triplicate (white, canary, pink) continuous sets.

QUANTITY	Single Product 9050-1	Triplicate Product 9050-3	Product 771 DU-O-VUE® Envelopes
6,000	$152.00	$348.00	$120.75
4,000	108.00	248.00	84.50
2,000	62.50	135.00	44.50
1,000	43.50	78.50	24.75
500	32.50	49.95	14.75

Versatile, easy to use
Special 9½" width allows for use on adjustable and fixed pin printers. Use as a statement, invoice, credit memo, collection notice, etc. . . Triplicate sets have "carbonless" paper to produce copies automatically.

Saves Addressing Time
Product 771 DU-O-VUE® Envelope (4⅛"x9"). Your printed statement heading shows through upper window, customer's name through lower.



Continued from page 7

the stops to make my short visit one I'll never forget. I had an opportunity to talk with the Prime Minister of the country; I went to a party at which I met most of the active hams; was given a tour of the whole island; went on a wonderful scuba-diving excursion; and was shown the red carpet treatment all the way around.

By the time you're reading this I'll have visited South Africa, Swaziland, Lesotho, Kenya and Tanzania on a short trip in August. I'll take some pictures and have a lot to tell you about computing in Africa. I'll be addressing the Microcomputer Exposition in Johannesburg, as well as the local computer clubs, ham clubs, and Apple dealers. I understand there are already 5,000 microcomputers there, and things are starting to move ahead rapidly. We've been distributing Instant Software in South Africa for two years now.

This will be my first trip to South Africa, though I've been to East Africa (15 years ago on a hunting safari). My ham contacts have provided me with friends in all of these countries for some 40 years now, so I'm looking forward to meeting them personally. ■

80 Microcomputing

To the best of my knowledge, *80* can lay claim to being one of the all-time successful technical publications. Imagine the situation: Here is the first magazine on a single manufacturer's product, that has no connections to the manufacturer and is not even distributed in the manufacturer's stores. Yet, its circulation is expected to pass 100,000 before the end of this year!

It has few rivals among technical publications, running over 300 pages per issue since early this year. In just the first six months of this year *80* published 207 feature articles. To put that into perspective, compare that with second place *Microcomputing* with 140 articles and *Byte* with only 73.

The main reason *80* has been such a success is *you*, the readers. You have been telling your friends about *80* and buying products advertised in the magazine at a rate no other magazine can even come close to. Many advertisers have bragged to me that they are getting five to ten times more sales from *80* than they are able to get from any other magazine. When I ask them for a letter to that effect they put me off, saying they don't want *80* to fill up with ads and cut their sales. They feel that advertisers in what is primarily a monthly catalog just get lost, and they have the sales figures to prove it.

Now I'm not putting *Byte* down. No, I'm proud of *Byte*... after all, it was my original idea and I spent six months of my life getting it started. For that matter, most of the crew of *73*, my first magazine, worked hard on *Byte* to get it off the ground. I think we all take some pride in my ex-wife being able to sell it to McGraw Hill for a reputed $7 million. Talk about a coup!

When you help *80* to grow with more subscribers, you get more articles, more information and more programs. It doesn't hurt to make the extra effort and mark up the Reader Service card and send it in. Many advertisers put great store in these responses. They should, for these replies are responsible for about half the monthly sales the magazine develops.

A recent poll of *80* readers showed a median age of 37 years, a median income of $32,000, and an average investment in computers which is difficult to believe. You, the readers of *80*, claim you are spending about $61 million per month on your computer systems. Most of the purchasing in the higher dollar volumes is for business and educational systems, but you can bet that the purchasers are reading *80* ads, looking for the best bargains they can find.

Kilobaud Microcomputing

We also publish *Kilobaud Microcomputing*, a magazine for the computer owner of *any* type. Like *80, Kilobaud* is aimed at the average computer owner, not the computer scientist. TRS-80 coverage in *Kilobaud* is minimal, mostly leaning to articles which may be too technical for the average *80* reader. *Kilobaud Microcomputing* is strong on Apple, Atari, OSI, and the other popular microcomputers.

Kilobaud was started in 1977 and has about 100,000 circulation. The tremendous success of *80* has eclipsed *Kilobaud* to some extent.

Why two computer magazines? The main advantage is directed at smaller firms in the field. By splitting our readership into two groups, advertising is affordable to even the smallest firms. I feel that this is important to the entrepreneurial spirit which makes our country strong. One large magazine would stop many small firms from advertising and we might lose out on another Apple.

Thus, smaller advertisers can select a magazine which reaches their particular customers and larger ones can use both our magazines and reach everyone. This gives you the opportunity to find new and exciting gadgets and programs put out by small firms.

Desktop Computing

While this new magazine is being written and edited for the non-technical businessman, I think you'll find the articles to be of substantial value. The articles will be written from the viewpoint of the average computer user rather than the manufacturer or hobbyist. We'll be covering the use of small computers in all sorts of businesses, schools, and so on.

If you are aware of any successful mi-

EDUCATION 80

Continued from page 8

involved in the action. The rectangular box may have more than one arrow leading to it but only one can lead from it. There is normally a box for each program statement but there are times when one box will be used for a series of statements.

That's all there is to it—except for the arrows. They point from box to box, the same way the program moves from statement to statement.

There is one situation which seems to cause a small problem. You will find For... Next loops handled in several different ways. The approach I find the least confusing is shown in Fig. 2. This is an illustration of the statement:

```
FORX = 1TO10:LPRINTX:NEXT
```

As you can see, when the For... Next loop is charted this way it is actually hard to misunderstand!

Rules

Flow chart rules are simple. First use the right kind of box for each program statement. Second, join the boxes with arrows which point in the direction of program execution. Actually, it is even easier to read a flow chart than a program. Simply go from box to box as the arrows point and answer the question at each diamond box.

A flow chart can be made for an entire program or for a section of a program. It can put the design of a program into an understandable perspective. After you make a broad general outline of a new program, take the time to make a flow chart before you start writing; charting time can be saved many times over by eliminating a lot of debugging and rewriting later.

Making a flow chart can also be extremely helpful when you are working on a program someone else has written. After all, modifying is really rewriting, and it is just as important to understand that program as one of your own. ■

crocomputer (or desktop minicomputer) system, we're interested in getting an article on it. Readers will want to know what hardware was used and why, what software, how much trouble it was to get running, how long before it was a part of the business operation, what problems were encountered and how they were resolved. One key point will be the benefits the system brought about, such as savings in personnel, the ability to provide better service to customers, speeding up service, and so on. Readers will want to have some estimates of the bottom line—cost vs. savings and improvements in the business.

Be sure to write clearly and simply, leaving out all of the technical jargon. Even skip bits and bytes, if you can. I realize that dedicated computer people speak computerese without even realizing that they are doing this and are compulsive about it. We pay generously for articles as long as we don't have to edit out technical buzzwords and put in explanations of terms you just couldn't keep yourself from using. Double space your copy, upper and lowercase type, good photographs help, generous margins for editing, and send it in. Publish or perish.

Desktop will be the first computer magazine you'll be able to show to your noncomputer friends so they can see what all the excitement is about. They'll be able to read and understand this magazine, as it will be written without the computer jargon. I hope you'll interest them in subscribing. We're looking for a 100,000 circulation for this new magazine before the end of 1982, but we can't do it without a lot of help from you.

> *"A recent poll of 80 readers showed . . . a median income of $32,000."*

Load-80

A couple thousand readers a month are now sending for Load-80 cassettes, which contain the major programs in each issue of *80*. These are still selling for $9.95, though I notice that one of the Apple magazines is selling a similar dump of their programs for double that with far fewer programs, too. Having done my share of keying in programs for magazines, I've been through the aggravation of trying to find my errors. This has often taken more time and sweat than the original keying. What a relief it is to put in the cassette, load the program into my TRS-80 and then save it on a disk for instant use when I want it.

The Encyclopedia

Beautiful set of books, these; I think you'll like them. With the success of Load-80 making the programs from each issue of this magazine available on cassette, similar plans are in the works for the Encyclopedia. There are a lot of first-rate programs in each volume, creating a need for an easy way to get the programs into your computer. Watch for an announcement. Of course, if your time is without value and your aggravation level extremely low, perhaps you'll want to key in the programs.■

DEMAND THE MOST FOR YOUR WORD PROCESSING DOLLAR?

THEN MOVE UP TO PROSOFT'S NewScript . . .

* Easy-to-use Full Screen Editing
* Typeahead — never loses keystrokes
* Headings, Page Numbering, Centering
* Form letters, Big Documents
* Global search and change
* Double-Width, Underlining
* Sub-scripts, Super-scripts
* Table of Contents, Indexing
* Proportional font right-margin justification on 737, 739, L.P. IV, D.W. II
* All 12 MX-80 fonts + underlining
* Italics on GRAFTRAX MX-80
* Supports Diablo, Microline, Anadex
* 160 pages of excellent documentation includes "EZEDIT", "EZSCRIPT", self-study tutorial, and hundreds of examples
* Runs under TRSDOS, NEWDOS, NEWDOS/80, LDOS, DOSPLUS.
* Requires 48K TRS-80 with one disk drive.
* Specify: Model I or Model III.

NewScript	**$ 99.95**
Mailing Labels Option	**$ 29.95**
Special: NewScript + Labels	**$115.00**

Order from your local dealer, or from:

PROSOFT

✓441

(213) 764-3131

Dept. C, Box 839 • No. Hollywood, CA 91603

ORDERING INFORMATION: We accept checks, MasterCard, VISA, Money Orders, C.O.D. (add $2.00) and even cash. Please, no Purchase Orders. Calif. residents add 6% tax, 10% outside North America.

NOTE:
Some features work only if your printer has the mechanical capability.

✓See List of Advertisers on page 354

80 ACCOUNTANT

by Michael Tannenbaum C.P.A.

"... simplicity of operation seems to have been the major objective of the Bookkeeper's designer."

One of the most unexpected events at Comdex 81—The National Spring Conference & Exposition for Independent Sales Organizations held in New York, June 23-25—was the discovery of a new accounting package. The Bookkeeper is available from Universal Software Studios Inc., 179 Smith St., Gallatin, TN 37066. This system, written in Cobol for the Model II, offers an integrated accounts payable and receivable, payroll and general ledger.

To the best of my knowledge this system has a unique file structure. Only one master file is maintained for all customer, employee and vendor names and addresses. This means growth in any application file will not cause the system to run out of space. This is an important advantage in a floppy-disk system.

Even more interesting is the elimination of transfer entries between applications.

Data is entered directly into accounts receivable and payable ledgers. At the same time, a simultaneous entry is created in the daily activity journal. This is the only journal contained in the system. Entries in the daily journal are automatically balanced. When the daily journal is printed, a summary journal entry is automatically created and posted into the general ledger. In this manner, control is developed over daily activity and the sales and purchase journals are eliminated. Since the general ledger has been updated it would be possible to generate an estimated financial report.

The need for a daily financial report is questionable; however, the ease of summarizing for reporting will certainly facilitate preparation of regular monthly financial reports. In fact, simplicity of operation seems to have been the major objective of the Bookkeeper's designer. All accounting applications are called from a single menu. As you might expect, this menu is quite cluttered. Despite this, it is easy to select the appropriate application program.

Application programs are grouped into five areas: general ledger, payroll and special functions; and accounts payable and receivable.

Under each of these areas, tasks are preceded by a code number. For the general ledger functions area, there are 10 tasks numbered 1-10. Under payroll, there are five numbered 20-24. Under accounts payable, there are three numbered 40-42. Accounts receivable has five and there are eight special functions. To extend the system into applications such as forecasting, a provision is made to call a submenu. In all, 32 different functions can be called from the main menu.

To familiarize potential customers with the use of the system, Universal Software offers a demonstration disk. Accompanying this disk is a well written 80-page manual which serves as an instruction guide to the system's features. Naturally, the demonstration disk does not contain all the programs required to maintain the accounting system. Missing, for example, is the end-of-the-month closing program.

Universal Software has provided a complete roster of employees, chart of accounts, customer and vendor lists. With all data in place, it is quite easy to prepare an impressive list of reports, ledgers and sample checks. I gather from their material that the demonstration disk may be used as a training aid by new customers. If so, it is most effective. It provides a complete introduction to the system and its unique features.

Since much of the system activity will be devoted to file maintenance, the programmers have attempted to make file updates, additions, and deletions as easy as possible. On entry into the file inquiry mode the message appears "File Number ?" A file can be selected by keying in 99999, accessing an alphabetic search routine. Typing in as many letters as you know will cause the program to search the file and stop at an account that has the letters selected. Since that account will be displayed in context, the missing account number will be quickly located. If the desired account is not on the screen, you can backscroll or advance the display. If the file is located or the search proves fruitless, the escape key provides an exit.

If a new name and address account is required, keying in 99998 will call the new-account entry routine. The account number will be automatically assigned. Once this has been completed, the computer will request you to designate how the file will be used. Keying in a response to that question displays the data entry format.

Once a valid account number is entered, a sub-menu is displayed to call the application desired and the data entry or change the routine. Within an application the same format is used to enter data or inquire about data previously entered. If the inquiry mode is selected, it is necessary to step through each line on the screen before the next invoice is displayed... a time consuming process. After the last item on file is displayed, the program automatically returns to the "File Number?" screen.

After 37 pages of descriptions, explanations and sample exercises, most people will be glad to follow the author's advice to "take a break."

Although there are many automatic features such as self-balancing payable and receivable entries (the offsetting journal entry to payables is automatically made when a purchase invoice is entered), the system never seems to run out of control. All entries and correcting entries are logged and appear in the daily journal. A printout of this journal serves as an audit trail to the original transaction. If any entry is required to a sensitive area such as the payroll ledger or a general ledger account adjustment, a user password is required. If the resulting general ledger adjustment throws the books out of balance, no financial reports can be printed until the imbalance is corrected.

I noticed entries into the system do not generate any feedback control totals. Such totals will be required if a post-invoicing entry to accounts receivable is made. Since I was working with an incomplete demonstration disk, an invoicing module was not included. I would expect such an option to be obtained unless the invoice volume was quite small.

The financial reports generated by the system include comparatives for the month and year to date. Although unexplained by the demonstration manual, the chart of accounts appears to have a "level" coding provision. If this provision is similar to other "level" coding methods

I have seen, it should be possible to group the accounts when printing the financial statements. Another possible use for this feature might be creating departments. If these presentation methods are required, the vendor should be contacted for clarification.

In addition to the financial statements, the system contains a budgeting module. This module permits entry of a budget for each account on a month-by-month basis. A comparative budget report can be printed to compare the actual results with the plan.

Within the limitations of the demonstration disk, I liked the system. It is easy to use and appears complete. At $2,495 it is not cheap but for a floppy disk Model II system, it is certainly worth looking at if an integrated system is desired. Unfortunately, it is not suitable for a public accounting practice. The reports generated do not meet the standards of reporting required and the system still does not retain detail beyond the current month. However, for a small business, the system might be ideal.

Tandy General Ledger Package

The professional accountant may have an opportunity to use the new three-disk Radio Shack general ledger package. Although I received it too late to review for this month's column, a quick look through the system manual indicated this system could prepare a number of reports including: balance sheets, profit and loss statements, statement of changes in financial position, analysis of changes in working capital, cash flow statement, and statement of retained earnings.

Next month's column will be devoted to an evaluation of the Radio Shack system.

Cornucopia Software

Another late arrival, but usable immediately, was an updated version of the Cornucopia Software Editor for the Model II. The original version reviewed last month did not integrate into Model II Scripsit. The new version does...and what a pleasure.

To use the new version, all that is required is to place the cursor over the document to be proofed and press M. The Microproof program takes over. Your document is read into memory and compared against the 50,000-word dictionary. At the conclusion of the comparison, a list of potential problems is presented on the CRT. After pressing Enter, each word in question is displayed and the user can:

- Add the word to the dictionary;
- Bypass the correction procedure;
- Correct the error; and
- Display the word in context when the correction procedure is in progress.

Once the selection process is completed, Microproof adds the new words to the dictionary and automatically corrects the misspellings. This is accomplished on the screen. Page by page, Microproof reads in the document and highlights the word being corrected. If the display-in-context option had been selected, a prompt "replace with" is displayed in the highlighted area and the program pauses. After the document is completely corrected, the Scripsit menu reappears.

"With the software available for the micro, powerful packages for manipulating data can be obtained cheaply."

This editor is a valuable addition to Scripsit and easy to use. During the limited time I have used the new Microproof, I have found only two limitations. The file to be edited must be on the same disk the Microproof program is located and the repagination utility must be run before editing if double spacing is used. Apparently the entire file must be in a consistent format for the editor to function. For those interested in this system, contact Cornucopia Software, P.O. Box 5028, Walnut Creek, CA 94953. An integrated version is also available for Model I and III users.

Mainframe Connections

All the new software packages—accounting and non-accounting—have made the Model II a valuable stand-alone computer. However, it has not linked with the IBM mainframe. In the July issue of *80 Microcomputing,* Ed Juge of Tandy announced the availability of three software packages that are of interest. These packages—Reformatter Cat. #26-4714, 3270 Emulator Cat. #26-4715, and 3780 Emulator Cat. #26-4716—deserve more than a casual mention at the end of Ed's column. Properly employed they will provide the Model II with the ability to integrate into a large mainframe system.

Consider the potential for a firm. With the software available for the micro, powerful packages for manipulating data can be obtained cheaply. If the data can be obtained from the firm's data base easily, the problems of entering data for applications such as tax planning and projections can be significantly reduced. Since these applications are usually interactive, "what if" projections must be done on line. Large numbers of on line users could degrade response time affecting other applications. The ability of the micro to "off load" the planners from the main frame could significantly improve the data processing system for all users.

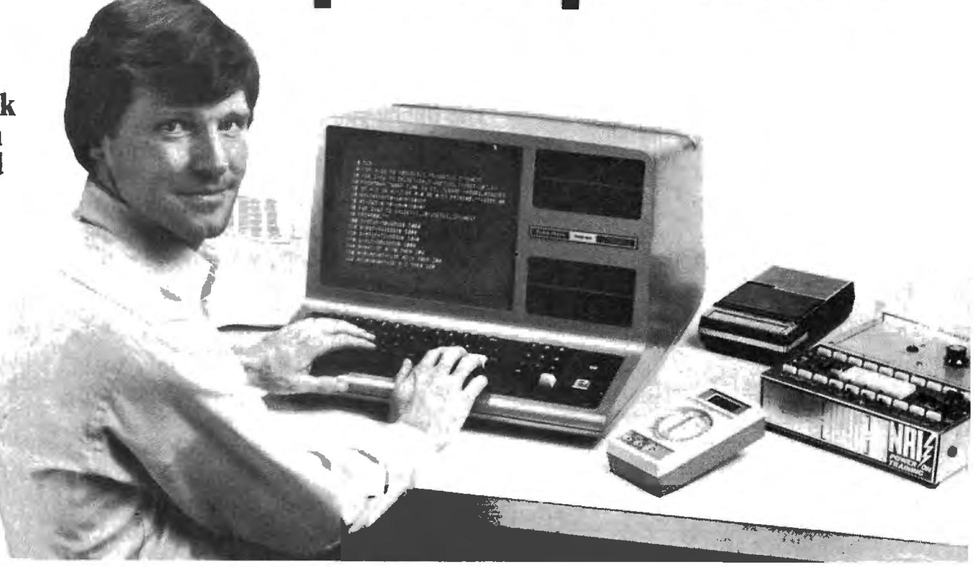
This is called "distributed data processing" and it is at the core of the new communications networks announced by most major computer vendors. It is surprising Tandy has not chose to advertise its potential in this area. With the enormous amount of software available for the TRS-80 under both TRSDOS and CPM and national service availability, the Model II is a logical choice for use in any distributed processing network.

Mass Software

In my July column, I suggested a new user should stick with software packages that have wide distribution. This advice may be difficult to follow with mass distribution software packages such as Radio Shack products. These products are all wide distribution products by the nature of Tandy's distribution system. Although this does not automatically mean they are "bug free," it does mean any major bug will be found and corrected immediately. You can imagine the irritation level in Fort Worth when hundreds of dissatisfied users are calling in with program problems.

Mass distribution of software has created a unique situation. Until this occurred, a software package that had over 500 users was considered a mass distribution product. The system "debug cycle" was measured in years and user groups were developed to make suggestions for product improvement. Now a system's life cycle has been radically altered. With 1,000 systems distributed for the same computer, the "debug cycle" has been reduced to a few months. However, we still need business users groups to provide suggestions to the vendor. If there is any interest in forming such a group, please contact me c/o Oppenheim Appel Dixon & Co., One New York Plaza, NY 10004. ■

Now NRI takes you inside the new TRS-80 Model III microcomputer to train you at home as the new breed of computer specialist!



NRI teams up with Radio Shack advanced technology to teach you how to use, program and service state-of-the-art microcomputers...

It's no longer enough to be just a programmer or a technician. With microcomputers moving into the fabric of our lives (over 250,000 of the TRS-80™ alone have been sold), interdisciplinary skills are demanded. And NRI can prepare you with the first course of its kind, covering the complete world of the microcomputer.

Learn At Home in Your Spare Time

With NRI training, the programmer gains practical knowledge of hardware, enabling him to design simpler, more effective programs. And, with advanced programming skills, the technician can test and debug systems quickly and easily.

Only NRI gives you both kinds of training with the convenience of home study. No classroom pressures, no night school, no gasoline wasted. You learn at your convenience, at your own pace. Yet you're always backed by the NRI staff and your instructor, answering questions, giving you guidance, and available for special help if you need it.

You Get Your Own Computer to Learn On and Keep

NRI training is hands-on training, with practical experiments and demonstrations as the very foundation of your knowledge. You don't just program your computer, you go inside it...watch how circuits interact...interface with other systems...gain a real insight into its nature.

You also work with an advanced liquid crystal display hand-held multimeter and the NRI Discovery Lab, performing over 60 separate experiments. You learn troubleshooting procedures and gain greater understanding of the information. Both microcomputer and equipment come as part of your training for you to use and keep.

Training includes new TRS-80 Model III microcomputer, 6-function LCD Beckman multimeter, and the NRI Discovery Lab with hundreds of tests and experiments.

(TRS-80 is a trademark of the Radio Shack division of Tandy Corp.)

Send for Free Catalog... No Salesman Will Call

Get all the details on this exciting course in NRI's free, 100-page catalog. It shows all equipment, lesson outlines, and facts on other electronics courses such as Complete Communications with CB, TV/Audio and Video, Digital Electronics, and more. Send today, no salesman will ever bother you. Keep up with the latest technology as you learn on the latest model of the world's most popular computer. If coupon has been used, write to NRI Schools, 3939 Wisconsin Ave., Washington, D.C. 20016.

NRI McGraw-Hill

NRI Schools
McGraw-Hill Continuing Education Center
3939 Wisconsin Avenue
Washington, D.C. 20016

All career courses approved under GI bill.
☐ Check for details

NO SALESMAN WILL CALL.
Please check for one free catalog only.

- ☐ Computer Electronics including Microcomputers
- ☐ Color TV, Audio, and Video System Servicing
- ☐ Electronics Design Technology
- ☐ Digital Electronics
- ☐ Communications Electronics • FCC Licenses • Mobile CB • Aircraft • Marine
- ☐ Basic Electronics
- ☐ Small Engine Servicing
- ☐ Appliance Servicing
- ☐ Automotive Servicing
- ☐ Auto Air Conditioning
- ☐ Air Conditioning, Heating, Refrigeration, & Solar Technology
- ☐ Building Construction

Name (Please Print) Age

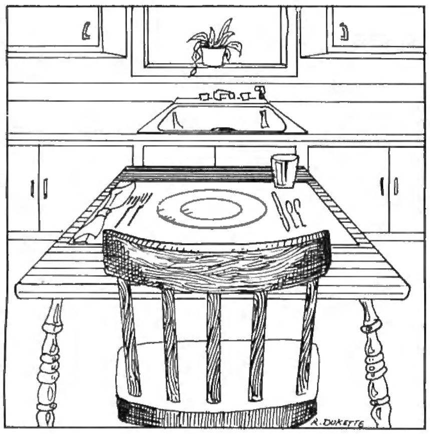
Street

City/State/Zip

Accredited by the Accrediting Commission of the National Home Study Council #179-091

News From KITCHEN TABLE SOFTWARE, INC.

by David Busch



Early in 1981, one of the top programmers at Kitchen Table Software Corp., a fictitious supplier of innovative programming for the TRS-80, put the finishing touches on a brand new disk operating system, DROSSDOS 8E, originally intended as a replacement for DROSSDOS 1.1, reviewed in *80 Microcomputing* several months ago. Unfortunately, the new DOS was compatible with no known computer system. A few ill-conceived but insurmountable bugs in the DOS made it impossible to run on the Z-80 microprocessor. The machine code used instructions that did not exist in the Z-80 chip.

Rather than scrap the estimated 47 person-years invested in developing this DOS, Kitchen Table immediately dispatched a staffer to Sri Lanka, long known as an oriental hotbed of solid-state high technology. There, arrangements were made for the development of an entirely new computer system, based on a new chip, the Z-79A, which was designed to include the new instruction set.

I'm happy to report that the innovative firm has undergone a name change to Kitchen Table, Inc., and has broadened its scope to include hardware marketing. Sometime this year, KTI will be importing the new micro, dubbed the TLS-8E by the manufacturer. The computer is of special interest to *80 Microcomputing* readers, because it is 100 percent downward compatible with the TRS-80 Model I or Model III.

The Unbiased Report

Because of my glowing reports on their earlier efforts, Kitchen Table asked me to prepare an equally unbiased report on the TLS-8E. I accepted a sample of the new micro (on *loan* only, we reviewers have scruples, too), and promised to give the machine a thorough workout sometime before the loan expires in 1991.

Less than three days later, the TLS-8E arrived. It took only five minutes to untangle the unit from the wadded paper packing (this later turned out to be the documentation), and to set up the machine. The TLS-8E consists of five major components: a keyboard/CPU, CRT unit, expansion interface, tape drive, and a long spool of ribbon cable. Odds and ends included with the kit were some loose PC card connectors, solder, a soldering iron, and a stack of 8-track tapes. Following the instructions, I set the CRT on top of the expansion box, and quickly snapped the edge card connectors onto appropriate lengths of ribbon cable. These were then plugged into likely-looking orifices on the keyboard, expansion interface, CRT and tape drive. I powered the contraption up, and, lo, the CRT sprang to life with the message "Memoly Size?"

The keyboard is simplicity itself. To keep costs down, KTI elected to go with a non-standard keyboard. Instead of the familiar QWERTYUIOP array, the keyboard is laid out in a 16-key matrix, four keys on a side. Each of the 12 keys in the top three rows may be pressed in four directions,—forward, back, left and right. This configuration yields a possible 48 characters to be entered using only 12 keys. The bottom four keys are SHF (shift), CTL (control), ESC (escape) and TBA (to be announced). When one of these keys is depressed along with one of the upper 12, the result is uppercase control characters and the Chinese alphabet. My source at KTI says the TBA key will be used to produce a selection of Japanese, Greek, Russian and popular Walt Disney characters. The company is still working on the character-generator ROM, which is said to weigh over five pounds.

Character arrangement on the keys follows a logical alphabetic order. KTI reports that the TLS-8E will not be sold to touch typists under any circumstances.

I broke the seal on my unit to look at the interior. The breadboard inside was impressively arrayed with transistors, capacitors, a few ICs and other complicated-looking gadgetry. A KTI spokesperson explained that the TLS-8E makes extensive use of hard-wiring in place of the less expensive (cheaper) printed circuitry. In a similar vein, more than a dozen ICs have been replaced by the equivalent transistors and other solid-state components, totalling in the thousands. Those integrated circuits that were used, such as the Z-79A microprocessor, have been solidly soldered onto their connecting wires to keep the computer working reliably, even through the estimated 1200 degree temperatures that develop inside the keyboard when the computer is left on for more than 30 minutes. Stainless steel was used for the keyboard case so the entire unit can function as a heat sink.

On to the CRT: It is not necessary to connect the CPU to the monitor. Instead, an unshielded RF modulator built into the computer automatically transmits directly to the CRT and any other television receiver within 100 yards. The monitor supplied with the TLS-8E was a generous 19-inch unit, which had Des Plaines Holiday Inn engraved on its top. Resolution of the image compared favorably to that of the TRS-80, which was unfortunate. We had hoped for better. However, I welcomed the addition of a Horizontal Hold button to my CRT controls.

The TLS-8E monitor provides the popular green screen favored by many computerists. However, and I believe this is unique to the industry, this unit allows the user to *adjust* the tint of green to suit personal preference. I added a touch of magenta to warm the image a bit.

Though I have learned that KTI plans to allow purchasers to start with a basic TLS-8E, and upgrade their systems a component at a time, the expansion interface is practically a necessity, even for first-time purchasers. First, without the expansion box, the CRT wobbles uncontrollably, for it is supplied with only three feet on the bottom. Second, and perhaps more important, *all* of memory is contained within the expansion interface. Both the ROM and RAM chips reside permanently in the optional interface. This possibly explains why the basic TLS-8E (keyboard, CRT and tape drive) is priced at $99, while the expansion interface carries a $599 price tag.

The good news is that the interface is fully supplied with a 16K ROM, 54K of RAM (yes, the Z-79A can address 70 thousand memory locations), a disk controller, RS-232 board, and other goodies. These can be put into service by the user with minimal wiring and soldering.

The Tape Drive

Special mention has to be made of the tape drive supplied with the TLS-8E. KTI has chosen to eschew the popular Kansas City and less popular Radio Shack standards for its own unique cassette I/O system. I don't understand the electronics, but I do appreciate the hardware: The TLS-8E makes effective use of an ordinary eight-track tape recorder. This innovative tack makes several new features possible. Now one can record eight *different* programs on the same length of tape. By setting the recorder in monaural mode, and switching among the tracks and channels, users can store more than 250 megabytes of program material on one 80-minute eight-track tape!

Alternatively, programs can be stored on four tracks, and the other four used for recording stereo music. The TLS-8E tape drive is the only microcomputer tape I/O device of which I'm aware that is supplied with interchangeable face plates to fit both GM X-body cars and most Toyota dashboards. The biggest drawback is that the tape cannot be rewound. Once a program has been CSAVED, or CLOADed, the user must fast-forward through the entire tape to get back to the same point.

This brings us to one peripheral that KTI did not send me to test—disk drives. I thought that odd, because it was the offshoot of DOS, DROSSDOS 8E, that spurred the creation of the computer in the first place. However, I have inspected the Kitchen Table literature carefully, and was able to glean a few details about the disk drives. I filled in the gaps with a long telephone conversation with KTI technical personnel.

The company's product development people combined features from many different drives in designing the KTI 100. The units are single-sided, 35-track, 40 ms track-to-track drives fully rated for single- or half-density operation. A rheostat that emerges from the top of the case allows the user to alter the speed of the drives plus or minus ten percent.

Design of the drives is unusual to say the least. Great emphasis has been placed on simplifying those components prone to breakdown, to provide greater reliability. For example, the write-protect tab sensor and door-open sensor have been eliminated entirely. As a matter of fact, the KTI 100 has no door at all. The user simply slips in a disk and enters the desired command at the keyboard. The drive cone clamps the disk when the motor starts, although it is sometimes necessary to jiggle the disk a bit to ensure proper centering and mounting. There is no LED on the front of the unit to indicate that the drive is operating. KTI spokespeople point out that the noise generated by the motor is sufficient indication. The stepper arm makes a grinding sound as it moves across the disk surface, so it is fairly easy to tell which drive is actually being accessed.

"The TLS-8E makes effective use of an ordinary eight-track tape recorder."

The KTI drives are supplied with laminated fiberboard cases. The units are compact because no power supplies are incorporated in the drives themselves. KTI deemed the 1250-watt supply built into the TLS-8E keyboard sufficient for all peripherals. Though the voltage supplied (+5 v. and −5 v.) to the drives can vary as much as 37 percent, the rheostats on top of the drives can compensates for fluctuations as needed. KTI also provides circuit diagrams for a simple rewiring, allowing the user to dim the room lights when the disk drives are not in use.

DROSSDOS 8E is the standard operating system supplied free with the disk drives. Mandatory Zaps are available for only $4.95 per month.

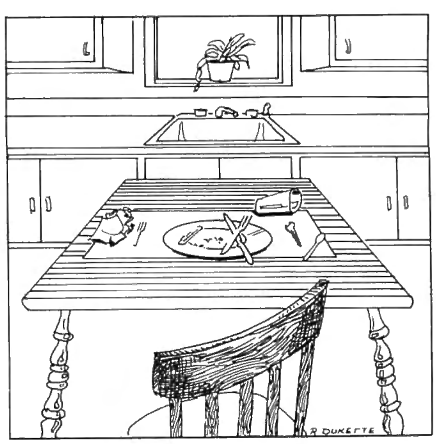
What about the compatibility with the TRS-80? Kitchen Table insists that their new computers should be 100 percent interchangeable with Tandy's micro for nearly any application. The key is the Z-79A processor. The Z-80's instruction set is a subset of the Z-79A's, which means that all machine language programs are directly transferrable. The Z-79A does have more than 114 additional instructions, such as SHOVE, DROP, FLIP, FORGET, and CREATE, which should be invaluable to creative programmers who don't mind limiting their market to the 1000 TLS-8E owners whom KTI estimates will blanket the nation by mid-1985.

The TLS-8E's Level XXII Basic, and Disk Basic supplied with DROSSDOS 8E, are similarly compatible with the Radio Shack equivalents for the Model II and Model III. KTI says that POKEs can be written into TLS-8E programs, and transferred directly to Radio Shack Model I and Model III computers. The company does caution that the result might not necessarily be the same, however.

For example, POKE 15360,65 in the TRS-80 will cause an uppercase A to be produced on the CRT screen. This same command in Level XXII instructs the computer to zero 4K bytes of memory, beginning at the starting address of the program. This could conceivably cause problems to the unsuspecting programmer. KTI is planning to provide a Superdupermap listing that will cross-reference the TLS-8E ROM with Tandy's.

Some Level II features have multiple applications in Level XXII. DIM may be used to dimension an array. Without an argument, DIM will lower the brightness level of the CRT by 50 percent. CLS will clear the screen if used at the beginning of a subroutine. Otherwise, it causes the computer to Cancel Last String.

There are too many enhancements to Level II in the KTI TLS-8E to discuss here. For a computer that is not officially on the market yet, and which may never be imported after this review is published, the TLS-8E has generated a great deal of interest. Keep reading *80 Microcomputing* for further developments from Kitchen Table, Inc. ■



THE EXCLUSIVE ORACLE

by Dennis Kitsz

Q. I have an alternate suggestion for Mr. John W. Schrage of Chester, NJ, concerning his problems with the real-time clock and LNW board (EXclusive ORacle, May 1981). Perhaps he has not seen the note at the bottom of page 5 of the LNW users manual:

"*If Floppy section is not installed, connect U23 pin 9 to ground. Note: U14 (FD1771B) must not be installed."

Also advise him to check for etch shorts. Mine had one under an IC socket! Every time I tried to load a tape from cassette, the UART would also become enabled, dumping all sorts of garbage onto the data line.

Ron Kiyomura
Huntington Beach, California

A. It seems many of us never saw LNW's footnote on enabling the clock. Thanks to Ron and other readers who called my attention to it. That small print can save Mr. Schrage the $24.95 cost of the floppy disk controller.

Q. I tried your "unlocking Reset" note (80 Applications, September, 1980), but it had no effect on the need for Break with Reset, and continues to show the memory size question when pressed. I am enclosing a sketch of my interpretation of your instructions. I have tried all arrangements of the 10,000 ohm resistor, but nothing worked except that by keeping the switch closed I still had my original system intact. Did I goof somewhere?

Dan Fielder
Atlanta, Georgia

A. My instructions might have been clearer, as you and several other readers pointed out. The trace cut you need to make is found between Z32 pin 4 (Z39 on newer interfaces), a type 74LS155 circuit, and Z23 pins 1 and 15 (Z50 and Z51 on newer interfaces). Your sketch shows you have cut the trace *next to* pin 4, which is not correct.

A schematic of the trace cut is shown in Fig. 1.

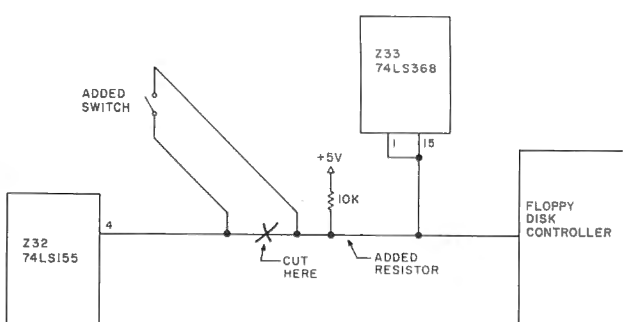


Fig. 1

Also, the Reset modification *can* be made to work for users of the LNW expansion board. The LNW design incorporates termination resistors clamping the data lines low (220 ohms to ground, 470 ohms to +5 volts). Even when the disk controller response is supposed to be in the invisible "tri-state" mode–your CPU should see all ones on the data bus (decimal 255)–the LNW resistors force it to see zeroes instead. I have contacted LNW about this, but they feel the problem is insignificant.

If you are using the LNW board and wish to unlock Reset, you must replace the termination resistors with other values. Replace the 220-ohm resistors with 680-ohm resistors, and leave the 470-ohm resistors as they are. The lines will still be immune to noise (the reason for the resistors in the first place), and Reset works once again. This will also prevent some unusual memory phantoming (the appearance of high memory where there actually is none) on LNW systems with only 16K expansion memory.

Q. I have problems with speeding up my TRS-80. The mod works fine in the high-speed mode as long as the program doesn't use any memory from the expansion interface. I purchased a 300 nanosecond memory from Godbout Electronics; my expansion interface was manufactured in January, 1980, and it does not have the buffered cable. I can't understand what is causing this problem.

Harry Ventin
Wilmington, Delaware

A. Speeding up the TRS-80 itself is not a hard task, but getting an entire system to work at high speed can be a problem. First, let me refer you to a note that supplements the original high-speed modification published in February, 1980, since changes are required for the newer expansion interfaces–see 80 Applications, "High Speed Update", May, 1981, page 54. This change speeds the memory selection in the expansion box, which Radio Shack set to a fixed rate.

A second change is made in the keyboard unit itself, and has been published in various forms in user group newsletters. This second change is necessary in situations where any part of the memory-select circuitry is marginal, and will improve reliability anyway. The simplest version: Locate Z69 in the keyboard unit, and find pin 5. You will see that a circuit trace goes off to the left, around, and back underneath Z69. It leads to pin 12. Cut that trace. Now attach a small jumper wire between pin 12 and pin 10. Assuming you have made both the changes to the expansion box and to the keyboard unit, your high-speed modification should work correctly.

By the way, if you have a great deal of patience and good eyes, you can install a super speed-up kit available from Archbold Electronics in Rancho Cordova, California. For about $45, you can double the speed of your machine. This board is also "intelligent", monitoring the activity of the computer so it automatically slows down during disk access. Tripling the

Now for the TRS-80™ Models I & III

LDOS

THE TRS-80™ OPERATING SYSTEM FEATURING:

- * Double Sided & Double Density Support
- * Hard Drive Support Up To 10 MEGS As A Single Drive
- * Inter mix 5", 8" and Hard Drive Up To 8 Total Drives
- A 250 Page Manual
- Complete Technical Information
- A TOLL-FREE 800 Number for Customer Service
- A Full Time Staff to Handle Customer Service
- A Bulletin Board on MicroNet
- An LDOS NEWSLETTER
- A Liberal Update Policy
- An ENHANCED BASIC
- A Complete Job Control Language
- Device Independent
- Media Compatible Model I to Model III
- A One Year Warranty

* SPECIFIC HARDWARE IS REQUIRED TO UTILIZE THESE FEATURES.

The Ultimate In Operating Systems Model I & III

Only $149.00

For Further Information Contact The Distributor Or Dealer Nearest You:

DEALERS

Foreign

Cesco Electronics Ltd.
Montreal, Canada

Calisto Computers Ltd.
Birmingham, England

Molimerx Ltd.
Sussex, England

Graphie
Montreuil, France

Domestic

Hobby World
Northridge, CA

Byte Shop of San Diego
San Diego, CA

Howard Industries
Anaheim, CA

Patio Computer Sales
North Hollywood, CA

J & R Inventory
Santa Ynez, CA

Jim's Computer Service
Laguna Nigel, CA

Micro Computer Tech, Inc.
Santa Ana, CA

Berkeley Microcomputers
Berkeley, CA

Dimensional Software
San Diego, CA

Byte Shop of Placentia
Placentia, CA

Computerland of 1000 Oaks
Thousand Oaks, CA

Computer Forum
Santa Fe Springs, CA

Teem Quest
Palm Desert, CA

H.W. Computers
Redondo Beach, CA

Lobo Drives Int'l
Goleta, CA

The Program Store
Washington, DC

Computer Junction
Ft. Lauderdale, FL

Adventure International
Longwood, FL

Personal Computing Center
Margate, FL

H.I.S. Computermation
Melbourne, FL

Thompson Sporting Goods
Rexburg, ID

Idaho Micro Computer
Buhl, ID

Information Concepts
Richton Park, IL

Advanced Operating Systems
Michigan City, IN

Radio Shack Computer Ctr.
Louisville, KY

The Computer Shoppe
Metairie, LA

Small Business Systems Group
Westford, MA

Computer Center
Garden City, MI

The Alternate Source
Lansing, MI

Computerland of Kentwood
Kentwood, MI

The 8 Bill Corner
Muskegan, MI

Soft Sector Marketing, Inc.
Garden City, MI

Tri County Electronics
Fenton, MI

Comp-U-Trs Software
St. Louis, MO

Forum-80 Headquarters
Kansas City, MO

The Software Shop
Belton, MO

J & J Electric/Radio Shack
Hackettstown, NJ

The Computer Forum
Redbank, NJ

M-T Electronics
Sparta, NJ

Computer Madness
Englishtown, NJ

Autel Electronics
Albuquerque, NM

Data Base, Inc.
Albuquerque, NM

Hurley Electronics
Las Vegas, NV

A World of Computers
Port Chester, NY

Programs Unlimited
Jerico, NY

B. I. Enterprises
Centereach, NY

Computerland of Nassau
Carle Place, NY

Computerland of White Plains
White Plains, NY

Computronics
Spring Valley, NY

Crowleys Radio Shack
Whitehouse Station, NY

John D. Owens Associates, Inc.
Staten Island, NY

Computerland of New York City
New York, NY

Digibyte Computers Center
New York, NY

Future Visions Computer Store
Melville, NY

Micro Mini Computer World Inc.
Columbus, OH

The Computer Store
Tulsa, OK

Vern Street Products
Sapula, OK

V.R. Data
Folcroft, PA

Microtronix Inc.
Philadelphia, PA

Stevens Electronics/ Radio Shack
Phoenixville, PA

Erie Computer
Erie, PA

Quality Software Distributors
Dallax, TX

Computers To Go
Austin, TX

Aerocomp
Dallas, TX

Custom Computer Center
Houston, TX

AT-80
Dallas, TX

Misosys
Alexandria, VA

80-US Journal
Tacoma, WA

Petted Micro Systems
Milwaukee, WI

Galactic Software Ltd.
Mequon, WI

DEALER INQUIRES WELCOME
LDOS is a product of LSI Inc.
TRS-80 a trademark of Tandy Corp.

(West)
LOBO DRIVES INT'L
354 S. Fairview Ave.
Goleta, CA 93117
(805) 683-1576

(Central)
GALACTIC SOFTWARE LTD.
11520 N. Port Washington Rd.
Mequon, WI 53092
(414) 241-8030

(The Common Market)
MOLIMERX LTD.
1 Buckhurst Rd., Bexhill
Sussex, England
(0424)-220391

(East)
MISOSYS
5904 Edgehill Dr.
Alexandria, VA 22303
(703) 960-2298

LOGICAL SYSTEMS INC. ✓35

✓See List of Advertisers on page 354

speed of the computer is also possible with this board, by replacing the Z-80 microprocessor with a hotter Z-80B, and adding a $20 delay line also sold by Archbold.

Q. I bought an Expansion Interface on a trip to the United States two years ago. It is an old version with a black box cable, but no DIN plug connector. I have installed memory chips from Jade and Godbout. However, with the E/I switched on, I only get 15572 and not the full 48K in response to the Print MEM command.

The Jade literature states: "You should be aware that there are some versions of Level II that do not give an indication of memory in the E/I. If you ask your machine to print the memory size and it gives you a number like 15572, this does not necessarily mean that the memory in the E/I is bad—simply that the software does not recognize its existence on a Print MEM command. Try reading and writing to the higher memory..."

I am able to write to and read from memory. However, when I try to set the memory size above the 16K limit, the computer hangs up, and when I use a memory test program on the upper memory it says that it is practically all defective. Is there damage to my system? Does my AC current of 50 Hz create timing problems?

Ken Lennan
Brussels, Belgium

A. First of all, the Jade literature is straight out of science fiction fantasy. All the Level II memory tests are the same—i.e., each potential memory location is read in order by the CPU. The complement of the value which has been read from memory is written back into the memory location under test. The new value is then read back. If the new value does not match the value written, the software assumes it has reached the end of usable memory.

Apparently this test is failing at the beginning of Expansion Interface memory. In my answer to Mr. Ventin (above), I mention the change necessary to the keyboard unit to produce faster memory access. This is doubly important with the buffered cable, which delays access about 20 nanoseconds longer. It's unlikely there is damage to the buffered cable, since other problems would show up. The 50 Hz power is irrelevant, except that your external power supplies will run warmer (don't enclose them in the slots provided in the expansion box!).

You should be sure to install the "twisted pair" modification—known by various names including the DIN cable and the XRX-I mod—because the buffered cable alone actually made things worse on some computers. These cables are still available at no charge from Radio Shack (or Tandy stores in Belgium) by special order. If you cannot obtain one, you can make your own using a six-pin DIN plug/jack pair.

In either case, Radio Shack provides no instructions for its installation, so here they are:

The twisted pair mod consists of three sets of wires (red–black, white–black, and green–black). Each pair of wires is twisted together in what is called "transmission line" configuration, which helps shield external noise from these sensitive select circuits. The female end of the cable connects to the keyboard unit, the male end to the expansion interface.

- Open the keyboard unit and locate Z72. Identify pins 3, 5, 8 and 9. Turn the board over; you will be attaching the cable to the bottom.
- Trim the bare ends of the female cable so that only 1/16 inch (1.5 mm) of tinned wire remains exposed.
- On the bottom of Z72, solder the red wire to pin 5 (Row Address Strobe, or RAS), solder the white wire to pin 9 (Column Address Strobe, or CAS), and solder the green wire to pin 3 (Memory Select Multiplexer, or MUX).
- Solder each of the black wires to the three available points leading from Z72, pin 8 (Ground).
- Find pins 1, 3, and 16 on the edge-card connector. These are the connectors normally used for RAS, CAS, and MUX. Cut all three foil traces leading from the edge connector. They will no longer be used, and are cut to lower induced noise.
- Route the cable out the back of the computer,and replace the cover.
- Now open the Expansion Interface and locate Z18 pins 1 and 8, Z22 pins 12 and 8, and Z24 pins 2, 4 and 7.
- Again trim the bare ends of the DIN cable (the male end this time) to 1/16 inch (1.5 mm) of exposed wire.
- Solder the red wire to Z22 pin 12, and its black partner to pin 8. Solder the white wire to Z18 pin 1 and its black partner to pin 8. Solder the green wire to Z24 pin 2 or 4 (they are connected by a circuit trace), and its black partner to pin 7.
- Now locate pins 1, 3 and 16 on the Expansion Interface edge card connector. The pin numbers are silkscreened on the circuit board, and are not a mirror image of those on the keyboard unit. Cut these traces at the edge connector.
- Route the cable out the front edge of the case, and close the unit. The modification is complete, and, together with the keyboard unit modification to Z69 explained in the letter above, should have your TRS-80 system running crash-free.

Q. I have a TRS-80 Model I Level II 48K cassette and Stringy-Floppy system, and a brand new Epson MX-80. I am using Scripsit. While it works well by itself, I cannot use any of the special modes available with the MX-80. It is of course possible to have the printer primed for a special type of printing before Scripsit is loaded, but then one is locked into that particular mode. Can you help?

Ranes C. Chakrovorty
Salem, Virginia

A. I can't help, but Jerry Goodwin can. Jerry has developed a few programs to patch Scripsit. The first program is called SCRIPTR, which modifies Scripsit to enable use of MX-80 control codes, as well as: entry and reentry of DOS; programmable display; printing of graphics; insertion during printout; complete printing control; a macro facility; and other features. SCRIPTR can be obtained from Jerry directly, and will be available from Instant Software for $40 within a month after you read this.

In response to my request for you and other readers who only need the MX-80 control-code feature (for the moment), Jerry has also prepared another program called SCRIPT80 to do just that. Send Jerry $7 (or a disk and $5) and he'll forward it to you: 1746 N.W. 55th Ave. #204, Lauderhill, Florida 33313.

Finally, Jerry has offered to provide a customized patching program for SCRIPTR to allow its use with just about any printer on the market. Once you've received a copy of SCRIPTR from Instant Software, send Jerry the complete specifications for your printer, together with $25, and he'll return you a patch program and a 25-page book of documentation within six weeks.

How's that for service, Ranes?

Updates

Reader Lenny Greenberg asked in May's column about interfacing 128 devices to the TRS-80, and my response included the line "So here's part of a circuit you might use:" Unfortunately, the schematic diagram was left out. So, Lenny, here *is* part of a circuit you might use (see Fig. 2).

In the same issue, a gremlin crept into my explanation to Dave Hildebrandt of the TRS-80 scrolling routine. The routine works like this:

```
LD    HL,3C40H
LD    HL,3C40H
LD    DE,3C00H      (not 3C41H)
LD    BC,3BFG       (not 37FH) LDIR
```

The original explanation would have had a screen full of identical characters! Thanks to Dennis Thurlow for pointing out the error.

Brian Donlon has written with a method of providing debounce to EDTASM. Here it is:

"The keyboard driver for EDTASM begins at 445FH. I replaced the routine here with the routine I found in KBFIX starting at 7FDAH through 7FFF. There was only one jump that had to be changed. After loading this routine over the one in EDTASM, I changed the instructions now residing at 4402H from F2 E2 7F to F2 F7 43.

"The code including the above instruction changes is shown in Fig. 3."

"You can zero 4415H to 445FH. There is enough room here, starting at 43EFH, to use KBEEPFIX (80 Applications, February, 1980) and you won't have to change any instructions. Just use the routine starting at 7F98H through 7FFFH."■

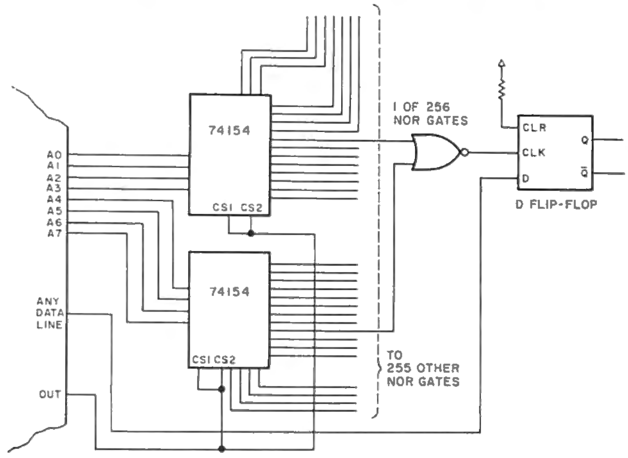


Fig. 2

Address your questions about the TRS-80 to Dennis Kitsz, Roxbury, Vermont 05669.

43E0	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	21
43F0	36	40	01	01	38	16	00	0A	5F	AE	73	A3	20	08	14	2C
4400	CB	01	F2	F7	43	C9	5F	C5	01	DC	05	CD	60	00	C1	0A
4410	A3	C8	C3	FB	03	–	–	–	–	–	–	–	–	–	–	–

Fig. 3

THE ASSEMBLY LINE

by William Barden, Jr.

"Ada. . . was thoroughly disgusted with the support software . . . and reputedly attempted upgrades."

Programmers have always been interested in how other people's software worked. Ada, the world's first programmer, was thoroughly disgusted with the support software on Babbage's Analytical Engine and reputedly attempted upgrades.

Given the commented source code for a particular Assembly program, it is sometimes virtually impossible to figure out what is going on. Without comments, the problem is quite a bit more difficult. Without source code, the problem is an order of magnitude more difficult.

Is it possible to disassemble machine code and to find out how a program works? It certainly is. In this month's column, we'll discuss some of the tricks of the disassembler trade, in a special report entitled "The Disassembly Line".

Why Disassembly?

Disassembling can be an educational experience. You can see how Microsoft implements their code, and get a first-hand report of the programming expertise at Radio Shack.

Secondly, disassembly may be necessary to correct bugs in the program. There are some assemblers that will not assemble LD A, (IX-3) properly; there are some operating systems that blow up on full diskettes; there are some Assembly mail-list programs that will not handle addresses in the 92692 Zip code.

Thirdly, creeping elegance rears its beautiful head. Why not improve programs? Why not add underlining, escape sequences, and proportional spacing in Scripsit? I am firmly convinced that software for the Radio Shack computers has become better because of the pressure of improved versions; without them, we'd still be at TRSDOS Version 1.0. The overall result is better Radio Shack software and a more competitive product.

MEMORY LOCATION OF INSTRUCTION	MACHINE CODE	DISASSEMBLER OUTPUT	
5522	CDB954	CALL	54B9
5525	ED5B537C	LD	DE,(7C53)
5529	DD4E00	LD	C,(IX+00)
552C	C9	RET	
552D	ED5B537C	LD	DE,(7C53)
5531	21627F	LD	HL,7F62
5534	B7	OR	A
5535	ED52	SBC	HL,DE
5537	2005	JR	NZ,553E
5539	DD4E00	LD	C,(IX+00)
553C	37	SCF	
553D	C9	RET	
553E	CD0D55	CALL	550D
5541	DD7E00	LD	A,(IX+00)
5544	91	SUB	C
5545	67	LD	H,A
5546	CDD953	CALL	53D9
5549	7C	LD	A,H
554A	91	SUB	C
554B	380C	JR	C,5559
554D	3A647C	LD	A,(7C64)
5550	91	SUB	C
5551	2600	LD	H,00
5553	69	LD	L,C
5554	19	ADD	HL,DE
5555	EB	EX	DE,HL
5556	0E00	LD	C,00
5558	C9	RET	
5559	6C	LD	L,H
555A	2600	LD	H,00
555C	DD7E00	LD	A,(IX+00)
555F	95	SUB	L
5560	4F	LD	C,A
5561	19	ADD	HL,DE
5562	EB	EX	DE,HL
5563	C9	RET	
5564	AF	XOR	A

Fig. 1. Typical Disassembly

First, Load the Disassembler

A disassembler is the inverse of an assembler. It takes machine code in memory and converts it into the equivalent Assembly program mnemonics. Take a look at Fig. 1. This is a portion of a disassembly of Scripsit. Because of copyright laws, the comments from the disassembly have been left out.

You can see that there are no symbolic labels as there are in a typical Assembly program listing. All jumps are to absolute addresses. Also, it may be possible for the disassembler to get lost when it encounters data, buffer or unused areas.

Fortunately, the disassembler will almost always get back into sync, although it is statistically possible for it to never recover from the point at which it first encounters garbage.

Some disassemblers output the disassembly listing to disk as a file. The file can then be edited and reassembled. Some disassemblers even provide labels in the disassembly.

One of the first disassemblers was Small Systems Software's RSM-1 package. Apparat followed with their DISASSEM. Two other excellent disassemblers are the Misosys DSMBLR, which provides labels, Equates, and Origins and outputs to cassette or disk files, and The Alternate Source's Tasmon, which provides symbols and disk files. (The Tasmon package is a powerful monitor, one of the best I've seen.) All disassemblers provide output to the system printer.

✓See List of Advertisers on page 354

```
5200  START
5202  TITLE MSG
F7F7  VERSION MSG. INSERT MSG
5C69  SMALL MSG AREA
69C3  LARGE MSG AREA + TABLES
6FB9  SMALL MSG AREA
7452  TWO MSGS
77BF  TABLES + LARGE MSG AREA
7AA4  END
```

Fig. 2. Rough Map of Scripsit

Next, Get a Listing

The next step in disassembly is to get a disassembly listing. You may have to do some fancy relocation footwork here, depending upon the disassembler. Some disassemblers provide a disassembly from a disk file, making it unnecessary to relocate the program to be disassembled. Tasmon relocates itself, a neat trick for a monitor program.

The starting address of the file to be disassembled may be obtained automatically by NEWDOS' LMOFFSET or other utility programs. At the worst, you can find the area loaded by clearing memory, loading the file by a DOS Load or Basic System command, and scanning memory to see which areas have been loaded.

Once the file has been loaded, disassemble it from beginning to end. If the program is any size at all, you'll wind up with a number of pages with approximately 50 instructions per page or 50 instructions per 100 bytes of program.

For illustrative purposes, we'll be working with a disassembly of Disk Scripsit, a moderately hard program to disassemble.

Finding the Messages

Set the listing aside for a while. Like the Shadow, it has the power to cloud men's minds at this point. We're going to chop it down to size by the next step.

Use your monitor, dump program or Debug to print out the ASCII equivalent of the memory area you just disassembled. Here again, you may not be able to do this easily. At the worst, write a short Basic program that uses PEEKs, or simply scan the area visually by disk Debug.

You must now correlate the ASCII with the disassembly listing. The Scripsit program we're using as an example uses many messages; other programs may not have more than few. Write the message opposite the starting point on the disassembly listing. In the case of Scripsit, this will probably take several hours.

What you now have is a disassembled listing with all messages denoted. In addition to messages, you may also have tables of ASCII values, which probably appear to be senseless. A rough map of Scripsit at this point is shown in Fig. 2. The first byte of every message in Scripsit, by the way, is the length, in bytes, of the message following. At this point the 10,400 bytes of Scripsit have been reduced to about 9,100 undefined bytes—not bad for a start. The step-by-step process now turns into an art, not a science. We'll offer some hints at how to continue, in a rough priority that has worked for us—modify it as required for your own uses.

Look for the Message Out Routine

Having the message locations, we can now look for the message-out subroutine. With such a large number of messages, we should find many references to such a routine. Sure enough, there are many Call 6BC8s, each one preceded by an LD HL, XXXX instruction, where "XXXX" is the location of the message.

Mark each call (two dozen or so) with the message to be output, as shown in Fig. 3. Since Scripsit messages are often associated with the function to be performed, you're essentially mapping the areas of the program by function, as shown in the figure. The Control R command, or Repeat, in Scripsit displays "Repeat How Many Times", waits for an input of the repeat count, and then displays "Enter Repeat Command". We can see from the figure that this is the sequence at location 6F42H. We can strongly suspect, therefore, that this is the portion of the program associated with the Repeat command.

Decoding the Tables

Scripsit uses many special Break sequences, where a Break key press is followed by the message "Special Command?". An input of a one-character command is then done—R for Replace, T for Tab, ? for system query, and so forth. Scanning our ASCII dump (or using Debug in the ASCII mode) we see a likely looking string at location 6AE4H—WPI?TSLFRDHSCVE. Each character corresponds to one Break special command. This is more than coincidence. What else can we deduce from the table?

With a table of this type, it's a good bet that there is a corresponding jump table of addresses somewhere else in memory. The index to the command character is used to access the jump address. Sure enough, the 14 addresses following the string at 6AE4H all look like valid addresses in the Scripsit program area, when transposed in Z-80 least-significant, most-significant fashion. There's a good possibility that there's a one-for-one relationship between the two tables.

```
6F3D BA         CP     D
6F3E C0         RET    NZ
6F3F 7D         LD     A,L
6F40 BB         CP     E
6F41 C9         RET
6F42 21F26F     LD     HL,6FF2    'REPEAT HOW MANY TIMES' MESSAGE
6F45 CDC86B     CALL   6BC8       CALLS TO 'MESSAGE OUT'
6F48 CDF86B     CALL   6BF8
6F4B FE01       CP     01
6F4D 28F3       JR     Z,6F42
6F4F F5         PUSH   AF
6F50 FD3611FF   LD     (IY+11),FF
6F54 21DD6F     LD     HL,6FDD    'ENTER REPEAT COMMAND' MESSAGE
6F57 CDC86B     CALL   6BC8       CALLS TO 'MESSAGE OUT'
6F5A CDD652     CALL   52D6
6F5D F1         POP    AF
6F5E 32317C     LD     (7C31),A
6F61 FD3502     DEC    (IY+02)
6F64 280B       JR     Z,6F71
6F66 CD0460     CALL   6004
6F69 3A3E7C     LD     A,(7C3E)
6F6C CDE852     CALL   52E8
6F6F 18F0       JR     6F61
6F71 FD361100   LD     (IY+11),00
6F75 C9         RET
```

Fig. 3. Finding Message Out Calls

```
15 References to these Locations

F  6F76  AF      XOR   A
F  6F77  CD7CGF  CALL  6F7CH
   6F7A  1812    JR    6F8EH
5  657C  07      RLCA
5 References to this Location
```

Fig. 4. DISASSEM Reference Notation

The Newest NEWDOS/80 Version 2.0 For Model I And Model III

THE HOTTEST DISK OPERATING SYSTEM FOR THE TRS-80® COMPUTER IS NOW READY FOR THE MODEL III AND VERSION 2.0 IS READY FOR THE MODEL I. MANY ENHANCEMENTS AND ADDED FEATURES SUCH AS NEW COMMANDS MAKE YOUR COMPUTER MORE POWERFUL!

$149.00

- **DOUBLE DENSITY ON MODEL I**
  Use of the LNW DOUBLER or the PERCOM DOUBLER to expand storage 80% under NEWDOS/80 Version 2.0, mixing single and double density specifications without any patches.
- **SINGLE DENSITY ON MODEL III**
  Will allow the MODEL III to read disks from MODEL I and to write disks the MODEL I can read, making it easy to move programs between the two machines.
- **EXPANDED DIRECTORIES**
  Directories can be expanded three times the normal number of available entries, even on DOS disks. This is extremely useful when using double density.
- **DYNAMICALLY MERGE IN BASIC**
  To allow sections of BASIC programs to be deleted and replaced with lines from a disk file during program execution. Also allows merging of non-ASC II format files.
- **SELECTIVE VARIABLE CLEARING**
  Allows the programmer to keep some variables and release the space used by the rest; also, specific variables may be erased releasing the space they use.

(CALL OR WRITE FOR MORE INFORMATION ON OUR COMPLETE LINE OF PRODUCTS AND HARDWARE REPAIR SERVICES FOR YOUR TRS-80 COMPUTER)

Apparat, Inc. ✓264
4401 South Tamarac Parkway
Denver, Colorado 80237

(303) 741-1778

"On-going Support for Microcomputers"

- **PAGE SCROLLING IN BASIC**
  Scrolling has been modified to allow the user to display programs page by page, in addition to the regular line scrolling.
- **REPEAT FUNCTIONS**
  Keys in MODEL I repeat when held down. Entering "R" as a DOS command causes the previous DOS command to be repeated.
- **ROUTING FOR DEVICE HANDLING**
  To send input and output from one device (display, printer, keyboard, etc.) to others or to a routine in main memory.
- **DISASSEMBLER OUTPUT TO DISK**
  The Disassembler will now write a source code file to disk, which the editor assembler can read and edit.
- **CHAINING ENHANCEMENTS**
  Features to allow chain files to be written from SCRIPSIT; also, chaining may be switched on and off without changing chain file positioning, and may be executed via CMD "xxx" and DOS-CALL.
- **SUPERZAP**
  has the ability to scan diskettes or disk files to find the occurences of specific values. Also will generate disk file passwords and hashcode.
- **FAST SORT ROUTINE**
  basic function CMD "O" provides direct or indirect in-memory sort of multiple arrays.
- **MERGING OF NON-ASCII BASIC PROGRAMS**
- **BASIC SINGLE STEPPING**

✓See List of Advertisers on page 354

THE ASSEMBLY LINE

Many large programs with dozens of functions will make considerable use of tables. Look for them to provide clues to the location of program functions.

To get ahead of ourselves, it turns out that the addresses in the second table represent the commands in reverse order. The sequence, in the jump table at 6AF3H is for E, V, C, H, etc. (We discovered this after decoding the scan-and-branch routine at location 58F0H.)

We can now search for tables using a similar format. A table at 6AD1H of WLMCNI is followed by six addresses, and these are obviously two tables of the same structure. Knowledge of Scripsit provides the clue. These tables are associated with the responses to Break ? commands, such as Report Length, Memory, etc.

A third table, at 77BFH, holds the string RTBJCVWLPFH, followed by 11 addresses. After some thought, we conclude that this is associated with format line processing ->RM = X TM = X, etc.

Having the addresses for each of the branch points, we can now label each section of code with the function involved, further mapping the disassembly functionally.

5200	Title, Initialization
5300	Get Character
5400	Small Subroutines
5500	
5600	
5700	Messages
5800	Block Mode
5900	Scan Table SR
5A00	Unknown
5B00	
5C00	Messages
5D00	Disk I/O Subroutines
5E00	
5F00	Printer
6000	Read Keyboard
6100	
6200	Unknown
6300	Cassette Subroutines
6400	
6500	Hyphenation
6600	Special Processing, UART
6700	Special Processing
6800	
6900	? Processing
6A00	Message Area
6B00	Messages, Tables
6C00	ASCII/Binary, Other Subroutines
6D00	Deletion, Window
6E00	
6F00	Special, Repeat
7000	Messages
7100	Unknown
7200	
7300	
7400	Messages
7500	Format Line Processing
7600	
7700	
7800	Messages
7900	
7A00	Unknown
7AA4	End

Fig. 5. Rough Scripsit Structure

Separate into Subroutines

We're now past the easy part. From here on in, it's a question of seeing what the code does. At this point we can look for obvious subroutines and functional blocks. We already have a good idea where many of the large functional blocks are from the jump table addresses. However, there are probably a myriad of smaller subroutines that are below the functional blocks in structure.

The Apparat DISASSEM provides some useful hints to the location of subroutines. A reference table at the end of the disassembly lists all references to values of zero through FFFFH. References to addresses within the disassembled area may define subroutines, especially if there is more than one reference to the address. The number of references to each location is also displayed to the left of the location value on the disassembler listing, as shown in Fig. 4.

Code looks like a subroutine when a Return terminates the code segment, especially if registers are POPped from the stack directly before the Return. One can work backwards from the end of the subroutine to find the subroutine entry point(s), possibly indicated by a number of PUSHes to save registers.

Another method of finding subroutines is to scan the reference table or memory (using Debug or another monitor) for addresses associated with specific functions. This is the disk version of Scripsit, and we know that somewhere within the code are calls to the TRSDOS I/O subroutines. These are fairly easy to find at location 5E13H on, where there are consecutive Calls to INIT, Close, Open, Read and Write. The Disk I/O calls are described in the Radio Shack TRSDOS manual; almost all programs using disk will use the TRSDOS Disk I/O routines to avoid a non-standard disk I/O structure.

Other obvious addresses to look for in Model I code are 37E8H (printer), 3801H (keyboard row 0), 3C00H (start of video memory), 0EXH (serial port) and 0FFH (cassette).

Defining the Structure

At this point the structure of the program (or lack of it) has been fairly well defined. Don't expect the structure to follow a beginning, middle, and end format, or even an upper, middle, and lower level structure. In this case, Scripsit isn't bad, but even here calls to the line printer are sprinkled over 10K, instead of being in one

SAVE/ on Software and Hardware for TRS-80®

SOFTWARE

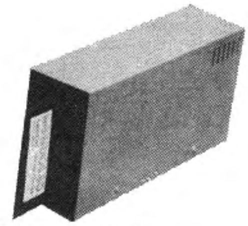
NEWDOS/80 version 2.0(specify Model I or III)	$149.00
Flextext I	$29.95
Utilize the additional features of the 737/739 under Model I or III Scripsit®	
Flextext II	$29.95
Utilize the additional features of the NEC spinwriter under Model II Scripsit®	
Flextext 80	(available soon)
Utilize the additional features of the MX-80 under Model I or III Scripsit®	
Chextext	$79.95
Spelling checker for Model I and III word processing	
Microclinic	$24.95
The ultimate in memory and disk diagnostics	
Meal Master	$24.95
Meal planning and grocery shopping aid	
Sprint 80	$14.95
Creates your own custom serial printer driver	
Professional Football Predictor	$29.95
Let your TRS-80 pick the winners	
Bionic Basic	$74.95
Expand your Model I disk basic features	
Interactive Business System	$500.00
An inventory control system for manufacturing	
Micro Accounting System	$489.00
The finest general business package we've seen	
HELP/CMD	$14.95
Adds "HELP" feature to NEWDOS/80 version 1.0	

NEW LOWER PRICES ON TANDON DISK DRIVES

Complete with power supply, chassis & configured for TRS-80® Model I or III

• Single sided 40 track	$295.00
• Dual 40 or Single 80 track	$425.00
• Dual sided 80 track	$515.00
• **Special** - 2 Dual sided 80's	$999.00

Model I DOS and drive package add $54.00
Includes above drives and NEWDOS/80 version 1.0-the established DOS.



MODEL III SUPER SYSTEM

Approximately 2.1 megabates of disk storage using 2 dual headed 40 track drives internally and 2 dual headed 80 track drives externally.

$3299.00 with NEWDOS/80 version 2.0
$1895.00 same as above with only 2 single sided 40 track drives(no DOS)

NOTE: These Model III computers contain Apparat installed disk drives and memory. They are warranted by Apparat, Inc. for 90 days.
Attention Dealers: Call or write for information on our new dealer discount package.

MISCELLANEOUS

Apparat's Data Separator	$29.95
16K TRS 80 or APPLE memory kits	$19.95
9½" x 11" blank white,tractor feed paper,full box	$24.95
9½" x 11" blank white,tractor feed paper,half box	$14.95
14½" x 11" green bar,tractor feed paper,full box	$34.95
3½" x 15/16" 'tractor feed mailing labels	$19.95
Model II add on drives	(call for prices)
Model II business software	(call for prices)

SERVICE CENTER SPECIAL

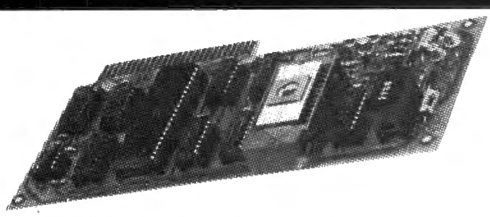
Disk drive tune-up $29.95
Complete-cleaning, alignment, and speed adjustment.
With any tune-up you may purchase your choice
1 box Verbatim diskettes $19.95
BASF head cleaner $9.95

PRINTERS

Call for our incredibly low prices on printers.
All printers come with free cable. We carry:
ANADEX CENTRONICS NEC
EPSON OKIDATA DAISY WHEEL

CABLES

Printer cables	$29.95
(specify printer and TRS-80 Model)	
Disk Drive Cables	
2 Drive	$29.95
3 Drive	$34.95
4 Drive	$39.95
Extender Cable	$12.95
Special Cables	(call for prices)



APPARAT'S PROM BLASTER

An eprom programmer for all 25 X X and 27 X X chips TRS-80® MOD I & III or APPLE $149.00
Bus Extender with power source, enclosure and 5 card slots $149.95
Mini version with 2 card slots and no enclosure $69.95

More Savings

MODEL I DOUBLE DENSITY PACKAGE

Converts a standard TRS-80 Model I to Double Density.
LN Doubler-NEWDOS/80 V.2.0 & box of 10 Verbatim Diskettes.
$330.00 value for only $269.00 • LN Doubler alone $149.00

DISKETTES

Double density-soft sectored-replacement guaranteed-Hard Hole protected. (5¼" only)

Verbatim Datalife 5¼" 40 track	$24.95
Apparat's No Name 5¼" 40 track	$21.95
Verbatim Datalife 8" Model II	$39.95

FLIP SORT SPECIAL

The ultimate file box filled with 50 diskettes 5¼"	$119.95
Flip sort alone	$29.95

VISA

Apparat, Inc. ✓264
"On-going Support for Microcomputers"

4401 S. Tamarac Pkwy. • Denver, CO 80237 • (303) 741-1778 • (800) 525-7674

Scripsit & TRS-80 are a registered trademark of Tandy Corporation.
Freight F.O.B. Denver-call for shipping charges. Foreign Orders shipped Air Freight

✓See List of Advertisers on page 354

functional module. The rough structure of Scripsit from the above tricks is shown in Fig. 5.

The structure of a program is dependent upon the standards of the programming department and/or the standards of the individual programmer. If these are lax, so will be the structure. There is at least one company that generates many different versions of software for different systems by a meta cut-and-paste technique that produces software and must be a horror to maintain. (But then again, this company only generates the initial version, and doesn't worry about maintenance.)

Making a Working Storage Table

The next step is to start a list of variables. You could probably find the starting point of the program and follow it from beginning to end, recording variables as you went along, but this is the hard way to go. Instead, start a list of memory locations roughly arranged in numerical order and jot down new ones as you encounter them. Scripsit appears to make many references to locations beyond the end of the program (beyond the disassembled area). We could reasonably assume that these are locations that hold variables—Scripsit's working storage.

Looking at the disk I/O calls in the 5E13H area, for example, we find that they are called from the code immediately prior to the calls, from location 5D23H on. DE holds the DCB address for disk I/O, and it's obvious that the DCB is located at locations 7CB1H. That location is marked Disk DCB on our variable list. A further nugget is gleaned by noting that an address of 7CD1H is used on the INIT call at 5DAFH. This must therefore be the disk buffer of 256 bytes. This makes sense, as the DCB is 32 bytes long in Model I TRSDOS. The 7CD1H is marked Disk Buffer on our variable list.

Variables or constants may sometimes be found within the program area, as is the case of the code at 6C81H. We found that this code was a binary to ASCII subroutine from its association with the ?M processing, which prints the amount of memory used. One technique of binary to ASCII conversion uses a power-of-10 table, and this is found at location 6CCAH with values of 2710H, 03E8H, 0064H, 000AH, and 0001H.

Another good clue to the location of variables is the initialization code. Just about every program will "initialize the world" on startup. Scripsit is no exception, and the code at 5247H initializes working storage variables in the 7CXXH area, in addition to some within the program itself.

Add to the variable list as you continue to figure out the code.

An interesting side note about disassembling Color Computer programs—in the Color Computer, the video display memory can be set to a 512-byte page anywhere in memory. It is possible to execute a program and watch a graphic display of variables changing, in color! This is truly state-of-the-art disassembly and debugging.

Looking for Special Characters

Another method of identifying processing within a program is to look for unique characters. As an example, look at Fig. 6. The 8FH character is used as the cursor character in Scripsit, and it's not hard to find references to it at 52A7H, 52D7H and 7A2DH, each of them associated with a load of the character into

```
52A1 0E3C      LD     C,3C
52A3 FD7135    LD     (IY+35),C
52A6 FD36078F  LD     (IY+07),8F
52AA FD360805  LD     (IY+08),05
52AE 218B6F    LD     HL,6F8B


52D5 E5        PUSH   HL
52D6 FD36078F  LD     (IY+07),8F
52DA CDCC5F    CALL   5FCC
52DD F5        PUSH   AF


7A28 56        LD     D,(HL)
7A29 21ED5C    LD     HL,5CED
7A2C FD36078F  LD     (IY+07),8F
7A30 C9        RET
```

Fig. 6. Looking For Special Characters

(IY + 07H).

Beyond Disassembly

You are now into the third level of disassembly. This is the analysis that is the most tedious. Each code segment must be scanned and commented. Check off each code segment you are sure of and put a question mark by others. A typical commented disassembly is shown in Fig. 7, which represents the ASCII to binary subroutine in Scripsit.

Of course, there is no reason to decode segments in which you have no interest. In my particular case, I was interested in adding a proportional spacing capability to Scripsit by processing the output from the P,I,S command (print invisible lines to serial device). I disassembled Scripsit and looked for all references to the line printer address of 37E8H and to the serial port address of 0EXH. The additional work on Scripsit stemmed from a desire to upgrade Scripsit.

Along with the analysis of each instruction, you may want to run the program you have disassembled with Debug. With the disassembly, it becomes fairly easy to breakpoint the program at proper places. Single-stepping may also be used if you are confused about program flow.

The ultimate in disassembly is editing and reassembling the source file derived from the disassembly and analysis. If you've reached this point, you now have a fully annotated source listing instead of the machine code you started with.

How long does a complete disassembly take? That depends upon the proficiency of the programmer and the complexity of the program to be analyzed. Assuming that he has good tools to work with, such as a disassembler with a reference table, a good Debug package, and moderately hard code, it may take hundreds of hours for a program the size of Scripsit. But then again, we're talking about a complete disassembly with comments for every instruction. To disassemble a program to add minor patches may only involve a dozen hours or so and be more than worth it to add desired capabilities.

Please feel free to write to me about disassembly methods or advice on disassembly tools. I can't, however, send you annotated disassemblies of any copyrighted software. Good luck in disassembly. ■

```
                                    ASCII DECIMAL TO BINARY
6C4C C5       PUSH  BC              SAVE REGS        ENTRY: HL→STRING
6C4D D5       PUSH  DE                               EXIT: HL = VALUE
6C4E EB       EX    DE,HL           PNTR TO DE
6C4F 210000   LD    HL,0000         CLEAR RESULT
6C52 08       EX    AF,AF'          SWAP??
6C53 1A       LD    A,(DE)   <-+    GET NEXT ASCII
6C54 13       INC   DE         |    BUMP POINTER
6C55 FE9B     CP    9B         |    TEST FOR?
6C57 2825     JR    Z,6C7E     |    RTN IF FOUND
6C59 FE21     CP    21         |    TEST FOR SPACE OR LESS
6C5B 3821     JR    C,6C7E     |    GO IF YES
6C5D CDFF53   CALL  53FF       |    TEST FOR LINE, PARA, PG, TERM
6C60 281C     JR    Z,6C7E     |    GO IF FND
6C62 FE2E     CP    2E         |    TEST FOR PERIOD
6C64 28ED     JR    Z,6C53     |    GO IF FND
6C66 FE2C     CP    2C         |    TEST FOR COMMA
6C68 2006     JR    NZ,6C70    |    GO IF DIGIT
6C6A 08       EX    AF,AF'     |    RESTORE
6C6B 3011     JR    NC,6C7E    |    RETURN IF?
6C6D 08       EX    AF,AF'     |    SWAP
6C6E 18E3     JR    6C53 ------+    GO FOR NEXT
6C70 29       ADD   HL,HL      ^    N*2
6C71 44       LD    B,H        |    SAVE N*2
6C72 4D       LD    C,L        |
6C73 29       ADD   HL,HL      |    N*4
6C74 29       ADD   HL,HL      |    N*8
6C75 09       ADD   HL,BC      |    N*10
6C76 D630     SUB   30         |    CONVERT ASCII TO BINARY
6C78 4F       LD    C,A        |    NOW IN C
6C79 0600     LD    B,00       |    NOW IN BC
6C7B 09       ADD   HL,BC      |    MERGE
6C7C 18D5     JR    6C53 ------+    LOOP FOR NEXT
6C7E D1       POP   DE              RESTORE REGS
6C7F C1       POP   BC
6C80 C9       RET                   RETURN
```

Fig. 7. Hand-Commented Disassembly

Special Grand Opening values on complete TRS-80™ Model III systems and disk drives, now through Sept. 15, 1981!

A.M. Electronics, Inc. has created the most powerful and reliable TRS-80™ hardware and software systems available to get *your* system up and running. Take advantage of special Grand Opening values like these:

New TRS-80™ Model III now with disk drives

Model III with dual 80-track disk drives—704K of reliable disk storage on only two drives. System features two double-density 80-track drives, 48K RAM, A.M. Electronics controller board. Complete, ready-to-run! **$2,295**

Model III with dual 40-track double-density disk drives, 48K RAM and A.M. Electronics controller board. Complete, ready-to-run! **$1,895**

DOS-PLUS operating system for TRS-80 Model III. **$69.95**

New! MAKE 80© now available for Model III

Converts Model I single-density 40-track or Model III double-density 40-track standard diskettes into Model III double-density 80-track-readable diskettes. **$29.95**

New Teac disk drives, complete with power supply, case and one-year warranty

Teac 40-track disk drive. **$265**
Teac 80-track disk drive with 10ms access time. **$375**
Teac 50F dual-headed 80-track disk drive with 10ms access time. **$499**

Disk drive cases & power supplies

5½-inch single drive unit with case and power supply. **$85**
5½-inch dual drive unit with case and power supply. **$120**
8-inch single drive unit with case and power supply. **$150**

More Grand Opening Specials!

16K 200 nanosecond RAM memory chip. **$2.00** each
Disk drive extender cable. **$8.95** each

Free Disk Drive Guide

Get your concise and fact-filled guide from A.M. Electronics, written in engaging Question/Answer style. Call or write us to order your Guide today! ✓452

Also, we specialize in complete, powerful software packages for business. Put your TRS-80™ to work for you!

Dealer/Service Center Record Package

This package, built in several modular packages, can be used separately, or in combination as a complete system. The modular units are self-contained. When combined, they will make the most complete and comprehensive service center bookkeeping/record-keeping package on the market.

A complete history of customer repair files is maintained on-line. Customer files can be accessed by account number or customer name. The information file includes customer name, address and telephone number, make of unit, service dates, name of servicing technician, services performed, and total billed. This information can be called to the screen or printer as a total list, or by the last service date.

The customer records module can post to accounts receivable module, payroll module and our *Inventory Control System.*

A Model I computer, with four 80-track drives, or a Model III computer, with two 80-track drives will hold approximately 1,800 customers and records. By adding additional drives to the Model III, more than twice this amount of information can be maintained. By switching diskettes in drives, the capacity is unlimited.

Versions of *Dealer Service Center Record Package* are available for Auto Dealers, Mechanics, TV Repair Centers, Electronic Technicians and Appliance Repair Centers. **$199.95**

Order Entry (With Invoicing & Inventory)

Orders are composed of name and address information (with optional company name), method of shipment, method of payment, items on order, quantities, prices, description, and stock number. The system also prints invoices.

At point of shipment, inventory is automatically updated and dollar sales recorded. Shipping charges are logged and shipping date recorded. All entries may be edited and partial shipping is permitted. The inventory module allows checking and updating stock, inventory reports, sales reports, shipping charges, etc.

Order Entry is a complete order entry system, designed for almost any retail and/or wholesale business. Its capacity is upward-expandable from a dual 40-track Model I system: 1,750 inventory items, 385 orders per diskette. **$199.95**

See your nearest A.M. Electronics, Inc. dealer

Here is a partial listing of authorized A.M. Electronics, Inc. dealers:

ARIZONA

Simutek Computer Products
PO Box 13687
Tucson, Arizona 85710

Microcomputers, Inc.
4322 E Fairmont
Phoenix, Arizona 85018

CALIFORNIA

RAC Products
3200 Knightswood Way
San Jose, California 95148

CONNECTICUT

Mountain View Software
Mirybrook Road
Danbury, Connecticut 06810

Computer Services of Danbury
14 Summit Street
Danbury, Connecticut 06810

DIST. OF COLUMBIA

Program Store
4200 Wisconsin Ave . N W
Washington, D C 20016

FLORIDA

Computer Worlds
2232 East Bay Drive
Clearwater, Florida 33516

Adventure International
507 East Street
Longwood, Florida 32750

All Systems Go
105 West Plant Street
Winter Garden, Florida 32787

ILLINOIS

3H Computer
1950 Bilter Road
Aurora, Illinois 60504

AAA Chicago Computer Center
120 Chestnut Lane
Wheeling, Illinois 60090

Midwest Computer Peripherals
1467 S Michigan Ave
Chicago, Illinois 60605

MASSACHUSETTS

Small Business Systems Group
6 Carlisle Road
Westford, Mass 01886

Omnitek Systems
1899 Main Street
Tewksbury, Mass 01876

Computer Plus
245A Great Road
Littleton, Mass 01460

MICHIGAN

Adapts —
600 28th Street
Grand Rapids, Michigan 49509

Remarkable Software
1508A Defense
Muskegon, Michigan 49441

James Buller
438 East Lake Street
Petoskey, Michigan 49770

Eight Bit Corner
722 Evanston Avenue
Muskegon, Michigan 49442

Soft Sector Marketing, Inc.
6250 Middlebelt Road
Garden City, Michigan 48135

Matrix Software
315 Marion Avenue
Big Rapids, Michigan 49307

Breeze Computing
PO Box 1013
Berkley, Michigan 48072

NEW HAMPSHIRE

Hardside
6 South Street
Milford, New Hampshire 03055

NEW JERSEY

Floppy Disk Services
40 Misty Morn Lane North
Trenton, New Jersey 08638

NEW MEXICO

Rocky Mountain Engineering
4749 Southern, S E
Albuquerque, New Mexico 87108

NEW YORK

John D. Owens Associates, Inc.
12 Schubert Street
Staten Island, New York 10305

B.T. Enterprises
171 Hawkins
Centereach, New York 11720

NORTH CAROLINA

Alpha Technology
1201 Wicker Drive
Raleigh, North Carolina 27604

OHIO

Electronics Unlimited
824 East 14th Street
Ashtabula, Ohio 44004

OREGON

1300 Centre Electronics
1300 6th Street
Umatilla, Oregon 97882

PENNSYLVANIA

Stevens Radio Shack
562 Nutt Road
Phoenixville, Pennsylvania 19460

TEXAS

Quality Software
11500 Stemmons Expressway
Dallas, Texas 75229

Inventory Control System

This system gives the user complete access to and control over an unlimited number of inventory items. (Note: It has been found that systems containing more than 13,000 items are difficult to handle, from a maintenance time standpoint. Operations, such as Block File reoganization can take more than 3 hours.)

As presently configured, a Model I computer with one 40-track drive will hold 1,925 items, or for a four-drive system a total of 6,625 items. Four 80-track drives will hold in excess of 13,000 items. A Model III computer, with two 40-track drives, will hold about 13,000 parts. If required, A.M. Electronics can modify both the program and hardware configuration to allow on-line storage of an unlimited number of items. As the file becomes larger, maintenance operations take longer. Average search time is six seconds, with 12 seconds the longest time.

Upon program initialization, the user specifies the item number and description digit length. This allows for item numbers up to 23 alpha-numeric characters. (As item number digits increase, digits for description usage are decreased.)

The program is completely menu-driven. Items can be added, edited or deleted from the file. Items can be placed on order, received to stock, or sold from inventory. Complete printout capabilities are available through the printout menu.

Item information includes: item number, description, supplier, re-order point, cost, *wholesale* and retail selling price, quantity, on-order, and total sold. The re-order point is calculated by the program, based on number of items sold over a period of time. The time period is established at program initialization. **$199.95**

Get a sample software printout—Free!

Call or write us to get sample program printouts for these programs, or if you need more information. We'll be happy to send it to you.

The power behind the drives®

A.M. ELECTRONICS, INC. ✓452

Ann Arbor: 3366 Washtenaw Ave., Ann Arbor, MI 48104 (313) 973-2312 Store Hours: Monday-Friday 10-6, Sat. 10-5
Grand Opening of our Radio Shack Authorized Dealership at: 111 North Grand, Box 1071, Fowlerville, MI 48836 (517) 223-7281
Attention Dealers, OEM's & Distributors: Call us for details on our attractive pricing.

VISA

Prices subject to change without notice.

✓See List of Advertisers on page 354

NEW PRODUCTS

Edited by Janet Fiderio

50 Programs for The Pocket Computer

50 Programs in Basic for Home, School and Office, is a collection of new software especially written for the TRS-80 Pocket Computer.

The booklet features such Basic programs as: Pocket Datebook, Profit Computer, Metric Converter and Deposit Doubler. Each will run without modification on the Sharp PC-1211 pocket computer and with only minor modifications on any Basic microcomputer.

The publication, priced at $9.95, is available from ARCsoft Publishers, P.O. Box 132EM Woodsboro, MD 21798.

Reader Service ✓ 325

Dairy Herd Management

Dairy Herd Computer Services has released a Dairy Herd Management program available for the TRS-80 Model II, 64K with printer.

With over twenty report categories for dairy herd management and statistics, this program records complete animal history, milk production, and medical history.

The program costs $699. A compatible calf program costs $179. For more information contact Dairy Herd Computer Services, 870 Mesa Drive, Arroyo Grande, CA 93420.

Reader Service ✓ 181

Mobile Entertainment Robot

The DC-2 is a mobile entertainment robot. It features complete mobility, model airplane-type radio controls, Sony color tv set/monitor, Audiovox components, and a turning head controlled remotely inside a protective dome.

If you wish more information on this mechanical man contact The Android Amusement Corporation, 2650 Myrtle Ave., B-7, Dootson Industrial Park, Monrovia, CA 91016.

Reader Service ✓ 180

Software Development Station

A new series of microprocessor development systems enables the Model I or Model III with at least 32K of RAM to serve as a software development station for a variety of single-chip processors.

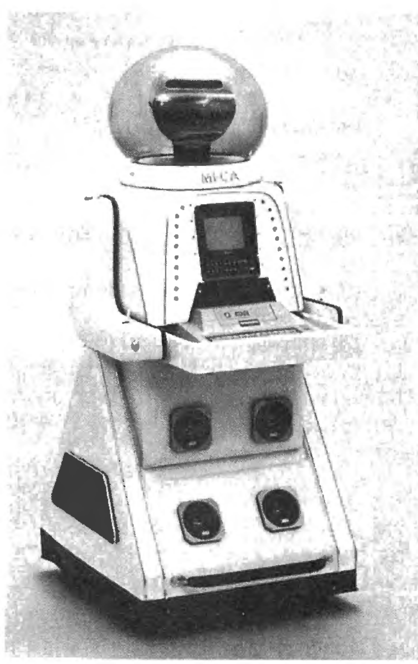
Each system is a unified editor/assembler and shares a common operational structure with uniform procedures for program entry, modification, assembly, and source file handling.

Each system is available for $75 on TRS-80 cassette (500), or the Model III disk. For additional information contact: Allen Ashley, 395 Sierra Madre Villa, Pasadena, CA 91107.

Reader Service ✓ 345

On Genealogical Topics

Genealogical Computing is a bi-monthly newsletter by the people who created the "Family Historians" bulletin board.



DC-2 Robot

It includes a directory of genealogy-oriented programs, information on computerizing family research records, program reviews and articles on genealogical topics.

Edited by Paul and Saralou Andereck, it is available for $2 an issue, or $12 for six issues from Data Transfer Assoc., 5102 Pommeroy Dr., Fairfax, VA 22032.

Reader Service ✓ 177

Medical Software

A new software package is available that permits complete patient and office record maintenance for medical offices on the Model I and Model III equipped with a printer and disk drives.

The Medical Office System Software can store up to 3960 patient accounts on the Model I, or up to 4200 accounts using the Model III.

The package is priced at $299 from Tandy/Radio Shack, 1800 One Tandy Center, Fort Worth, TX 76102.

Reader Service ✓ 334

New Model III System

Level IV products announces the TRS-80 Model III computer with four MPI model B-92 double head, double side, 80-track, 5 1/4-inch disk drives. The system sees them as eight separate 80-track drives.

Level IV has used the Radio Shack disk controller board, thereby maintaining full serviceability and compatibility with other standard software.

Prices begin at $1,859 from Level IV Products Inc., 32461 Schoolcraft Rd., Livonia, MI 48150.

Reader Service ✓ 174

It Takes a Thief!

It Takes a Thief is an adventure game in which you teach your computer how to be a thief.

The object of the game is to steal as many items of value as you can and get away.

A tape version for the 16K Level II and the 32K disk system are provided on opposite sides of a cassette, for $10 from RanDob Computer Paraphernalia, P.O. Box 1662, Boca Raton, FL 33432.
Reader Service ✓ 329

A Magazine on Cassette

Chromasette Magazine is a monthly magazine for the TRS-80 Extended Basic Color Computer. It is "printed" on a cassette tape.

The cassette contains six to eight programs that directly load and run on the Color Computer. Included each month are games, tutorials, utilities, and practical programs.

The cost is $45 for a one year subscription (12 issues) or $25 for a half year. For more information contact: Chromasette Magazine, P.O. Box 1087, Santa Barbara, CA 93102.
Reader Service ✓ 339

Table Top Robotics

MiniMover 5 is a table-top robot arm that can be run with a Level II TRS-80. Controlled by stepping motors, the arm is a five-jointed unit with a parallel linkage gripper and 17.5 inch reach.

Possible applications for the MiniMover range from computer games, computerized construction and computer assembly to computer art.

The Arm unit is priced at $1,695. For more information contact Microbot, 1259 El Camino Real, Suite 200, Menlo Park, CA 94025.
Reader Service ✓ 169

PMC-81

Robotics Reference And Applications Manual

The MiniMover 5 Robotics Reference and Applications Manual is available for $16.95.

The manual goes beyond the operation of the company's light-weight arm and covers other robotic units available on the market. It is written for those interested in learning manipulator operation; subjects include mechanicals, electronic/computer interfaces, software and mathematics.

For more information contact Microbot, 1259 El Camino Real, Suite 200, Menlo Park, CA 94025.
Reader Service ✓ 170

The PMC-81

Personal Micro Computers has added the PMC-81 to its TRS-80 Model I work-alike product line.

This micro has 16K of memory, 14K of ROM, utilizes a Z-80 chip and provides a keyboard, cassette interface and video monitor interface. The PMC-81 uses the EXP-100 Expander to add interfaces for minifloppy disks, printer, RS-232C and S-100 bus.

It retails for $740 or $939 with the VDM-81 green monitor screen from Personal Micro Computers, Inc., 475 Ellis St. Mountain View, CA 94043.
Reader Service ✓ 331

Mini Mover 5 and Reference Manual

On Musical Applications

The next time you want to compose music and experiment with a variety of sounds, read *Musical Applications of Microprocessors,* by Hal Chamberlain.

The author explores digital microprocessors, sound, music synthesis and applications for the more powerful 16-bit micros.

The book is available for $24.95 from the Hayden Book Company, Inc., 50 Essex St., Rochelle Park, NJ 07662.
Reader Service ✓ 328

Bookkeeping Products

Bookkeeper II, a new group of bookkeeping products for CP/M-based and Model II systems, is written in Microsoft Basic.

The software includes general ledger, accounts receivable and payable, fixed asset, payroll capabilities and more. It is also useful for maintenance of tax tables, W-2 forms, quarterly tax reports, and more.

Bookkeeper II is available at $600 per module or $1,500 for the complete package. Contact Data Train, Inc., 840 NW 6th St., Suite #3, Grants Pass, OR 97526, for more information.

Reader Service ✓ 327

MTI Offers Dual Disk System

Microcomputer Technology, Inc., is now marketing an upgraded Model III system.

MTI expanded the memory of the basic 16K Model III to 48K bytes and added the MTI double density, dual disk drive system. The system is fully compatible with the Radio Shack DOS and peripherals.

This product sells for $1,998 from Microcomputer Technology, Inc., 3304 W. MacArthur Blvd., Santa Ana, CA 92704.

Reader Service ✓ 175

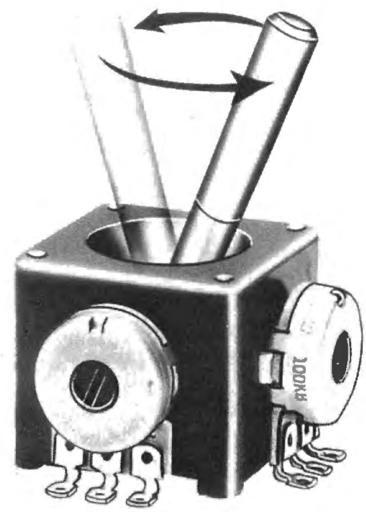
Compact Joystick Control

A joystick-controlled pair of 100K Ohm linear-taper pententiometers is now available.

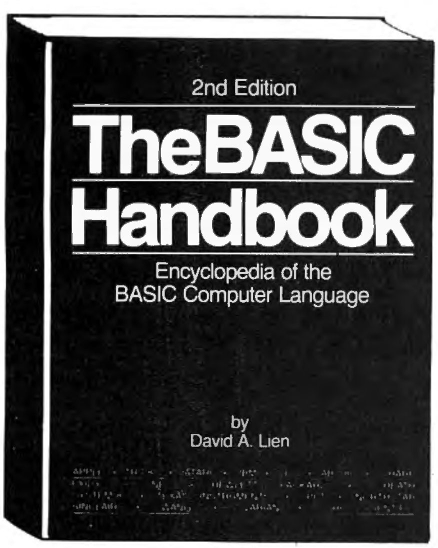
This compact (1 7/16-inch square) joystick control is intended for a variety of applications, from radio controlled model cars to video games.

It is available for $4.95 from Radio Shack stores. For more information, contact Tandy/Radio Shack, 1800 One Tandy Center, Fort Worth, TX 76102.

Reader Service ✓ 161



Joy Stick Control



The Basic Handbook

Mailing List Program

Demi-Mail is a TRS-80 Model I and III compatible mailing list program. It has three to five-line address formats, 26 record types in a single file, five and nine-digit ZIP codes, five print and display options, individual and global delete functions, user defined dates for each file and variable-size compressed data fields, to provide disk efficiency.

It can also merge name and address information with word processing files for list processing.

It is available for $24.95 from Demi-Software, 6 Lee Road, Medfield, MA 02052.

Reader Service ✓ 165

New Basic Handbook

The Basic Handbook (second edition): An Encyclopedia of the Basic Computer Language, by David Lien has just been released by CompuSoft Publishing.

More than a revision, the handbook has been reorganized and researched resulting in a definitive compendium of almost 500 Basic words.

The Handbook is available for $19.95 from CompuSoft Publishing, 1050-E Pioneer Way, El Cajon, CA 92020.

Reader Service ✓ 171

Etch—Sketch With the TRS-80

Etch allows you to create pictures with graphic dots on your video screen.

The 4K Level II version will save pictures on tape and the 16K disk version saves pictures on disks which can then be Listed by TRSDOS. Both versions are provided on one cassette.

Etch is priced at $9 from RanDob Computer Paraphernalia, P.O. Box 1662, Boca Raton, FL.

Reader Service ✓ 160

The Memory Box Adds Memory

The Memory Box is a memory expansion unit that provides 1024 bytes of RAM in an unassigned area of memory (3000–33FF hex) and can be plugged directly into the CPU or expansion interface.

Two utility programs written to reside in the Memory Box are included in the package, for $64.95 from Displayed Video, 7538 Jackson Road, AnnHarbor, MI 48103.

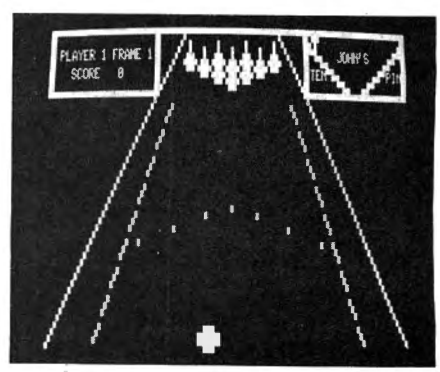
Reader Service ✓ 335

Bowl with the TRS-80

Tenpins is a machine code bowling game for one to four players. This graphic game has realistic action and sounds, and beginner and advanced levels. Scoring, pin setting and ball return are all automatic.

It is available for the Model I or III on cassette for $14.95, or on disk for the Model I for $20.95. Inquiries should be directed to Acorn Software Products, 634 North Carolina Ave., S.E., Washington, D.C. 20003.

Reader Service ✓ 347



80 Bowling

Daily Tribun

.040 EDITION ★★★★ FORECAST: H(

RETURNS ARE IN. CPAIDS VOTED MVP.

Title of "Most Valuable Program" Bestowed on Master Tax Package

Barely four years after its introduction into the microcomputer market, CPAids Master Tax software is being praised by hundreds of accountants as their Most Valuable Program.

Accountants nationwide have prepared thousands of returns using CPAids Federal 1040 software and today CPAids announced their enhanced programs for 1981.

The most notable addition to all CPAids software, the Fixed-Asset Accounting Program collects data year round and carries information over to the appropriate schedules and forms at tax time. Continuing an era of efficiency and cooperation, General Ledger II interfaces with the Master Tax Program to generate Schedule C. The Master Tax Program also generates a proforma tax organizer, providing necessary information from the previous years' returns.

The Time Is Right For CPAids

In an informal interview, CPAids dedicated its products to eliminating the frenzied feeling plaguing accountants during tax season and vowed to serve professionals well, not just at tax time, but all year.

Authorized dealers have pledged their continued support and offer a *free* 1981 update to those who subscribe now.

MASTER TAX $1500

—Federal 1040 pages 1 and 2
—Schedules A, B, C, D, E, F, G, R, RP, SE, TC, ES
—Forms 1116, 2106, 2119, 2210, 3468, 3903, 4136, 4255, 4625, 4726, 4797, 4874, 4972, 5695, 5884, 6251, 6252
—Client instruction letter
—Exclusive CPAids print options:
- preprinted forms
- overlays
- generated computer forms
- printing on government forms

—Loan amortization program
—Input sheets

Also available this year, our new 1120 corporate program.

GENERAL LEDGER II $450

—General ledger, payroll fixed-asset accounting
—Loan amortization
—Stores all transactions
—Payroll interactive with the general ledger section
—Statement of changes in financial position
—Compilation letter
—Fixed-asset program stores up to 10,000 assets

CPAids 1040 tax software basks in early victory amidst cheering throng of enthusiastic well-wishers

For only $55 CPAids will send you a demonstration disk and manual.

Call or write for free brochure and the name of the dealer nearest you.

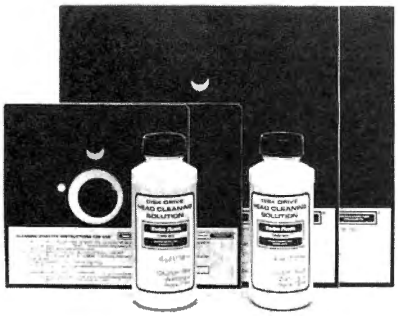
CPAIDS®

1640 Franklin Avenue,
Kent, OH 44240 (216) 678-9015

Programs require a minimum memory of 56 Kb. Two disk drive, CP/M version 2.2 Runs in Microsoft Basic version 5.2

Compute Improve as Tax

Studies sh the intensit complete a dimensions that IRS w
No two s it comes t three yea what Am is on the



Disk Cleaner

Cleaning Kits for Disk Drive Heads

Radio Shack is offering disk drive head cleaning kits which permit the cleaning and maintenance of the read/write heads of Model I, II, and III disk drives.

Two kits are available, one for 5 1/4-inch disk drives, and one for 8-inch disk drives.

The price is $29.95 from Tandy/Radio Shack, 1800 One Tandy Center, Fort Worth, TX 76102.

Reader Service ✓ 173

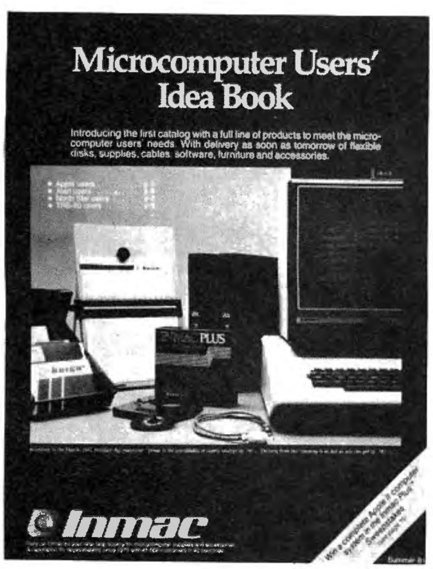
User's Idea Book

The Microcomputer User's Idea Book is a catalog dedicated to meeting the entire supply, accessory, and cable needs of micro users.

The 32-page publication is broken into separate sections by major microcomputer companies. In addition, extensive cross-referencing shows compatibility within systems.

A free one-year subscription can be obtained by writing: Inmac, Department 12, 2466 Augustine Drive, Santa Clara, CA 95051.

Reader Service ✓ 179



Idea Book

Graphics From the Line Printer VII

The Line Printer VII offers quality impact printing and dot addressable high-density graphics for $399.

The printer also features: either 80 or 40 upper and lowercase 5 x 7 dot matrix characters per eight-inch line at 30 characters per second; adjustable width and plain paper printing; self re-inking ribbon; and switch-selectable standard interface options.

For more information contact Tandy/Radio Shack, 1800 One Tandy Center, Fort Worth, TX 76102.

Reader Service ✓ 168

Color-Coded Files with Compu-Key

Wenner Business Systems has introduced a filing system that businesses can use with desk-top computers or word processors.

Compu-Key is a computer-generated color-coded filing system which enables you to set up files alphabetically, numerically, by date, territory, and more. The computer codes these sections with colored blocks, while keeping track of purge dates or other data for future recall or management reports.

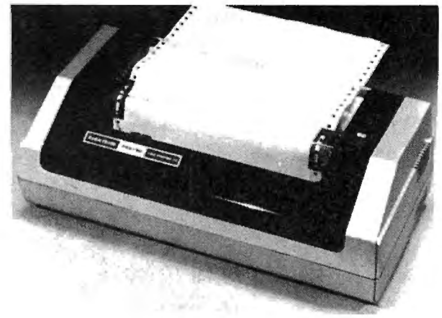
A box containing 1,000 labels is priced at $56.50. For more information, contact Wenner Business Systems, P.O. Box 831, State St., Los Altos, CA 94002.

Reader Service ✓ 330.

Litigation Support Software

The new Litigation Support software package from Radio Shack is of direct interest to law firms, legal offices and attorneys.

The package assists the attorney in implementing a flexible storage and retrieval filing system. Litigation Support creates a client file and a forms file, both of which provide complete report generation.



Line Printer VII

This software requires a 64K Model II computer and a 15-inch wide tractor-feed line printer. It is available for $299. For more information contact Tandy/Radio Shack, 1800 One Tandy Center, Fort Worth, TX 76102.

Reader Service ✓ 336

Micromouth Says 143 Things

Micromouth is a TRS-80 compatible limited-vocabulary voice synthesizer which uses the new Digi-Talker DT1050 chip set from National Semiconductor to store a vocabulary of 143 expressions.

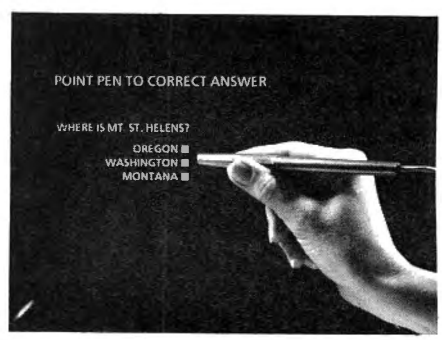
Micromouth plugs into the keyboard, screen printer port or expansion interface and programs directly in Basic.

It comes with case, power supply and interconnecting cable in kit form for $150, assembled for $175 from Micromint, 917 Midway, Woodmere, NY 11598.

Reader Service ✓ 176

Light Pen for Models I and III

A self-contained light pen which plugs directly into the Model I and III has been



Light Pen

✓See List of Advertisers on page 354

announced by the 3G Company.

The pen makes it possible to bypass the TRS-80's keyboard and interact directly with information displayed on the CRT screen.

A demonstration game cassette, sample program and programming booklet are included with the pen. The package sells for $39.95 from the 3G Company Inc., Rte. 3, Box 28A Gaston, OR 97119.

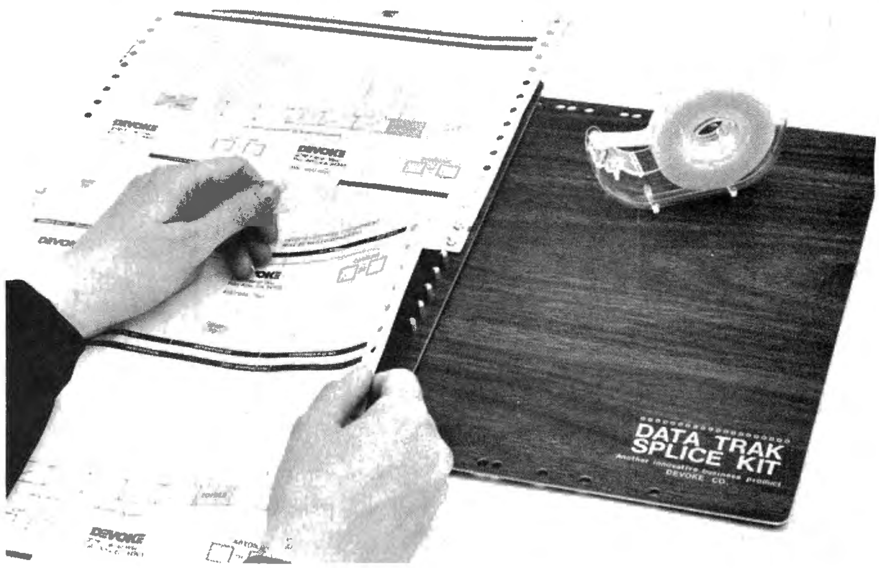
Reader Service ✓ 338

Fast, Precise Splicing

The Data Trak Splice Kit makes fast, precise splicing of two or more cartons of pin-feed forms possible. This kit is most useful where unattended printers are used on long runs.

Data Trak is designed to handle 16 different widths of pin-feed forms (from 4 1/4-inch to 14 7/8-inch). The cost is $38.50 from Devoke Data Products, 3780 Fabian Way, Palo Alto, CA 94303.

Reader Service ✓ 162



Data Trak Splice Kit

An Offset Printing Estimator

The Offset Printing Estimator is used in the design and final stages of cost estimating black and white offset printing.

It is intended for use by business executives, consultants, and advertising executives in the planning stages of preparing sales literature, bulletins, catalogs, and price lists.

The Estimator runs on the Model I and the Model III, and is available for $500 from Pasadena Technology Press, 3543 E. California Blvd., Pasadena, CA 91107.

Reader Service ✓ 178

Operating System Modification

Cordos is a modification for the Model II operating system that allows fully transparent use of rigid disk drives, such as the Corvus Winchester drive and the Cameo cartridge drive.

Up to 98 volumes, or logical drives, may be used. Cordos has facilities for password protection for the full drive, as well as a volume-level password protection.

The price is $300 for an eight-inch floppy disk from American Business Computers, 118 South Mill St., Pryor, OK 74361.

Reader Service ✓ 340

Gold Edge-Connectors For the Mod I

The Gold Plug '80 Kit is a micron over nickle edge-card plug that is soldered directly to the existing edge card plug on the Model I CPU and Expansion Interface.

The kit eliminates problems that occur due to oxidation of the tin and lead surface of the Model I expansion ports.

Gold Plug '80 kits are available for keyboard/CPU to expansion interface for $18.95, or expansion interface to disk, printer, screen printer, or RS-232 ports for $9.95 from E.A.P. Co., P.O. Box 14, Keller, TX 76248.

Reader Service ✓ 166

Cassettes Formulated By Agfa-Gevaert

Displayed Video is now marketing certified data cartridges made of top grade tape formulated by Agfa-Gevaert of Germany.

These microcomputer cassette tapes are certified to be 100 percent error free and carry an unconditional money-back guarantee.

The price for the C-10 length 12-pack is 68 cents each, and for the 24-pack, 58 cents each. The C-20 12-pack sells for 88 cents each, 78 cents each for the 24-pack. For more information contact Displayed Video, 7538 Jackson Road, Ann Arbor, MI 48103, 313-426-5086.

Reader Service ✓ 344

Color Computer Programs

Two new editing programs for the Color Computer are written in M6809 machine code, available on compatible TRS-80C machine code cassette tape.

The Text Editor features screen editing commands to allow insertion, deletion, changing or adding to text lines.

The second program is a co-resident editor/assembler that allows the user to create, edit, and assemble machine code programs.

Their prices: the Text Editor, $19.95; the co-resident Editor/Assembler, $39.95. Both programs as a package cost $49.95 from Cer-Comp, 5566 Ricochet Ave., Las Vegas, NV 89110.

Reader Service ✓ 333

Sketch Complex Graphics

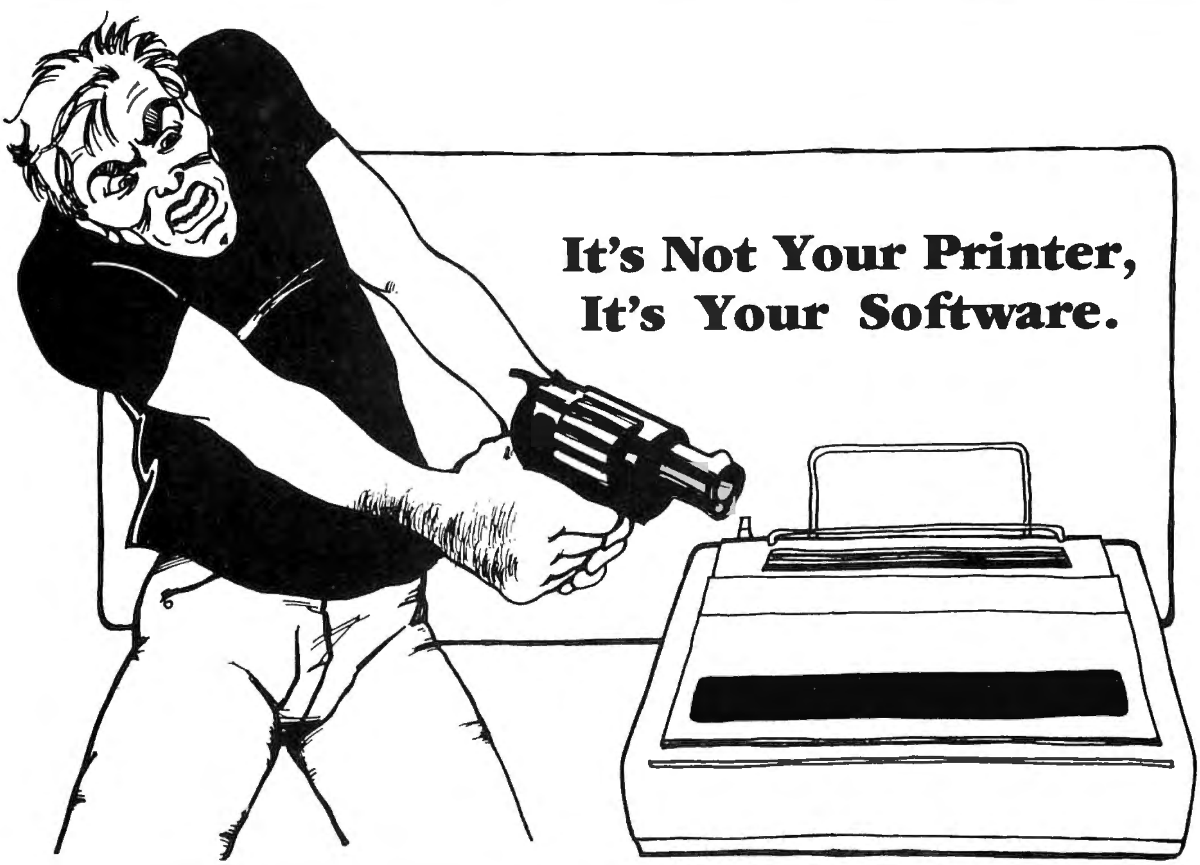
Graphic Writing Machine enables users to easily sketch complex graphics displays using simple keyboard commands.

The program features two internal memories which provide temporary storage for the screen contents. The sketch can be saved to tape or disk. GWM will also generate Basic statements so that the image may be included in a Basic program.

Versions are available for 16K and up, Model I or III, on tape or disk, from Micro Pro Systems, Rte. 2 Box 533, Cummings, GA 30130, 404-887-6814.

Reader Service ✓ 164

ASYLUM
ATARI 400 COMPUTER TO BE AWARDED TO ESCAPEE!
Asylum is the most sophisticated, sinister, challenging 3-D graphics adventure ever written! You are placed in an asylum for DEATHMAZE survivors. Your only goal: *ESCAPE!*
Asylum has over 1200 locations! Hallways recede into the screen, *as though you are actually there!* Doors open and close! Beds, desks, and even guards and other inmates are drawn on the screen! And graphics are generated instantaneously! Simply press an arrow key to move or turn, and you are moved. Absolutely *no* waiting!
English language commands may be issued at any time from a vocabulary of almost 300 words. Commands may include complete statements. *"Get everything on the desk except the matches"* is perfectly legal!
Will anyone escape? Who knows? Intelligence, cunning and patience will be necessary. Send us your correct solution, postmarked no later than November 15th. On November 30th we will draw from the entrants for an ATARI 400 16K color computer!
NO HINTS WILL BE GIVEN OVER THE PHONE!
ASYLUM 16K cassette $ 14.95
ASYLUM 32K disk $ 19.95
ASYLUM Hint Sheet $ 1.00
TRS-80 Level II, Model I and Model III only
Med Systems Software
P.O. Box 2674-A ✓128
Chapel Hill, NC 27514
(919) 933-1990



It's Not Your Printer, It's Your Software.

Features of Qwerty 3.0

1. Automatically prints in proportional print, with a suitable format. Transitions between the three print styles are easy, including all expanded print modes.
2. **Qwerty 3.0** adds 75 new symbols, including upper and lower case Greek letters, mathematical symbols such as integrals and summations, arrows, brackets, and probability symbols.
3. Any character can be used as a subscript or superscript, even simultaneously. Carats, bars, and tildes can be placed over any character, with precise position control.
4. Underlining, with or without underlining of spaces, including long ratios and mathematical expressions.
5. **Footnotes** can be placed on any page so that they remain on the desired page, even if text is inserted later.
6. **Table** commands enable positioning of the print head anywhere on a line. Invaluable in printing neat mathematical layouts, tables, columnar material, etc., in proportional print.
7. **Pretty** commands allow printing of repetitions of a chosen character. When combined with **Table**, decorative borders can be produced with ease.
8. **Folio** format produces output in two or three columns per page, in either proportional or 16.7 cpi mode. Ideal for newsletters.
9. Supplies a third output mode, in which only **Scripsit** commands are obeyed. Allows printing of special **Qwerty** commands for future reference.
10. **Page End** indicates where pages will end, and the page number, without printing the text. One can prepare an almost error-free document without ever using paper.
11. Correction of **Scripsit**'s errors and inconveniences, extensive documentation, and much more!

Do you own a Centronics 737? 739? Lineprinter IV? Tired of the limitations of SCRIPSIT? **Qwerty 3.0** will let you accomplish word-processing feats you never dreamed possible. Like Greek letters. Simultaneous subscripts and superscripts. Even footnotes and two or three column folio formats. See the list of features in the box. No other piece of software of this type can match **Qwerty 3.0**. We know, because we bought the others, and were disappointed.

Qwerty 3.0 is more than "fully tested." It has seen thousands of hours of use in a university environment. A masters thesis and a statistical doctoral dissertation were produced and accepted right off a Centronics 737 using this package.

Qwerty 3.0 is the finest SCRIPSIT addition available for the Centronics 737, 739, and Lineprinter IV. You must own one of these printers, a copy of SCRIPSIT, and a disk drive to use it. It is the best. Period. WE GUARANTEE YOU WILL AGREE! If for some reason, you find that this program doesn't meet your needs, return the entire package within 14 days for a prompt and cheerful refund.

For cautious buyers, we offer the manual (about 50 pages) for $10.00. When you decide to buy **Qwerty 3.0**, we will credit the full manual price.

Qwerty 3.0 Disk, Manual, Reference Card, and Printer Table Rule $49.95

Qwerty 3.0 Manual Alone $10.00

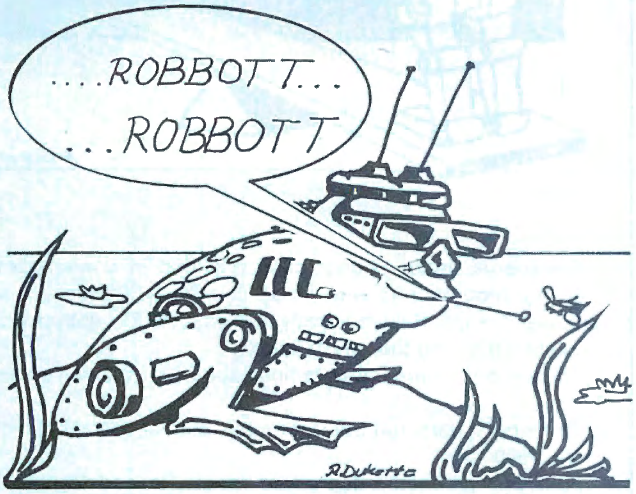
Med Systems 128
(919) 933-1990

✓128

✓See List of Advertisers on page 354

It's All Robotese to Me

by Calvos Gisamte



Stories about robots and mechanical persons are nearly limitless, which is great if you're being paid by the word. However, discretion—while not my middle name—is something I'm not ignoring nearly as much as I used to. So I decided to give this article some direction. But, which way? Suggested topics I mulled over and subsequently discarded (tentatively) are:

A Robot Glossary

Acrobots: somersaulting mechanical bipeds which do that flying-trapeze bit as effortlessly as you or I might tumble into an open sewer. Also known as Robocrobats.

Automaton and *Automatoff*: robotic states of consciousness and unconsciousness. Sometimes called Automatron and Automatroff.

Automatonomat: coin-operated robot diner with terrible pies.

Automatonous Rule: preferred government of sentient microchips.

Bot: a robot hallucinogen, macrobotswana (capekis sativa). Users are called Bottops.

Bureaubots: officials of the sentient microchip government, prone to stamping "UNUSUAL" on all documents. Bureaubots are managers of Roblivion (viz.).

Crockbots: part of the French Maid Package; stew cooks all day in its belly while the Crockbot dusts the china and straightens beanbag chairs.

Crowbots: birdlike amusements for the garden; volume can be adjusted for all environments.

French Maid Package: luxury robot kit from RUR Industries that cooks, cleans, curtseys, and speaks in a foreign language. See details under Crockbots, Radiobots, and Robotisserie. Prurient models are available on special request (to adult over 18 only).

'Frobots: hair stylist robots designed by RUR industries.

Frogbots: mechanical amphibians, whose nightly serenade goes, "robbott . . . robbott . . . "

Great Robot Mystery: whatever happened to Erector Sets?

Harmless Robots: mythical characters (the Robbits) in the Lord of the Springs.

Herobots: general term for highly esteemed robot leaders; also, a kind of layered food inside a long, flour-based crust.

Introbot: sentient microchip dating service, part of RUR Industries.

Karel Capek: affectionately referred to as "Daddy" by all robots; head of RUR Industries.

Microbots: general-purpose miniaturized robots for human edification; microbots can be used for keeping teeth and tongues clean, scratching insect bites, cleaning fingernails automatically, etc.

Nerobots: activated during fires, these robots provide soothing, randomly generated violin music.

Radiobots: not to be confused with radio-controlled robots, these are an option in the French Maid Package. May not be used simultaneously with Stereobots (viz.) due to competitive programming that results in ear-splitting easy listening.

Roblivion: storage area for damaged or discontinued robots, maintained by a corps of Bureaubots.

Roboat: passive relaxation vehicle designed to carry robots across the River STX to Roblivion.

Robo-Coupe: a twentieth-century food processor which has nothing to do with this glossary.

Robo Robozo: financier of RUR Industries.

Robosco: chocolate topping for milktabs.

Robotcrobat: see Acrobots.

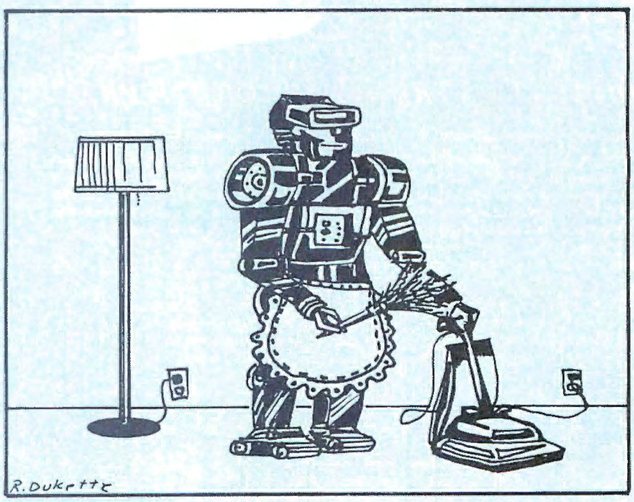
Robotcrats: see Bureaubots.

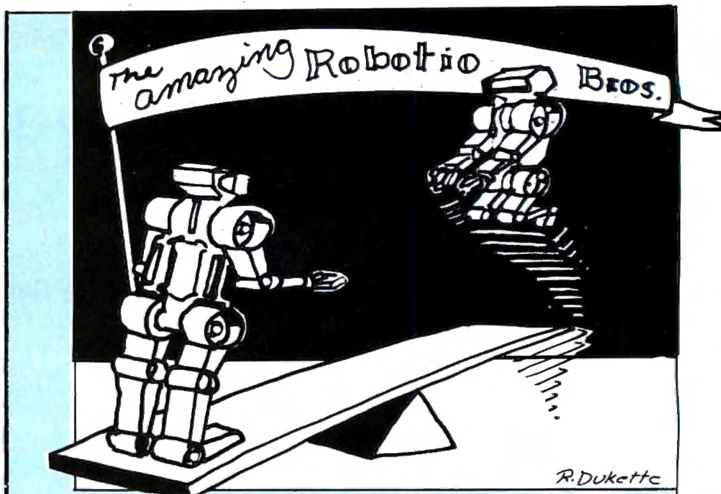
Roboticelli: famous robot artist who painted "Venus on the Half Chip."

Robotisserie: part of the French Maid Package from RUR Industries; roasts chickenclones for the automatonomat.

Robottles: glass containers for Robosco. Also, a femiclone nickname.

RUR Industries: theological mainstay of robots, headed by Karel Capek.





Stereobots: robot soundclones provided in contemporary housing; programmed with mood sensitivity, Stereobots are capable of responding musically to any mood. Stereobots must be disabled during therapy sessions.

Thoreaubot, Henry David: famous author of "Walden Version 3.2."

Throwbots: personal frustration toy, similar to the pet rocks of the 1960's.

Zerobots: perenially depressed robots; leased to humans who are in need of therapy, Zerobots are capable of countering any depressing human story with an even more depressing one.

Zorrobot: swashbuckling star of holofilms, known only to his confidants as Don Data.

The Robot/Vegetable Connection

Did certain gardens in Mexico cultivated by the Mayans really influence the development of the Univac?

What exactly is the effect of disk drives on hothouse radishes?

Can organic humus turn the tide in computerized robot tennis?

Is Wayne Green truly a prophet of cybernetic doom or merely a cog in a computerized cantalope wrestling scheme?

Robot Jokes

A robot goes into a bar and orders a martini. The bartender serves it and says, "There y'are; that'll be six bucks. Y'know, we don't get many automatons in here." The robot fishes some currency from his utility pouch and responds, "And at these prices, it's no wonder!"

Robots Are Your Friends

Actually, that's not necessarily so. Take aberrant behavior, for example. A human suffering from an attack of schizophrenia can compensate by snorting a couple of Sominex, taking a week off from gainful employ, or by a quick trip to Dr. Loon's black couch. But when a robot goes off on a mechanical snit, watch out! Those laser eyes aren't exactly twinkling when a titanium arc is extended to squash your shoulder socket as easily as it might hiccough!

Other Possibilities

Robot cosmetics; robot ballroom dancing; or robot music (editorially speaking, we already have some of that being ballyhooed over certain "progressive" airwaves). Alas, these topics are of but marginal interest. ■

In the interest of those readers who are not marginal (and at the request of the editors), the author ended his robotisizing here, saving the fully sentient world from yet another robot story.

Dungeon Escape

At last!! An adventure which takes quick thinking and strategy! Different from fixed adventures.

- **Super graphics** (with comprehensive map on screen)
- **Interactive Sound** (directions included for use)
- **Uses latest programming techniques** (Machine routines, etc.)
- **Fast Paced** - the creator's ghost has sensed an intruder, and he is determined to hunt you down. (His intelligence in tracking was set by you). This feature makes this simulation unlike others for you're always "on the go."
- **Written by an experienced gamer** in fantasy role-playing, based on the game Dungeons & Dragons.

Objective: Your character begins on the first of a three level dungeon, searching for magical "stones" which permit you to descend to the next lower level.

The Adventure: There are numerous traps, and over a dozen hostile monsters that come in various sizes, shapes and degrees of nastiness. It has various treasures and magic items (weapons, elixers, cloaks, scrolls, etc.) You can become one of three classes, a fighter, a thief or a magic-user. Each class has its own advantages and disadvantages. Each time you play you get a totally different and exciting game.

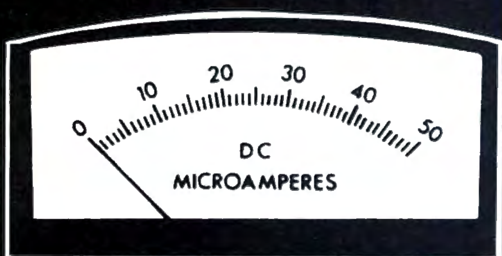
This short description only begins to tell you of the many adventures and some of the features of this exciting game.
This will be your favorite adventure game!

Complete documentation included.

You must specify Model I or Model III. Avialable on:

Tape, 16K $14.50
Diskette, 32K $19.50

★★★ SUPERMETER ★★★



Finally, a device to let you set correct volume levels for loading any tape. The biggest problem in loading tapes has been the volume control. Prerecorded tapes are produced at differing volume levels. You will now load any tape the FIRST time. **SUPERMETER** plugs in (no cutting or soldering) between your tape recorder and the computer, and lets you set the volume to the level that your computer wants.

SUPERMETER $29.00

DISKS

Buy the very best diskettes. Percom ran extensive tests during their research and design of the doubler and reached the conclusion that WABASH made the best 5 ¼ inch double density diskettes.

Why buy a diskette that just barely passes the ANSI certification tests when you can buy the best diskette on the market at these discount prices?

These disks EXCEED all ANSI standards. These are the newest, most improved disketttes from WABASH (including an extra smooth polished surface for greater head life). Wabash quality is UNSURPASSED in the industry. These diskettes are so good they are guaranteed for 2 years. If for any reason you are not happy with these superior diskettes return them for a prompt refund. (80 track diskettes can run in a 40 or 35 track disk drive.) Price is per box of 10 disks.

Single density, one side $25.95
Two or more 24.95
Double Density 80 track, one side 32.99
Two or more 31.99
Double density 80 track, two sided 35.99
8 Inch Floppies, Double Density 39.00

PERCOM'S DOUBLER II

If you haven't got a doubler for your Model I, order one now. this is the greatest thing to come along for the Model I. Since it actually reduces disk read/write errors, your system will run better with a PERCOM doubler. And you will also save both time and money by having more programs per diskette.

PERCOM'S DOUBLER II $159.00

NEW MODEM GAMES

Play Checkers, Chess, or Othello over your modem with your friends! Each game features full graphics, sound, and a chat mode permitting you to send messages and sounds (to get the other persons attention). Games are for two players, either one or two computers. Complete documentation. All three:

Tape .... **$14.95** **Disk** **$19.95**

Super Soft

1691 Eason
Pontiac, Michigan 48054

Feel Free to Call to Request more information on any product

(313) 673-2224
or
(313) 673-8700

(Dealer Inquires Invited)

Add $1.50 per order for Shipping & Handling

Phone Orders Welcome

Please call between 8-2 E.S.T.

Same day Shipment!

Robotics—The Microcomputing Connection

by Chris Brown
80 Microcomputing Technical Editor

For most of us, the word "robot" conjures up the image of a cute wheeled being jabbering away in a flurry of synthesized speech, while rolling obediently along behind a handsome starship captain. Though the precocious activities of R2D2 and C3PO may have endeared them to the popcorn set, the stereotype they represent is far from the reality of present-day robotics. Currently, the level of intelligence these fantasy figures exhibit can only be created in one way—by stuffing a human midget (preferably British) into a robot suit. It is only after seeing an intelligent robot, its wheels spinning, jammed hopelessly into a corner or catapulting headlong down a flight of stairs, that one realizes that intelligence, as it applies to robots, is a relative term.

Implicit in any concept of robotics is the notion of performance. Except when employed as a test bed for artificial intelligence research, robotics should have as its end goal the accomplishment of some useful activity. When a robot is combined with a microcomputer, a primitive synthesis of mind and body occurs that results in a robot's ability to replicate simple human activities. In our industrial society, the ability to duplicate operations ad nauseum is of great value. Thus, the employment of robotics gains new and increased importance in human terms.

Though the field of robotics has a long history, it is only recently that the microcomputer has made inroads. And today's use of microcomputers only scratches the surface of the possible. As computing power and memory storage continue to decrease in cost and increase in capacity, their use in robotics will become widespread.

Industrial Promotional and Educational

Robots fall into three general categories based on their function; industrial, promotional and educational. Industrial robots are highly specialized devices, usually taking the form of arms, that are capable of per-

Photo 1. Android Amusement's DC-2 show robot is advertised as a "crowd builder." It uses an on-board Atari 400 microcomputer to generate graphics and play games.

forming tasks akin to those of their human counterparts. They are used in applications like paint spraying, spot welding, picking and placing tasks. Operating under computer control, industrial robots can move accurately along six axes and, when combined with tv cameras, enjoy a primitive equivalent of vision.

Promotional robots (or show robots as they are known in the trade) are the machines we cautiously accept literature from at computer shows, see at the opening of shopping malls and watch with fascination on tv talk shows. (In the latter case, the robots have often displayed more intelligence than the host.)

Show robots do not, as a rule, have the intelligence of industrial or educational robots. Often they are nothing more than large radio controlled models with microcomputers in their chests. They have had an impact on the robotics industry, however.

Educational robots are used in classrooms to illustrate basic mathematical or abstract geometric concepts.

All three types of robots are on the verge of impressive advances because of the now popular microcomputer and the cheap computing power it represents. Cost reductions in both memory and microprocessing have made it possible for robot manufacturers to offer very reasonable prices for on-board intelligence.

In addition, the proliferation of microcomputers has stimulated new interest in the field of robotics as many users endeavor to justify the existence of their machines. When a microcomputer, in union with a robot, can instruct a machine to water the lawn, mix a drink or oversee a manufacturing process, the microcomputer achieves

Industrial Robots

Industrial robotics, a common occurrence in many countries, remains an unusual application in the United States. U.S. labs are developing robots as complex and well-built as those in any country, but those machines so far have not wormed their way into the hearts of potential owners.

Jack Shimek of New Developments Co., Nashua, NH, is, therefore, a pioneer. He is attempting to sell New England factory owners on a simple industrial robot made by Applied Robotics Co. of nearby Hudson, NH.

The product he is selling has little in common with the popular idea of a robot which walks, may talk, and is self-controlled. The 95-pound Series 100 Industrial Robot is a simple machine by lab standards. It has one arm with a choice of grippers. The arm has two vertical positions, a two-position horizontal stroke or in and out movement, and a four-step swing that can move up to 270 degrees.

The machine is air powered, normally working under 60 pounds per square inch pressure. It cannot move and does not even include a built-in microprocessor controller.

Although there is only one working model of the little air-powered machine in existence, Shimek is optimistic about its future. He sees its very simplicity as a big asset.

"Everybody in manufacturing is getting into robots, but they are all afraid of putting a lot of money into a machine that will break down," Shimek said.

Tough Environment

The problem is that a manufacturing plant is an extremely rigorous environment for a robot. In a lab an experimental machine can be delicate. It is essentially nothing more than a demonstration machine. If it works at all and does what its designers want even for a short while they are happy.

A manufacturer, however, wants dependability. He wants a robot that can load and unload a drill press 500,000 times a day, five days a week, 50 weeks a year with no trouble. A delicate machine, one which is difficult or expensive to repair, or which uses exotic, expensive and hard-to-get components, can spell trouble, Shimek argues.

He points out that the Series 100 robot is built simply and ruggedly. It is air powered from an external compressor which, among other things, means that if someone or something gets in the arm's way, the machine will neither break the object nor do large amounts of damage to itself when its arm jams.

It is built out of commonly available components, Shimek said. The only parts that are special are those like the frame and collars which are not likely to wear out.

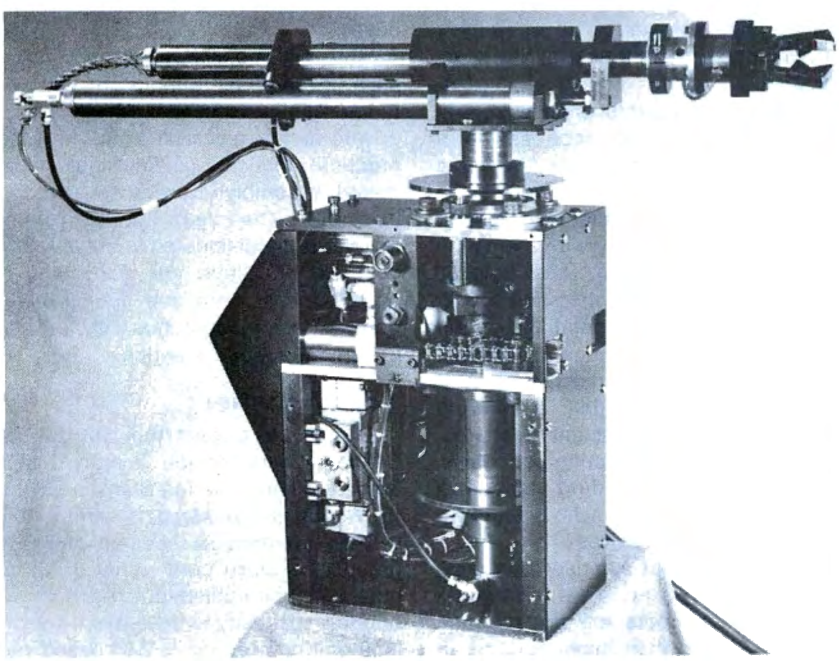
"All this is is a bunch of air cylinders put together in a usable configuration," he said. "It really demonstrates the point that the technology for a robot exists and has existed for some time."

According to Shimek, the robot is ideal for loading and unloading machines and doing other operations which involve picking up items one at a time from a set place and putting them down in another set place. If the expected item is not where it is supposed to be, however, the machine has no way of finding it. Its only sensor is a short-range affair that tells the machine whether it has something it its "hand."

Shimek said the designer, Arthur Haines, purposely did not build a controller into the robot because customers would want to use the microprocessor they already own, in many cases.

Shimek is using a 5-TI industrial controller from Texas Instruments to run his demonstration robot, and this machine can run several Series 100 robots at once. However, any industrial controller or computer, including a TRS-80, could be used to do the same job.

Shimek said the potential market for industrial robots in the U.S. is huge. "Many plants are making high technology products like computers using methods from the turn of the century."■



The Applied Robotics Co. Robot

"After seeing an intelligent robot catapulting headlong down a flight of stairs, one realizes that intelligence is a relative term."

new legitimacy. Its union with process-oriented robots in real world/real-time applications is a step in the direction of practicality. Lately, interest and activity have been on the increase in the robotics industry as entrepreneurs recognize the potential of the robotics market.

A Mobile Minibar

A leading company involved in the manufacture of show robots is Android Amusement Inc. of Monrovia, CA. Android Amusement is one of the first manufacturers of show robots that incorporates microcomputers in their creation. Its best known effort thus far is the Drink Caddy. This alcohol dispensing android is designed to perform yeoman service at cocktail time. The Drink Caddy is advertised as a "mobile minibar and entertainment machine" and carries a price tag of $4,998. Original versions of this machine did not feature an onboard microcomputer and were marketed as technological curiosities. One eventually found a home in the Playboy mansion in Chicago with other marvels of technology like electronic pinball machines and computerized beds.

An expensive offspring of the Drink Caddy, the $10,000 Ralph Rodger Robot, incorporates an Atari 400 microcomputer and monitor. The computer is placed in Ralph's chest and used for graphics generation and game playing. However, none of Ralph's vital functions are controlled by the computer.

Gene Beley, the youthful president of Android Amusement who once sawed a TRS-80 in half in a futile effort to fit it into a robot, feels that show robots represent a new art form in America that incorporates the wonders of technology with the gaudiness of the carney world. Beley told *80 Microcomputing* that his robots are "sexy" marketing tools that are finding increasing application in public relations and promotion. He also indicated that, for the moment, demand is outpacing supply at Android Amusement. With frank insight he characterizes his company's function as follows: "Basically, we are building expensive status symbols for people who want to attract attention to themselves."

Beley admits he was surprised by the initial success of his venture but feels confident that show robots are going to be money makers for their manufacturers in this era of high technology snobbery. For Beley, a major milestone in the robotics industry was reached when his company was able to employ mass production techniques using standardized parts in the making of robots. As yet, no robots are used to assemble robots at Android Amusement but Beley says the possibility exists.

A Bit of Overkill

Another successful manufacturer of show robots is The Robot Factory of Cascade, Co. The Robot Factory produces several anthropomorphic entertainers, some of which incorporate microcomputers for added pizzaz.

Mary Bolner, Robot Factory vice president, candidly states that the current use of microcomputers in show robots is overkill. While her company does offer an Apple II microcomputer in one of its robots, the computer is there for entertainment purposes only. The Robot Factory intends to have a robot which responds to voice commands available soon. To this end, they have been negotiating with Tandy Corp. in hopes of coming to some agreement on hardware procurement. Bolner said "We already use many Radio Shack parts in our robots. We find them to be high quality, reasonably priced and readily available. I hope we can come to some working agreement with them soon." The Robot Factory's microcomputer controlled speech recognition robot may be the first real microcomputer/robot marriage in the show robot industry.

More serious and potentially lucrative uses of robots lie in industrial and educational applications. In both public and private sectors research is underway to find practical applications for robots in automated assembly procedures and in the classroom. The role of the microcomputer has yet to be delineated in these infant industries, but there are many who have gambled their faith and finances on the hope that microcomputers will play a major role in automating America.

A Natural Marriage

One such consortium is the robot Builders Group of the Boston Computer Society. Headed by Ted Blank, a columnist for *Robotics Age Magazine* and a 10-year veteran of robotics research, the Boston group is a loose partnership of six computer/robotics aficionados, each of whom brings a speciality to the project. The group is attempting to build a Z-80 based, microcomputer controlled robot arm with substantial lifting capacity. The project is in the design stage now but once the objectives of

"This alcohol dispensing android is designed to perform yeoman service at cocktail time."

the group are set they intend to incorporate, seek venture capital and go into the business of manufacturing industrial robots.

In Ted Blank's opinion there is a natural marriage that must occur between microcomputers and robots. Blank told *80 Microcomputing* that the robotics industry is just beginning to address the essential applications that take place in real world/real-time settings. In many instances progress has been hampered by the reluctance of those in robotics research to share information. Says Blank, "A sense of secrecy pervades the entire industry because everyone is looking for the big payoff."

'Everyone' includes toy manufacturers, major universities like Stanford, MIT, Carnegie-Mellon, and large research institutions like The Stanford Research Institute, (SRI) International and the Hughs Research Laboratories, all of whom have substantial robotics projects in the works. Blank also feels that the development of practical robotics has been stifled by an overemphasis on artificial intelligence within the robotics community. He described the situation as "playing God by duplicating human brain functions as closely as possible." In Blank's view this preoccupation is taking attention away from the true problem area of robotics research, i.e., real-world/real-time applications. He provides as an example a West Coast university's robotics project that resulted in a highly intelligent machine that incorporated sophisticated visual recognition with large amounts of on-board data processing. The robot could move through an obstructed field by avoiding obvious hazards. The problem was that the data processing involved took so much time that the robot could only manage three feet of travel every 15 minutes. While certainly on the forefront of technological innovation, such a sluggish machine is, at best, impractical in most industrial settings.

For Ted Blank and the others involved in his fledgling robotics company, the future of the industry lies with the microprocessor. He concedes that new, high-level languages must be written to provide the control that complex networks of microprocessors, running in multi-tasking/multi-processing environments, will require. The result, the attainment of extremely high levels of intelligence and control in tomorrow's robots, will be worth the effort in Blank's view.

Presently, however, Blank and his group meet several nights a week to hash out the more mundane aspects of their project, such as who will deal with the servo-mechanical and hydraulic problems of the beast.

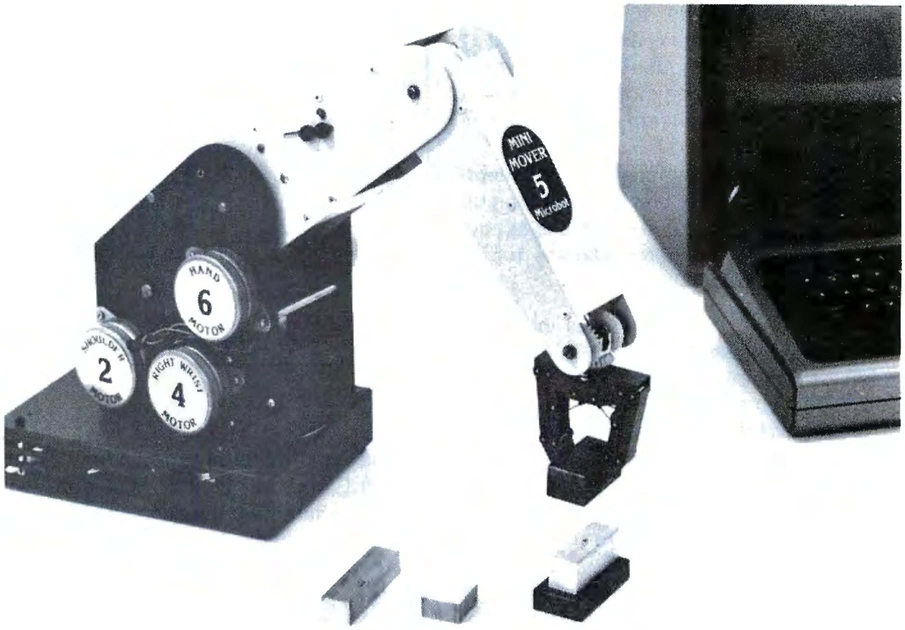


Photo 2. The Mini Mover V robot from Microbot Inc. The arm has an eight-ounce lifting capacity and connects directly to a TRS-80 microcomputer.

A TRS-80 Meets a Robot

The Boston group's project has been anticipated to some degree by a West Coast company called Microbot, Inc. Microbot, of Menlo Park, CA, is one of the only companies currently manufacturing a robot arm that can be connected directly to a microcomputer and easily programmed to perform pick and place tasks. The Mini Mover V Arm will plug directly into a Level II TRS-80 and respond to programs written in a language called Armbasic.

In an article that appeared in the summer, 1980 issue of *Robotics Age Magazine*, John Hill, chief design engineer and president of Microbot, described his robot arm as a low-cost ($1,700) manipulating system. He envisioned its use by light manufacturing industries and by those who are involved in robotics and artificial intelligence research. Hill's design objective was to provide a complete robotics package including arm, computer interface, software for coordinate conversion and control, and a high-level programming language. His intent was that the user's needs not undertake a full-blown development effort to put a robot into operation.

The Mini Mover V is a segmented arm mounted on a stationary base with a lifting capacity of eight ounces. A system of differential gears, pulleys and cables allow the arm five degrees of freedom within a radius of 17.5 inches. The drive unit for each joint (base, shoulder, elbow and wrist) consists of a stepping motor, reduction gear and cable drum. The robot arm is controlled by signals generated by a TRS-80. These signals pass through an interface where they are organized in seven four-bit parallel outputs and one four-bit input. The computer controls the four-bit phase patterns sent to the arm's stepping motors and thus, the arm's motion.

Armbasic

Armbasic, the machine's programming language, is an extension of Radio Shack's Level II Basic. It adds five commands to the Level II repertoire which enable the arm to move in various axes as well as open and close its grippers. These commands are Step, Close, Set, Reset and Read. When used in the context of a program, proper syntax requires that each be preceeded by the @ (at) character and succeeded by a delay expression and expression to indicate the number of steps each motor is expected to move. A typical instruction line

"The youthful president. . . who once sawed a TRS-80 in half in a futile attempt to fit it into a robot. . ."

would look as follows: 10 @STEP(D), (J1), (J2),. . . . (J6). D is a delay expression and J1 through J6 are instructions for the number of steps each stepping motor is to make. A program to perform a simple pick and place task appears in Fig. 1.

All Armbasic commands are implemented in a Z-80 Assembly program to speed up execution. The extension command set occupies about 1K of memory at the top of Level II Basic.

```
5 REM PROGRAM FOR SIMPLE PICK AND PLACE TASK

10 @STEP S, -P1, -P2, -P3, -P4, -P5
20 @CLOSE
30 @STEP S, P1, P2, P3, P4, P5
40 @STEP S, Q1, Q2, Q3, Q4, Q5
50 @STEP S, R1, R2, R3, R4, R5
60 @STEP S, 0, 0, 0, 0, 0, GR
70 @STEP S, -R1, -R2, -R3, -R4, -R5
80 @STEP S, -Q1, -Q2, -Q3, -Q4, -Q5
90 GOTO 10
100 REM S SPECIFIES SPEED, GR = GRIPPER STEPS
```

Fig. 1

Designer Hill feels that since industrial robots are largely computers in terms of cost, low-cost microcomputers will result in low-cost robots. He optimistically told *80 Microcomputing* that a huge market awaits the innovative and that "drastic" price reductions are in the offing.

Hill envisioned tomorrow's robots using individual microprocessors to control each joint. These ganged processors will make possible applications that are unheard of today. In addition, Hill foresees more people getting involved in robotics through microcomputing. He is confident great strides will soon be made in the field simply due to the number of people becoming involved.

Hill's young company has already felt the pressure of incipient success. A large midwestern manufacturing firm has been trying to acquire 51 percent of Microbot by dangling the venture capital carrot in front of Hill.

Tom Clark, president of Michigan Rivit, the firm attempting the acquisition, makes no bones about his desire to get involved in the robotics business. He told *80 Microcomputing,* "Do you realize that most of the component parts of your car weigh less than a pound? And that virtually all of the welding and paint spraying on GM assembly lines is already done by robots? Hey, it's just a matter of time until a beefed-up unit similar to the Mini Mover V will be in every modern manufacturing plant in the country."

Tom Clark is slightly baffled by Hill's and Microbots' reluctance to sell out. It is likely that John Hill has had the same vision as Tom Clark however, and he intends to hang tough through the lean times in hopes of the big payoff.

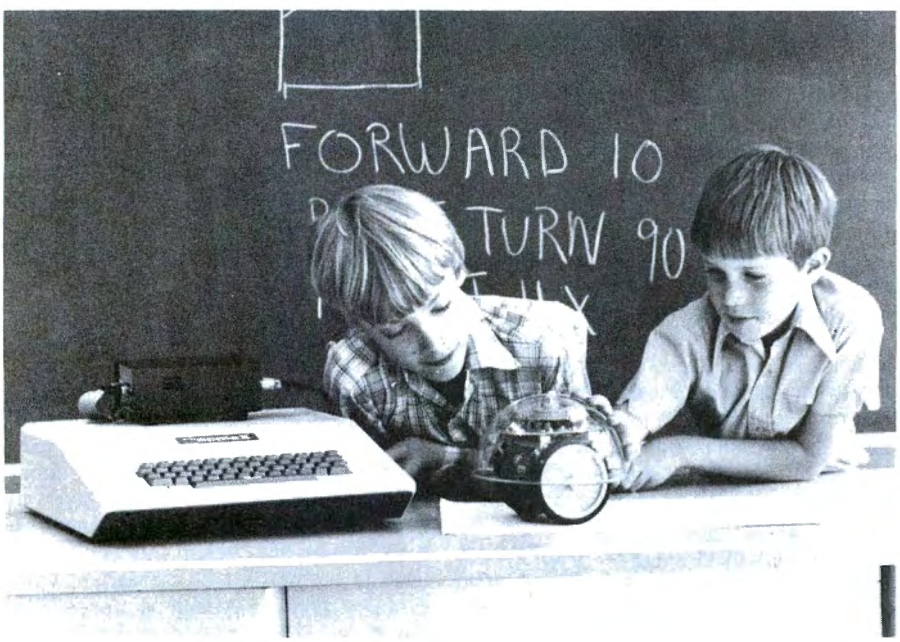


Photo 3. The Terrapin Turtle has found a home in the classrooms of America where it is used to illustrate the concepts of mathematics and the social sciences. It has also been used in work with autistic children.

Half a Grapefruit on Wheels

One of the more unusual locales in which computerized robots have been found is the classroom. Here, a homely looking robot manufactured by a Boston-based firm called Terrapin, Inc., has been gaining the attention and respect of educators in some of the nation's more progressive schools. Known as the Turtle, Terrapin's robot is an updated version of a machine created by English experimenter Grey Walter in 1940. Under the auspices of MIT's artificial intelligence lab, researcher Dan Hillis developed the modern Terrapin Turtle in the early 1970s. He decided to market it in 1978.

The Turtle, a small wheeled robot that connects to a microcomputer by a 12-conductor umbilical cord, resembles half a grapefruit on wheels. It has blinking eyes, an audio tone generator, touch sensors and a tracing pen that can be raised or lowered on command. The Turtle's drive wheels are independently controlled so that it may travel in either a straight line or curving trajectories, according to the commands it receives in Turtle Talk, its high-level programming language. Turtle Talk also provides commands to raise and lower the tracing pen, activate the touch sensors and sound audio tones.

Terrapin currently manufactures several versions of the Turtle for Apple, Atari and S-100 bus microcomputers. Prices vary from $400 for the kit to $600 for a fully assembled Turtle. Interface and software can add an additional $200 to the price tag. At present, Terrapin does not offer a specific interface for the TRS-80. Turtles have

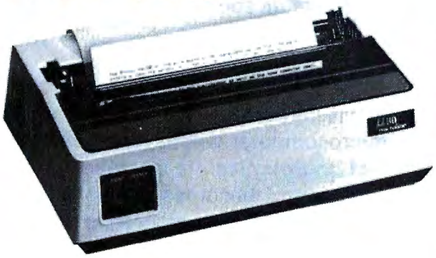
Select the write printer –

PRESTO!

Overnight your computer becomes a word processing, data processing or graphic system.

The NEW OKIDATA line of bidirectional dot matrix printers – high quality with continuous duty 200-million character printheads. The NEW printers have correspondence quality with their new 9x9 dot matrix heads and have full forms control, serial and parallel interfaces, as well as many other features. The Microline 80 is a very reliable, low cost printer. All products are available from our stock at The Stocking Source along with the optional cables, interfaces and memory buffers.

Under $500



Okidata
Microline 80 – 7x9 Head,
80 Column, Low Price

NEW
Microline 82A – 9x9 Head, 80 Column, 120 CPS

NEW
Microline 83A – 9x9 Head, 136 Column, 120 CPS

NEW
Microline 84 – 9x9 Head, 136 Column, 200 CPS

NEW
Model 2350 – 9x9 Head, 136 Column, 350 CPS

ASK FOR OUR INSTANT DISCOUNT From Roy Hawthorne Talk To Bill Tokar On Applications	CALL TOLL FREE U.S.A. 1-800-521-2764 MICHIGAN 1-800-482-8393	Remember: We are open 8:30 AM to 6:00 PM EST Monday – Friday	master charge VISA No Credit Card Penalty	"The Stocking Source" 24069 Research Drive Farmington Hills, MI 48024 313-474-6708 ✓438

STEREO AND PERCUSSION!

- STEREO—Separation by instrument! For example, play trumpet and oboe through channel A, clarinet and organ through channel B. You can switch instruments from channel to channel at any time!
- PERCUSSION—Now you can add a wide range of percussive sounds and special effects to your music!
- Existing Orchestra-80 files load and play in stereo automatically!
- Optional 5th Voice for use with speed-up mods!
- New editing features and commands!
- Plugs into any 16k Model I keyboard or expansion interface without voiding warranty!
- Includes tape and disk versions on cassette, 4 sample music files, manual, and fully assembled and tested printed circuit board!

Orchestra-85™ $129.95

plus $2.00 postage and handling
($5 for overseas airmail)
CA residents add 6% sales tax
MasterCard and Visa accepted

WRITE FOR INFORMATION ON:
- **Upgrading**—Orchestra-80 to Orchestra-85. Switch to Stereo!
- **Greatest Hits**—Volumes I and II. All new music! Ready to load and play!
- **Orchestra-90**™—Model III version of Orchestra-85. Available soon!

Software Affair
✓356 858 Rubis Drive, Sunnyvale, CA 94087
(408) 295-9195

"Robots represent a new art form that incorporates the wonders of technology with the gawdiness of the carney world."

been adapted for use on '80s by enterprising users, however.

According to Terrapin's promotional literature, the uses of the Turtle are limited only by the user's imagination. While Turtles have been used to teach the fundamentals of computer programming and robotics, their more interesting uses are in the fields of mathematics and the life sciences. By programming the Turtle to trace geometric figures on a sheet of paper, students actively participate in the creation of the concepts of geometry, gaining, for instance, the insight that a curve is made up of an infinite number of straight lines. (Ergo the differential!)

Programming brings up mathematical issues and, as programming problems are solved, students engage in mathematics the way mathematicians do; on a conceptual and intimate level. Where the life sciences are concerned, the Turtle can be made to simulate human behavior. By using directional feedback gleaned from its touch sensors, the robot is able to interact with its environment according to societal rules authored by its programmers.

A particularly intriguing use of the robot has occurred with autistic children. Research indicates that these children relate better to the world of machines than to the world of men and the Turtle has served as a bridge between these worlds by encouraging autistic children to verbalize and communicate.

Terrapin president Sheridan McClees told *80 Microcomputing* that his company sold approximately 100 Turtles last year. He also indicated that a wide cross-section of people have been using his robots in elementary and secondary schools, as well as in the home. When queried about Terrapin's lack of hardware for the popular Radio Shack family of microcomputers, McClees indicated that there are two basic reasons why Terrapin has not pursued the '80 connection. He said, "The main reason is that our engineers feel that it is much easier to build interfaces for other computers due to the TRS-80's design idiosyncrasies. Secondly, Radio Shack is not the easiest company to work with in terms of technical cooperation."

"A sense of secrecy pervades the entire industry. . ."

McClees did imply that Terrapin was rethinking its TRS-80 interface policy and hinted that an announcement would be forthcoming in October. He also said that Terrapin plans to double the size of its staff, (currently three full-time employees) and make a more sophisticated version of Turtle Talk available this fall.

The Malaise

It is inevitable that the proliferation of the microcomputer will have a measurable effect on Robotics. As increased processing and control becomes possible, the incorporation of microcomputers into robots will be more commonplace. In addition, as commercial applications like the Mini Mover V, Terrapin Turtle and even the Drink Caddy achieve visible commercial success, entrepreneurs will be encouraged to take the financial plunge and get into the business of robots.

The present malaise affecting American productivity will supply added impetus to robotics research. American plant operators, many of whom now look over their shoulders toward the East for cues on management, realize that Japan is the most automated country in the world. Once the assumption that all of our industrialized society's ills can be cured by robots takes hold in the board rooms of America, the field of robotics may explode as the field of computers did 10 years ago.■

TRS-80* Computer Owners...
Great New Access Store!

NEW, Fully Tested AFD Mini-Disk Drives from $275.

- Every Access Floppy Disk drive is electrically tested, mechanically tested and burned-in 48 hours under operating conditions. Look for the signed test list in the shipping carton.
- AFD drives are capable of *either* single- or double-density operation. AFD-100™ drives store 102-Kbytes in single-density, 180 Kbytes in double density. AFD-200™ drives store 205 Kbytes in single-density, 364 Kbytes in double density. And these are *formatted* file storage capacities. Moreover, you can store twice these amounts on a single diskette using AFD "flippy" drives (AFD-100F™ or AFD-200F™ drives).
- 15-day free trial offer . . . comprehensive 90-day limited warranty . . . unbeatable prices . . . AFD-100 @ $275.00, -100F @ $329.00, -200 @ $429.95, -200F @ $449.95.

NOW! *Percom* Drives for Your Model III !

Internally mounted drives that feature "flippy" capability, double- *or* single-density storage, comprehensive pre-delivery testing, 48-hour burn-in and, best of all, Percom's widely acclaimed disk controller. A First-Drive System includes the four-drive disk controller, one or two drives, cables and a fully illustrated owners manual. Use Model III TRSDOS*, Percom OS-80† and other Model III compatible DOSs. Prices: Complete First-Drive System w/1 drive @ $560.00, First Drive System w/2 drives @ $860.00.

Note: If you prefer not to make the First-Drive System installation, we will do it for only $29.95, plus shipping (if shipped). This includes a 48-hour operating burn-in of your Model III computer, installation of the drive system and final checkout of the expanded system.

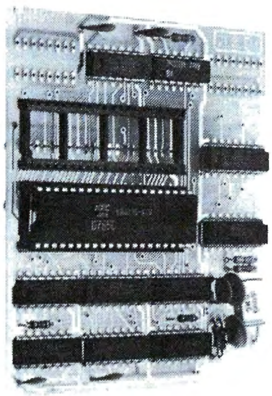
How to order — Order by calling Access Unlimited toll-free on **1-800-527-3475**. Or order by mail. Orders may be charged to a VISA or Master Card account, or paid by cashier's check, certified check or money order. COD orders require 25% deposit. Sorry, we cannot accept personal checks. We pay shipping and insurance charges on orders over $1,000.00. Add approximate insurance and shipping charges for orders under $1,000.00. If in doubt about these charges, ask when you call in your order. Texas residents include 5% sales tax. Minimum mail order $20.00. Allow 2 to 4 weeks for delivery.

◇ Technical & Texas: **(214) 690-0206.**

Ask about *quantity discounts* for 'TRUGs!

Use your credit card and save. VISA and Master Card charges are not deposited until the day your order is shipped.

Same Old Savings Galore.



Upgrade Your Model I to Double Density

only $159^95

And not with some unproven hacker's kit, either. But with an assembled and fully tested *Percom DOUBLER-II*†. The DOUBLER-II† simply plugs into the disk drive controller socket of your Expansion Interface. Then it lets you store over four times more on one side of a diskette, depending on the drive, than you can store on standard Tandy Model I drives. Other features: • Reads, writes and formats *either* single- or double-density • Includes DBLDOS™, a TRSDOS* compatible disk-operating system • Runs TRSDOS*, NEWDOS††, Percom OS-80† and other single-density software immediately. Change to double-density operation when convenient • Lets Model III software be read on your Model I • Includes on-card high-performance data separation and write precompensation circuitry.

Microline-80 Printer

Grand-Opening Special $399.95 (REGULAR $446.00)

Compare the Microline-80 feature-for-feature with *any* other competitively priced printer. • Uses standard roll paper, fan-fold forms or individual sheets. • Prints expanded characters and *graphics*. • Prints 96 ASCII characters (9x6 dot format), 64 graphic characters (12x6 dot format). • Throughput is 80-characters per second, standard-width characters. • Print head is rated at 200 million characters. • Compact, lightweight, quiet.

ACCESS Goes Accessible!

No longer a mail-order-only operation, we're in a new, walk-in retail mart on Central Expressway in suburban Dallas. Now *that* makes us really ACCESSible!

Come by.

Get acquainted with Sam or Connie.

Check out our computer servicing capability. We not only "service what *we* sell," we're also an *authorized Percom Service Center.*

You'll notice, too, that besides our TRS-80* department, we've added a "Hobbyist Corner." For rack-jobbing selected computer parts. Moreover, we're now carrying the full Percom System-50 (SS-50) product line. And selling books. And magazines. And . . . well, just come by and see for yourself.

PERCOM
COMPUTER SERVICE CENTER

If you can't come by, at least call about our August GRAND OPENING SPECIALS.

Make ACCESS your one-stop shopping center for TRS-80* computing. From game programs to sophisticated system software. From power line filters to double-density adapters. From Model I add-on drives to Model III First-Drive Systems. Shop ACCESS and save!

Call our toll-free order number for free descriptive product literature and latest prices.

PRICES AND SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE.

Toll-Free Order No.

1-800-527-3475◇

(orders & literature only)

ACCESS UNLIMITED

401 N. Central Expressway #600
Richardson, Texas 75080
(214) 690-0206

™Trademark of Access Unlimited
†Trademark of Percom Data Company, Inc.

✓229

*Trademark of Tandy Radio Shack Corporation.
††Trademark of Apparat Corporation.

Build a light, sensitive, voice-controlled companion for your computer.

One Man's Robot

Don McAllister
4709 Rockbluff Drive
Rolling Hills Estate, CA 90274

Last year I decided to interface Kenner's R2D2 robot to my TRS-80. It has its own radio control system, is well built and inexpensive. I knew I could mount various sensors to the robot. With a hardware interface and the following software, R2D2 now responds to my voice as well as infrared light to execute a preprogrammed set of motions.

Not bad for a $29 robot and a $34 interface and power supply.

All of the following analog to digital conversions are performed in machine language embedded in single-line strings of Level II Basic. This eliminates the need to load System tapes into protected memory and offers you the ease of Basic programming and the speed of machine language.

The robot controls its two-dimensional movement by latching OUT port data bits 0 and 1. The latched bits control the relays with a hex inverter-driver integrated circuit. These relays, in turn, key R2D2's radio control mechanism and let it move in any preprogrammed pattern.

The robot also responds to infrared light. By means of a sensor infrared data is input to the CPU in serial form by using only data bit D7. There it's converted to a number proportional to voltage by the machine language analog to digital string in Basic.

Software controls the scanning device. Specifically, the software turns the robot in increments of 7.5 degrees, alternating this with periods of infrared data collection. When the robot sees a light with any intensity above a software-controlled threshold, he tracks the light moving toward its source, until he reaches it and the light becomes diffused. Then the robot turns off.

If you wish to control the robot by voice you must press play and record at the same

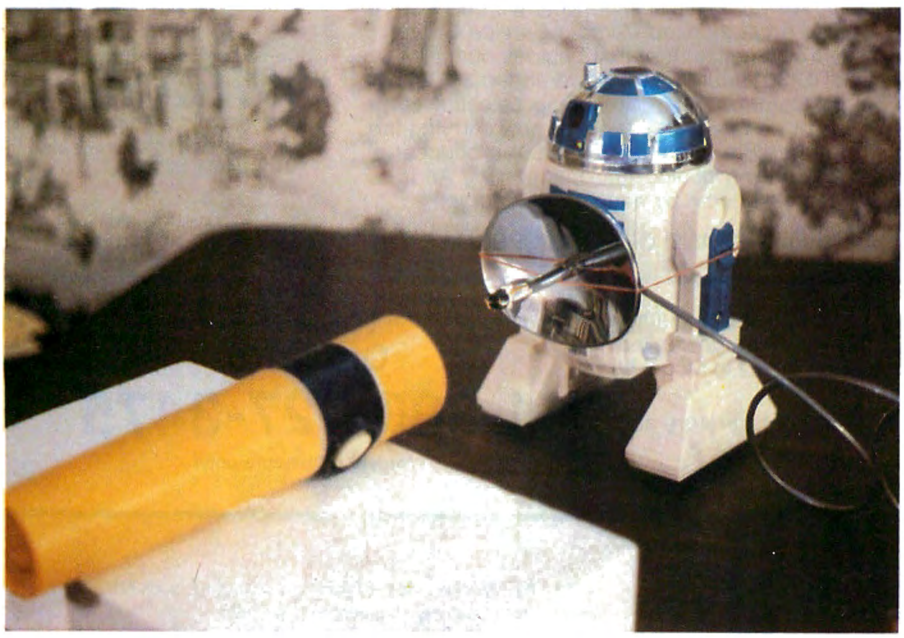


Photo 1. Robot arriving at a light source after a 180-degree scan and an eight-inch translation to reach it. This mode uses a Z-80 machine language subroutine embedded in Basic to convert the infrared intensity data to digital form.

Quantity	Part Designation	Current
1	7805 Voltage Regulator (5 Volts D.C.)	5 milliamps
1	7404 Hex Inverter	12 milliamps
1	7406 Hex Inverter/Driver (Open Collector)	30 milliamps
1	7475 Quad Latch (Level Sensitive)	32 milliamps
1	74367 Hex Bus Driver (Tri-State)	65 milliamps
1	555 Timer	3 milliamps
2	Archer SPDT Sensitive Mini Relay #275-004	12 milliamps
1	Infrared Photo Detector #276-142	25 milliamps
1	2N2222-2N2907 Transistor pair	

1,1 meg ohm trim pot	2,sub-mini phone jacks
1,50 kilo ohm trim pot	2,16 pin low profile sockets
2,.01 micro farad capacitors	3,14 pin low profile sockets
1,1 micro farad tantalum capacitor	1,8 pin low profile socket
1,1000 micro farad electrolytic	1 Archer printed circuit board

Table 1. Parts List for Interface Control Board

"With a hardware interface and the following software, R2D2 now responds to my voice as well as infrared light."

time. The cassette recorder acts as a microphone and puts voltage on the cassette IN port. An X = INP(255) Level II Basic command reads this value and sets X = 255, if a sound is present; or sets X = 127, if there is no sound.

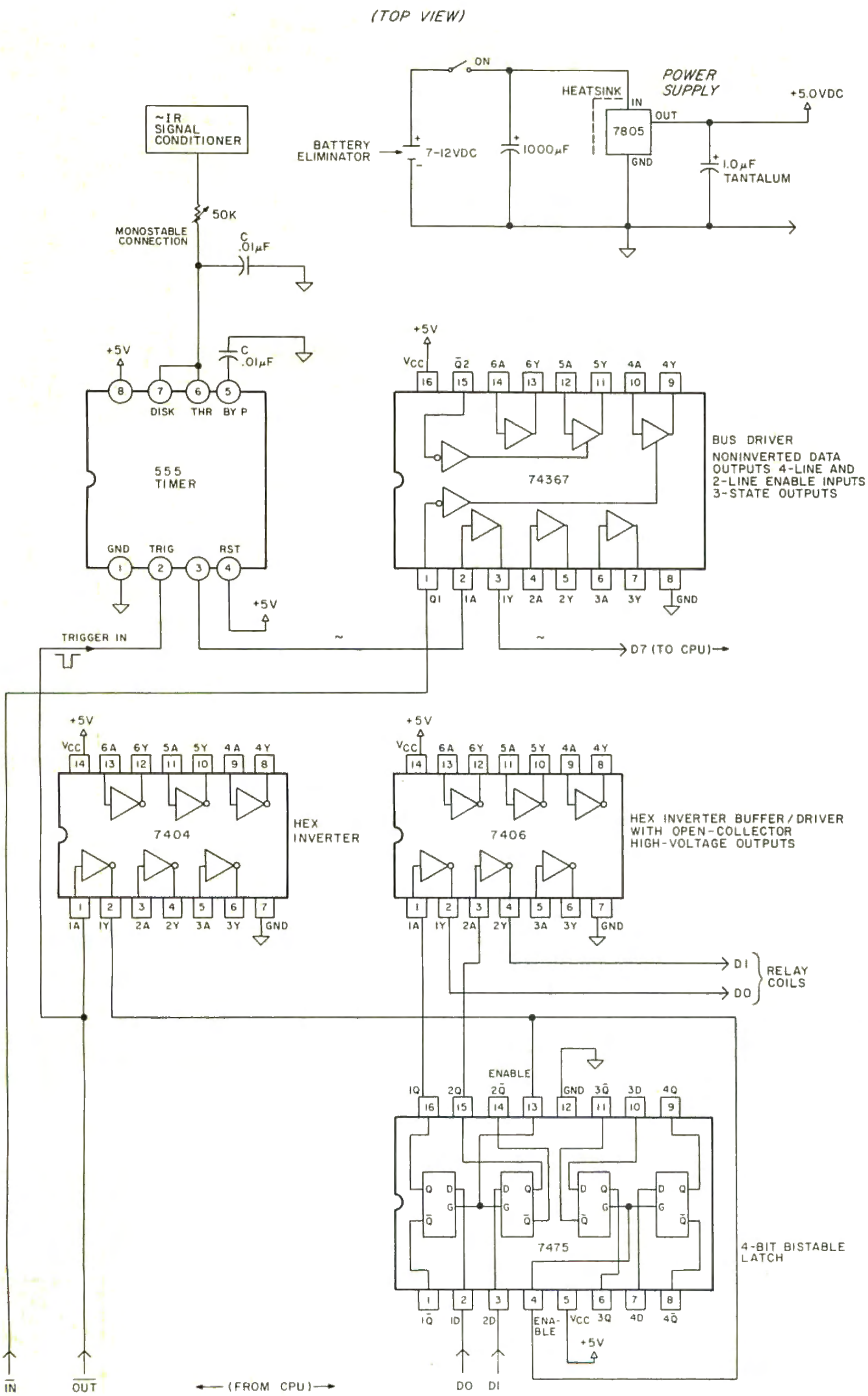


Fig. 1

Building the Interface Control Board

The interface control board connects to the TRS-80 port edge card in back of the keyboard by means of a soldertail 40-pin connector with .1 inch spacing.

I used a rainbow cable to help keep the bus signals straight when wiring the printed circuit board. I found an ohmmeter useful to check for continuity and incorrect wiring during assembly (see Photo 2).

A schematic of the interface board is shown in Fig. 1 (top view). Fig. 1 also shows the approximate placement of the integrated circuit chips on the printed circuit board. The entire circuit easily fits on an Archer epoxy-glass printed circuit board (#276-155).

The infrared signal conditioner is shown in Fig. 2. The analog to digital conversion is performed by the interface control board in Fig. 1 using a variable pulse duration from the 555 timer integrated circuit.

Fig. 3 gives the pin outs for the TRS-80 port edge card. Note that the IN and OUT port signals are also represented as negative (logical 0), true input or output designated by $\overline{\text{IN}}$ and $\overline{\text{OUT}}$ in Figs. 1 and 3. In addition to $\overline{\text{IN}}$ and $\overline{\text{OUT}}$, D0, D1, D7 and the signal ground GND (don't forget to connect it to the TRS-80 interface connected ground) must be wired to complete the connection of the interface board to the CPU. The address lines don't need to be wired for this application because no address decoding or port decoding is used in the generation of strobe signals. If you want, this could be incorporated at the cost

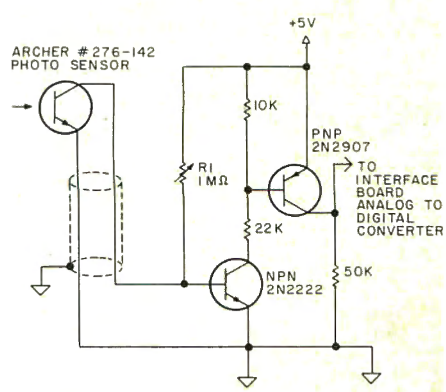


Fig. 2. Optical signal conditioner converts light input into a voltage proportional output.

"I decided to interface Kenner's R2D2 robot to my TRS-80. It is well-built and inexpensive."

of a few more chips.

Robot Motion Control

Controlling the robot's movement uses only a portion of the circuitry shown in Fig. 1. The integrated circuit chips 7404, 7406 and 7475 control the robot's translation and rotation via the relays D0 and D1. They key R2D2's radio control device.

Integrated circuit chip 7404 is a hex inverter that inverts the OUT signal from the CPU to produce an OUT signal to strobe or enable the 7475 latch. The OUT signal is applied to pins 4 and 13 of the four-bit bistable latch 7475. This latches data bits D0 and D1 (from the expansion port edge card) with the output being held on pins 16 and 15 of the 7475 chip.

The latched D0 and D1 output signals then drive the open-collector hex inverter 7406, which in turn drives the D0 and D1 relays. The following are the outputs on pins 2

P/N	SIGNAL NAME	DESCRIPTION
1	RAS*	Row Address Strobe Output for 16-Pin Dynamic Rams
2	SYSRES*	System Reset Output, Low During Power Up Initialize or Reset Depressed
3	CAS*	Column Address Strobe Output for 16-Pin Dynamic Rams
4	A1Ø	Address Output
5	A12	Address Output
6	A13	Address Output
7	A15	Address Output
8	GND	Signal Ground
9	A11	Address Output
10	A14	Address Output
11	A8	Address Output
12	OUT*	Peripheral Write Strobe Output
13	WR*	Memory Write Strobe Output
14	INTAK*	Interrupt Acknowledge Output
15	RD*	Memory Read Strobe Output
16	MUX	Multiplexor Control Output for 16-Pin Dynamic Rams
17	A9	Address Output
18	D4	Bidirectional Data Bus
19	IN*	Peripheral Read Strobe Output
20	D7	Bidirectional Data Bus
21	INT*	Interrupt Input (Maskable)
22	D1	Bidirectional Data Bus
23	TEST*	A Logic "Ø" on TEST* Input Tri-States AØ-A15, DØ-D7, WR*, RD*, IN*, OUT*, RAS*, CAS*, MUX*
24	D6	Bidirectional Data Bus
25	AØ	Address Output
26	D3	Bidirectional Data Bus
27	A1	Address Output
28	D5	Bidirectional Data Bus
29	GND	Signal Ground
30	DØ	Bidirectional Data Bus
31	A4	Address Bus
32	D2	Bidirectional Data Bus
33	WAIT*	Processor Wait Input, to Allow for Slow Memory
34	A3	Address Output
35	A5	Address Output
36	A7	Address Output
37	GND	Signal Ground
38	A6	Address Output
39	+5V	5 Volt Output (Limited Current)
40	A2	Address Output

NOTE: *means Negative (Logical "Ø") True Input or Output

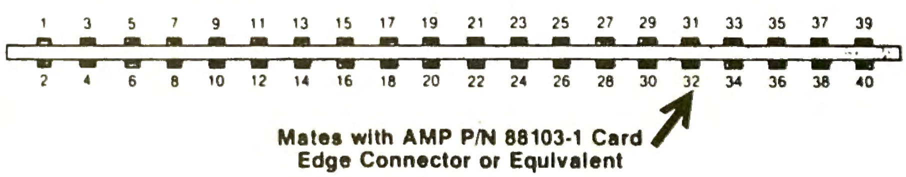


Fig. 3

and 4 of the 7406 integrated circuit:

	ground if D0 = 1 and OUT = 1	: Relay D0 = ON
	ground if D0 = 1 and OUT = 1	: Relay D0 = ON
Pin 2 of 7406 =	open circuit if D0 = 0 and OUT = 1	: Relay D0 = OFF
	open circuit if OUT = 0	: Relay D0 = OFF
	ground if D1 = 1 and OUT = 1	: Relay D1 = ON
Pin 4 of 7406 =	open circuit if D1 = 0 and OUT = 1	: Relay D1 = OFF
	open circuit if OUT = 0	: Relay D1 = OFF

Pins 2 and 4 of IC 7406 are connected to the ground side of the D0 and D1 relays respectively (the other side of the coil is energized with +5 volts D.C.). Thus, if D0 = 1 and OUT = 1, then relay D0 will close; and, if Df1 = 1 and OUT = 1, then D1 relay will close. Note that no port or address

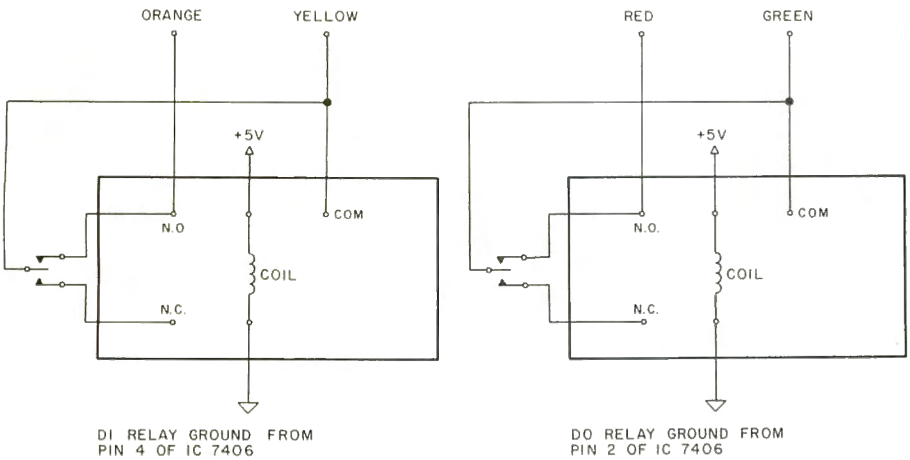


Fig. 4. Robot control relay connection schematic shows color coded relay outputs: D0 = 1 and D1 = 0 command translation; D0 = 1 and D1 = 1 command head rotation; Do = 0 and D0 = 0 give no motion.

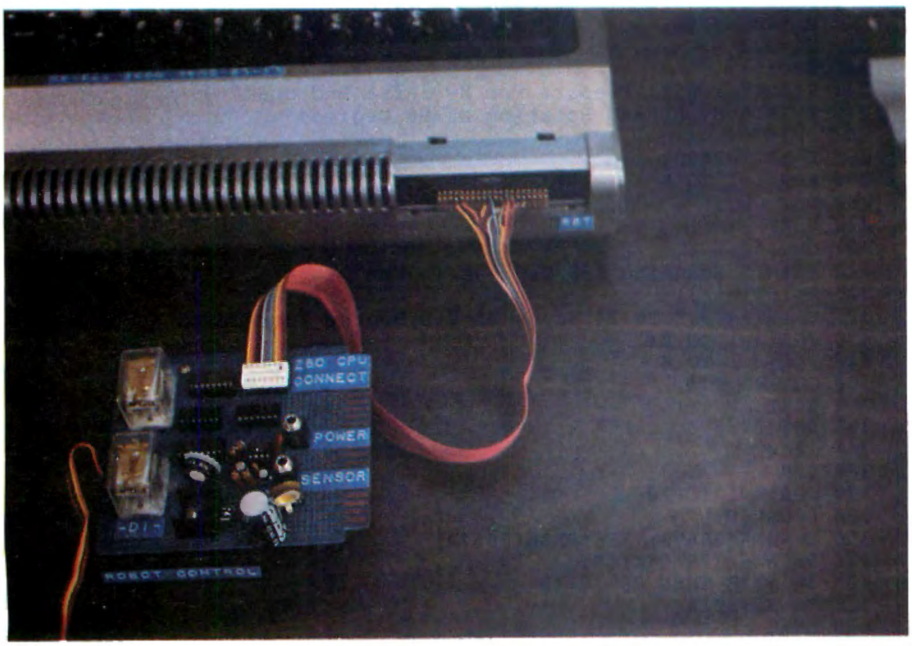


Photo 2. Interface control board connects to the TRS-80 by means of a 40-pin edge connector. Note the 14-pin DIP connector at the other end of the rainbow cable.

Program Listing 1

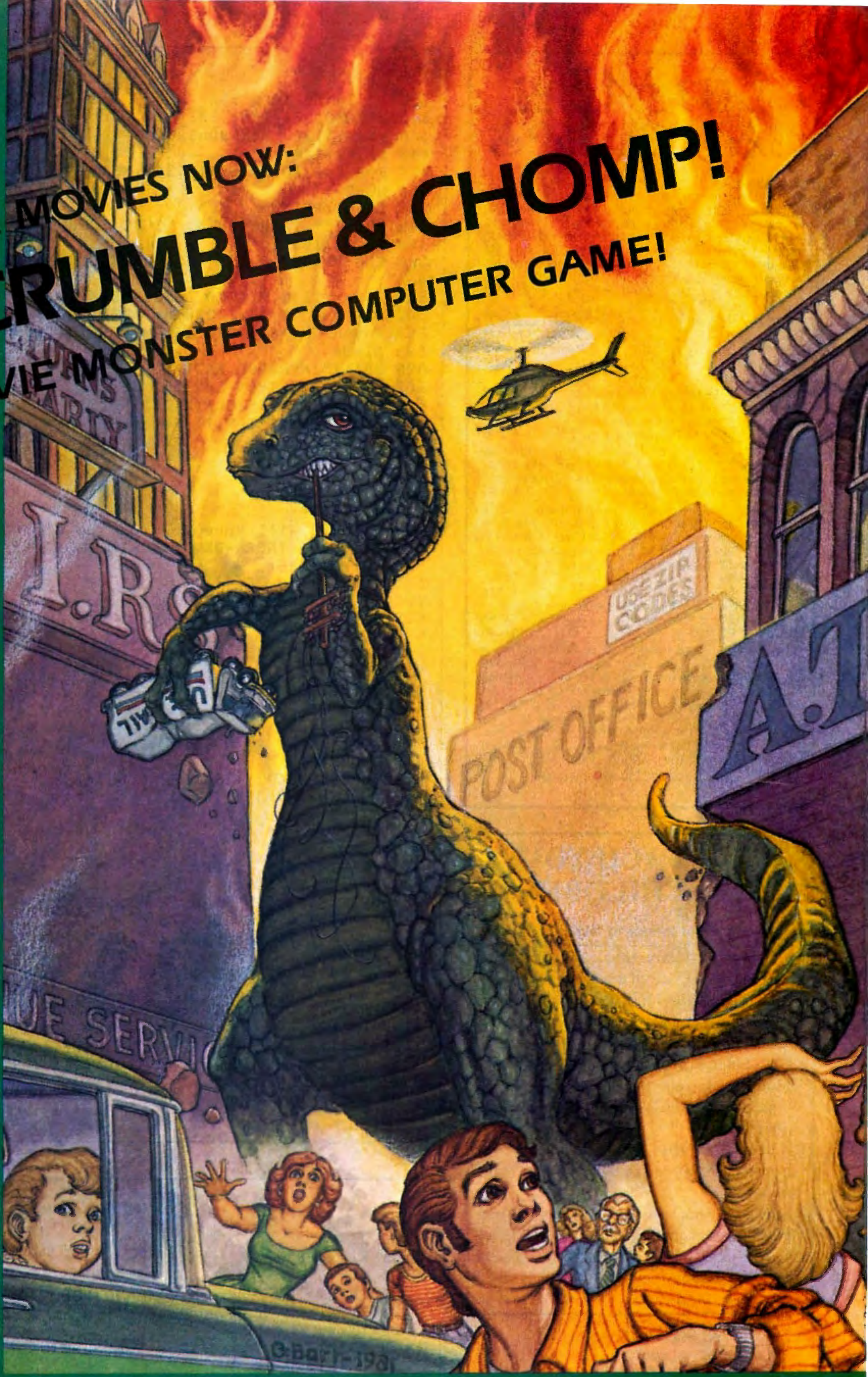
```
1 E$="SQR.SQR&FREDEFSNGSQR>AUTO( 79!85 FRE!85ENDANDCLOSEERR="
2 I=VARPTR(E$)
3 POKE 16526,PEEK(I+1):POKE 16527,PEEK(I+2)
10 M$="+?MD> )OR!3A- .<)OR!3A- .MKD$LPRINTOR!3AINKEY$"
20 I=VARPTR(M$)
30 POKE 16526,PEEK(I+1):POKE 16527.PEEK(I+2)
35 A$="QR+?GONOTXVARPTRINPNOT?MKS$C 0 FIXINKEY$ERTER"
38 'LINE 1 CONTAINS THE LOWERCASE IMPLEMENTATION MACHINE
39 'LANGUAGE STRING DENOTED BY E$
40 'LINE 10 CONTAINS THE MACHINE LANGUAGE MUSIC PROGRAM
50 'DENOTED BY M$
52 'LINE 35 CONTAINS THE INFRA-RED ANALOG TO DIGITAL
54 'CONVERTER - - MACHINE  LANGUAGE PROGRAM  IN A$
55 'THE FOLLOWING INSERTS THE MACHINE CODE INTO STRINGS E$,M$,A$

56 '* AFTER THE PROGRAM IS RUN LINES 60 TO 85 CAN BE DELETED *
60 I1=VARPTR(E$)
61 I2=VARPTR(M$)
62 I3=VARPTR(A$)
63 B1=PEEK(I1+1)+PEEK(I1+2)*256
64 B2=PEEK(I2+1)+PEEK(I2+2)*256
65 B3=PEEK(I3+1)+PEEK(I3+2)*256
66 DIM E(30),M(30),A(30)
67 FOR J=0 TO 30
68 READ E(J):POKE(B1+J),E(J):PRINT PEEK(B1+J);
69 NEXT
70 DATA 221,110,3,221,102,4,218,154,4,221
71 DATA 126,5,183,40,1,119,121,254,32,218
72 DATA 6,5,254,128,210,166,4,195,125,4,32
73 FOR J=0 TO 27
74 READ M(J):POKE(B2+J),M(J):PRINT PEEK(B2+J);
75 NEXT
76 DATA 205,127,10,77,68,62,1,105,211,255
77 DATA 45,32,253,60,105,211,255,45,32,253
78 DATA 13,16,238,175,211,255,201,32
79 FOR J=0 TO 27
80 READ A(J):POKE(B3+J),A(J):PRINT PEEK(B3+J);
81 NEXT
82 DATA 211,16,205,127,10,71,79,3,203,88
83 DATA 192,219,16,203,127,237,67,1,112,32
84 DATA 242,201,69,82,84,69,82,32
85 'END OF MACHINE CODE INSERTION ROUTINES
90 POKE 16526,PEEK(I1+1):POKE 16527,PEEK(I1+2)
91 POKE 16414,241:POKE 16415,66
92 POKE 16526,PEEK(I2+1):POKE 16527,PEEK(I2+2)
135 CLS
140 PRINT"             ** INFRA-RED ROBOTIC VISION DEMONSTRATION
**"
141 PRINT,"        by Don Mc Allister"
150 FOR I=2060 TO 2120:OUT00,0:L=USR(I):OUT00,0:NEXT
190 X=VARPTR(A$)
191 POKE 16526,PEEK(X+1):POKE 16527,PEEK(X+2)
200 K1=80
210 K2=2.3
215 IR=35
220 PRINT" I Will Prepare For My Infra-Red Scan"
230 PRINT" By A Head Rotation of 90 Degrees"
240 PRINT
260 B=INT(K2*90)
270 FOR I=1 TO B
280 OUT 00,3
290 NEXT
300 OUT 00,0
310 PRINT" Scanning 180 Degrees To The Right"
320 B=INT(K1/3)
325 PRINT@448," In 24 Scans Of 7.5 Degree Increments Each"
330 FOR I=1 TO 24
340 FOR J=1 TO B
350 OUT 00,1
360 NEXT
361 OUT 00,0
365 S0=0
366 C=0
370 FOR K=1 TO 10
380 L=USR(0)
390 S=PEEK(28673)+256*PEEK(28674)
400 PRINT@(379+5*K),S;
405 IF S>IR THEN S0=S0+IR
410 S0=S0+S/10
420 NEXT
422 PRINT@320," "
424 S0=INT(S0)
```

Program continues

FIRST: MONSTER MOVIES NOW:

CRUSH, CRUMBLE & CHOMP!

THE GREAT MOVIE MONSTER COMPUTER GAME!



And guess who stars as the movie monster. You! As any of six different monsters. More if you have the disk version.

You can terrorize and destroy four of the world's largest and most densely populated cities in over 100 possible scenarios. From Tokyo to the Golden Gate, you are the deadliest creature in the air, on the land, or in the sea.

You can be the deadly amphibian who simultaneously smashes street cars, lunches on helpless humans and radiates a ray of death.

If you were a giant winged creature, think of the aerial attacks you could make on the terrified but tasty tidbits beneath you.

But as in all the best monster movies, you're up against everything the human race can throw at you—even nuclear warheads and a strange concoction developed by a team of mad scientists.

For only $29.95 you get 6 stupendous monsters, each with its own monstrous summary card, 4 teeming metropoli displayed in graphic detail on your computer display and mapped in the accompanying 48-page illustrated book, the awesome sounds of monsterly mayhem, and spine-tingling, real-time, edge-of-your-seat excitement.

GET CRUSH, CRUMBLE & CHOMP
now at your local dealer for your APPLE, ATARI, or TRS-80 . . **before it's too late.**

EPYX
COMPUTER GAMES THINKERS PLAY

48

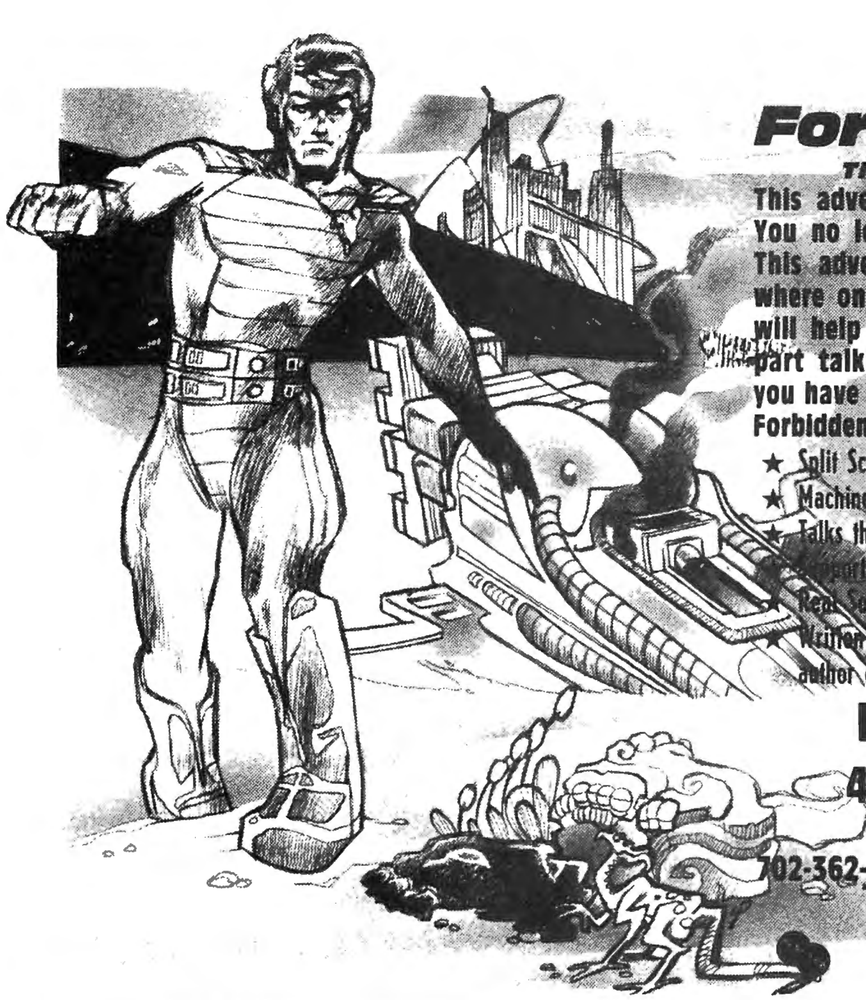
Program Listing 1 Continued

```
425 IF SØ>IR THEN 5ØØ
44Ø PRINT@(5Ø4+I*8),SØ;
445 IF SØ>126 THEN SØ=126
446 PRINT@64Ø," "
45Ø FOR X=1 TO SØ:SET(X,34):NEXT
46Ø NEXT
465 PRINT:PRINT
47Ø PRINT"* THERE IS NO LIGHT SOURCE -- So I Will Turn Off *"
48Ø GOTO 11Ø5
5ØØ CLS:PRINT"          * I HAVE SEEN A LIGHT -- I Will Investigat
e *"
5Ø1 I=VARPTR(M$):POKE 16526,PEEK(I+1):POKE 16527,PEEK(I+2)
5Ø2 FOR P=1 TO 1ØØ:OUTØ,Ø:L=USR(2Ø6Ø):OUTØ,Ø:NEXT
5Ø3 M=VARPTR(A$):POKE 16526,PEEK(M+1):POKE 16527,PEEK(M+2)
5Ø5 S1=SØ
5Ø6 PRINT"INFRARED RADIANCE =";SØ
5Ø7 IF SØ>126 THEN SØ=126
5Ø8 FOR X=1 TO SØ:SET(X,28):NEXT
51Ø PRINT:PRINT"I Will Rotate My Head Towards The Light"
6ØØ B=INT(K2*27Ø)
61Ø FOR I=1 TO B
62Ø OUT ØØ,3
63Ø NEXT
64Ø OUT ØØ,Ø
66Ø PRINT"I Will Move Towards The Light"
665 PRINT,"* INFRA-RED TRACKING MODE *"
67Ø B=INT(K1*6)
68Ø FOR I=1 TO 1Ø
682 IF I>1 THEN B=INT(K1*3)
685 SØ=Ø
69Ø FOR J=1 TO B
7ØØ OUT ØØ,1
71Ø NEXT
712 GOSUB 73Ø
715 NEXT
716 GOTO 1ØØØ
73Ø FOR K=1 TO 1Ø
74Ø L=USR(Ø)
75Ø S=PEEK(28673)+256*PEEK(28674)
76Ø PRINT@(379+5*K),S;
77Ø SØ=SØ+S
78Ø NEXT
781 PRINT
785 PRINT@32Ø," ",:PRINT
79Ø SØ=INT(SØ/1Ø)
8ØØ PRINT@512,"INFRARED RADIANCE =";SØ
81Ø IF SØ>126 THEN SØ=126
815 FOR X=1 TO SØ:SET(X,28):NEXT
85Ø PRINT
9ØØ IF SØ<IR THEN GOTO 1ØØØ
95Ø RETURN
1ØØØ PRINT"      * I Have Arrived At The Light Source *"
11ØØ PRINT"           * Turning Off Sensors *
11Ø5 I=VARPTR(M$):POKE  16526,PEEK(I+1):POKE 16527,PEEK(I+2)
11Ø6 FOR I=1 TO 1Ø STEP 1/4:OUTØØ,Ø:L=USR(3E3-ABS(15Ø-3*I)):OUTØ
Ø,Ø:NEXT
12ØØ END
```

Program Listing 2

```
1Ø K1=7Ø
2Ø K2=1.86
25 CLS
26 PRINT@15,"* VOICE COMMAND ROBOT DEMONSTRATION *"
27 XØ=X1=X2=X3=Ø
28 XT=Ø
29 PRINT@128,"* 1Ø-2Ø = TRANSLATION *  * 2Ø-3Ø = ROTATION *"
3Ø PRINT@192," "
31 PRINT@384," "
4Ø K=Ø
5Ø FOR I=1 TO 1ØØØ
51 XØ=Ø
55 OUT 255,ØØ
6Ø X=INP(255)
7Ø IF X=255 THEN K=K+1
71 PRINT@(256+K),K;
8Ø IF XT=1 AND K>1Ø THEN GOTO 1Ø6
86 IF X=127 AND K>1 THEN XØ=1
87 X3=X2:X2=X1:X1=XØ
```

Program continues



Forbidden Planet

The First Talking Adventure

This adventure talks through the cassette port. You no longer need to have a voice synthesizer. This adventure takes you to a desolate planet where only your skill and your talking computer will help you survive. This is Part 1 of a multi-part talking adventure. Like no other adventure you have ever played. The first five people to solve Forbidden Planet will get the next adventure FREE.

★ Split Screen
★ Machine Language
★ Talks through Cassette Port
★ Supports Lower Case
★ Real Sound Effects
★ Written by William Demas
author of "Scott Adams Adventure #12"

Price $39⁹⁵

48K Disk Model I & III

Dealers Inquiries Invited

702-362-1457 Forum 80 702-362-3609

Fantastic Software ✓225
3110 Polaris #3
Las Vegas, Nevada 89102

VISA

TRS-80 OWNERS REJOICE!

Our professional quality software development tools let you program your **TRS-80** with more ease and power than you ever imagined possible. Programming will become a pleasure when you use our full screen editors. These editors are not just patched up word processors. They have been specially designed to enhance the BASIC and ASSEMBLER programming languages. Our assemblers provide full screen option menus to relieve you of the burden of remembering command names. Combine user oriented features like this with the finest and most extensive documentation in the industry and you will see why programming a TRS-80 is now better than ever.

T-ZAL: TAPE BASED ASSEMBLER: Assemble to memory or tape. Create relocatable SYSTEM tapes! Includes relocating LOADER pgm. 8 character symbols with XREF. Upgrades to disk when you do!
Mod 3 Cat # 1250-20 $ 49.95

FULL SCREEN PROGRAM TEXT EDITORS: All CAU full screen editors support four way cursor motion and scrolling with user selectable auto repeat and scroll speeds. Totally visual character insert, delete, and change. Block move, copy, delete. Global find and change. Editor for BASIC also provides automatic line renumbering and 26 user-definable macro keys.

For Tape and Disk BASIC:
Mod 1 Cat # 1010-20 $24.95
Mod 3 Cat # 1210-20 $29.95

For Mod 1 EDTASM source files:
Tape Cat # 1010-30 $24.95
Disk Cat # 1010-31 $34.95
(Disk version shipped on tape.)

M-ZAL: DISK EDITOR/ASSEMBLER: Modular Assembler Language development system includes full screen text editor, multi-pass assembler, and object module linker. Allows 8 character symbols and source file chaining/nesting with *INCLUDE cmd. Full support for relocation and global symbols. Symbol table with cross reference. Req. 32K Disk system.
Mod 1 Cat # 1050-10 $149.00
Mod 3 Cat # 1250-10 $149.00

XBUG: DEBUGGING TOOL & MONITOR: Display/Modify memory and regs. Set up to 10 invisible and transparent breakpoints. Multi-speed single step. Load/Copy/Create SYSTEM tapes. Uses only 2.5K and can be dynamically relocated to any address.
Mod 1 Cat # 1020-10 $19.95
Mod 3 Cat # 1220-10 $19.95

Note: these products are not available for level 1 machines.

When ordering specify Model, RAM size, # of disks.

Send check, money order, or MC/VISA numbers and expiration date to:
Computer Applications Unlimited
P.O. Box 214, Dept. EEW
Rye, New York 10580

N.Y. State residents add applicable sales tax.
Allow 4-6 weeks for delivery.
Dealer Inquiries Invited.

CAU™ **COMPUTER APPLICATIONS UNLIMITED**™ ✓107
A Div. of CAU, Inc.
Quality•Reliability•Service
*TRS-80 is a TM of Tandy Corp.

Program Listing 2 Continued

```
88 XT=X0*X1*X2*X3
89 IF XT=1 AND K<10 THEN GOTO 25
95 SET(2*K,22):SET(2*K+1,22)
96 IF K>30 THEN GOTO 25
100 NEXT
105 IF K=0 GOTO 30
106 CLS:K=INT(K/10)
110 IF K=1 PRINT@0,"* YOU HAVE SELECTED MY TRANSLATION MODE *
120 IF K=2 PRINT@0,"* YOU HAVE SELECTED MY HEAD ROTATION MODE *"
121 PRINT@192," "
122 PRINT@384," "
130 IF K=1 THEN A=1
140 IF K=2 THEN A=3
150 K=0
151 X0=X1=X2=X3=0
152 XT=0
155 FOR I=1 TO 10000
156 OUT 255,00
157 X0=0
160 X=INP(255)
170 IF X=255 THEN K=K+1
171 PRINT@(256+K),K;
```

Program continues

decoding is used in controlling the robot's motion (an OUT command to any port causes the relays to operate). This is all right if no other peripherals using data bits D0 and D1 are being operated at the same time.

A cassette recorder can also be operated at the same time the robot is being controlled by using OUT 0 byte commands for motion control. Since the recorder is port FF = 255 (decimal) it will be unaffected by OUT 0 commands and the robot is unaffected by OUT FF,04, which turns the recorder on. Note that up to four relays can be controlled by the circuit in Fig. 1 without adding more chips.

The D0 and D1 grounds to the relay coils in Fig. 1 are connected to the ground side of the coils in the relays shown in Fig. 4. The plus side of the coils are connected to the + 5 Volt D.C. supply. Photo 3 shows how the

"After you are certain everything is working as it should, . . . turn on the power. The robot should immediately start rotating his head."

color coded relay output switch wires are connected to the R2D2's radio control unit.

N.O.	NORMALLY OPEN
N.C.	NORMALLY CLOSED
VOLTAGE	6-9VDC
RESISTANCE	500Ω
CURRENT	12MA
CONTACT RATING	1AMP

Once you have assembled the robot interface control board, and its connection to the radio control unit and microcomputer, plug in the power supply to an unregulated 7-12 volt D.C. source (a 300 milliamp plug-in battery eliminator does nicely). Double check the circuit for proper voltage levels and continuity before connecting it to the computer.

After you are certain everything is working as it should, connect the interface to the computer bus and turn on the power; the robot should immediately start rotating his head. To stop the motion hit Clear. The mo-

Program Listing 2 Continued

```
172 IF X=127 AND K>1 THEN X0=1
173 X3=X2:X2=X1:X1=X0
174 XT=X0*X1*X2*X3
175 IF XT=1 AND K<10 THEN GOTO 25
176 IF XT=1 AND K>10 THEN GOTO 209
190 IF K>60 THEN GOTO 209
195 SET(2*K,22):SET(2*K+1,22)
200 NEXT
205 IF K=0 THEN GOTO 25
209 K=INT(K/10)
210 IF A=1 PRINT@64,"* FOR A TRANSLATION OF";K;"INCHES *"
220 IF A=3 PRINT@64,"* FOR A HEAD ROTATION OF";K*90;"DEGREES *"
230 IF A=1 THEN B=INT(K1*K)
240 IF A=3 THEN B=INT(K2*90*K)
250 OUT 00,0
260 FOR J=1 TO B
270 OUT 00,A
280 NEXT
290 OUT 00,0
295 CLS
300 GOTO 10
400 END
```

"Position the robot's head so he is looking straight ahead; The robot will move about the table top."

tion is stopped when the D0 and D1 relays are latched to 00. Now you are ready to load the motion control in Program Listing 1.

After the program is loaded, position the robot's head so that he is looking straight ahead (red LED eye in front). Then, run the Basic program in Listing 1, and the robot will go through a preprogrammed sequence moving about the table top in a square. Notice that the program starts and stops the cassette recorder before each motion. This allows you to record a verbal description of the robot's maneuvers on cassette for playback on subsequent runs. If you don't want to make use of this feature delete lines 270–300, 710–730 and 920–940 from Listing 1.

If the motion is not as desired, try modifying the constants K1 and K2 in Listing 1. They control the magnitudes of the robot's moves. For example, if you input a 360-degree head rotation and the robot's head only rotates 340 degrees, increase K2 by 15 percent.

```
43B5:  D3 10         OUT (10H),A       ; trigger A to D converter
43B7:  CD 7F 0A      CALL 0A7FH        ; initialize converter
43BA:  44            LD B,H            ; with most significant byte
43BB:  4D            LD C,L            ; least significant byte
43BC:  03            INC BC            ; increment counter
43BD:  CB 58         BIT 3,B           ; limit max converter value
43BF:  C0            RET NZ            ; return if value exceeded
43C0:  DB 10         IN A,(10H)        ; input sensor data D7
43C2:  CB 7F         BIT 7,A           ; check if D7=1
43C4:  ED 43 01 70   LD (7001H),BC     ; store count in 7001H
43C8:  20 F2         JR NZ,43BC        ; keep counting if D7=1
43CA:  C9            RET               ; stop count when D7=0 and
                                         return to BASIC
```

Program Listing 3

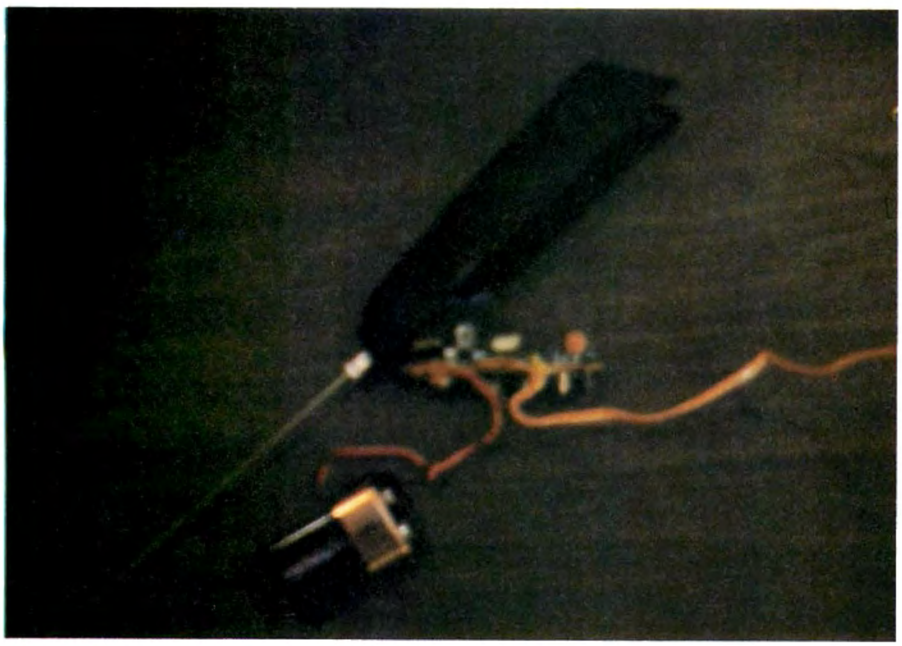


Photo 3. The robot interface control board relays D0 and D1 connect to the radio control unit as indicated here.

✓See List of Advertisers on page 354

"It's fun to whistle command your robot around the floor. You can also construct a robot obstacle course to test your skill."

```
4362:  CD 7F 0A     CALL 0A7FH       ; pass argument in USR( )
4365:  4D           LD C,L           ; frequency
4366:  44           LD B,H           ; duration
4367:  3E 01        LD A,01H         ; for square wave start
4369:  69           LD L,C           ; get frequency
436A:  D3 FF        OUT (FFH),A      ; begin positive half cycle
436C:  2D           DEC L            ; count to end of positive
436D:  20 FD        JR NZ,436CH      ; half cycle of square wave
436F:  3C           INC A            ; for next half cycle
4370:  69           LD L,C           ; get frequency
4371:  D3 FF        OUT (FFH),A      ; begin negative half cycle
4373:  2D           DEC L            ; count to end of cycle
4374:  20 FD        JR NZ,4373       ; continue till done
4376:  0D           DEC C            ; increase pitch
4377:  10 EE        DJNZ,4367        ; if duration + more waves
4379:  AF           XOR A            ; end of square waves
437A:  D3 FF        OUT (FFH),A      ; from M$
437C:  C9           RET              ; return to BASIC program
```

Program Listing 4

```
42F1:  DD 6E 03     LD L,(IX+03H)    ; address of cursor LSB
42F4:  DD 66 04     LD H,(IX+04H)    ; address of cursor MSB
42F7:  DA 9A 04     JP C,049AH       ; jump if VIDEO read
42FA:  DD 7E 05     LD A,(IX+05H)    ; check if cursor is on
42FD:  B7           OR A             ; jump to control check
42FE:  28 01        JR Z,4301H       ; if not on
4300:  77           LD (HL),A        ; blank if cursor on
4301:  79           LD A,C           ; control character check
4302:  FE 20        CP 20H           ; control character?
4304:  DA 06 05     JP C,0506H       ; jump if control character
4307:  FE 80        CP 80H           ; graphic character?
4309:  D2 A6 04     JP NC,04A6H      ; jump if graphic character
430C:  C3 7D 04     JP 047DH         ; patch back to video display
                                       driver in ROM
```

Program Listing 5

Program Listing 6

```
0 REM ROBOT CONTROL DEMONSTRATION
1 CLS:PRINT"MY NAME IS R2D2 AND I WILL NOW EXECUTE A SEQUENCE"
2 PRINT"OF PRE-PROGRAMMED MOTIONS"
3 PRINT
4 K1=81.
5 K2=1.96
6 OUT 255,4
7 FOR W=1 TO 6000:NEXT
8 OUT 255,0
10 DIM A(40),B(40),C$(40)
```

Program continues

Robot Voice Control

Sounds of various durations can be used to designate the type and magnitude of the robot's motion. It is fun to whistle command your robot around the table top or floor. You can also construct a robot obstacle course to test your skill.

To use the voice control mode load the program in Program Listing 2 into your computer. Press play and record on the recorder simultaneously (you'll also have to insert a blank tape) and remove the black plug from the microphone jack. By whistling into the cassette microphone for various durations the robot can be maneuvered in any desired pattern when Listing 2 is run.

The program can also be modified to combine computer controlled motions with verbal commands.

Infrared Sensor Light Source

The infrared sensor (Photo 1) actually responds to visible light as well as infrared light (in the 1 micro-meter range). Infrared LED's emit light over a narrow band of wavelengths in the infrared spectrum as shown in Fig. 5 (a tungsten filament lamp is a good generator of infrared light). For this reason ordinary flashlights or penlights can be used as infrared light sources (with or without filters). A flashlight also has the advantage of producing a beam of light which can be seen over great distances.

Infrared data is converted from analog to digital impulses and input to the CPU through ICs 555 and 74367 in Fig. 1. The actual analog to digital conversion is performed by the machine language subroutine in Listing 3, which can be called from Level II Basic using the USR command.

Machine code can be either POKEd into the string locations, or can be entered directly into string locations using a monitor program such as RSM or T-Bug. It is helpful to enter first the Basic program in Listing 6, using arbitrary keyboard charac-

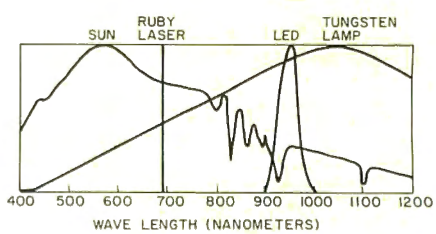


Fig. 5. The spectra of various light sources used in robotics experiments.

ters to fill out the strings. After this is done, the machine code in Listings 3, 4 and 5 can be inserted into the proper string from the vantage point of a high memory monitor program. Two restrictions in doing this are that the program must be less than 256 bytes, and must contain no zero byte op-codes. A zero byte causes the Basic interpreter to think it has reached the end of the string definition. Careful coding can get around these restrictions in most instances.

When an infrared source above a software selectable threshold is seen by the robot during a scan, a musical note is sounded through the cassette auxiliary cable. You can connect this cable to the phonojack input of a stereo amplifier, so that the tones generated when the robot sees a light are heard in high-fidelity. The musical tone is formed by means of a Z-80 machine language subroutine, which is also embedded in a string. This program is given in Listing 4.

Listing 5 gives the Z-80 code for the display of lowercase as well as uppercase letters on the video monitor.

Finally, Listing 6 is a Level II Basic program that uses the machine language subroutines in Listings 3, 4 and 5, and converts your robot into an infrared scanning, detecting and tracking system.

After loading the program, plug the sensor coaxial cable into the interface board. As a first test of the sensor, turn the robot off and set the IR threshold to a large number (IR = 1.E10) in line 350 of Listing 6. Run the program, and observe the infrared intensity displayed. The numbers should be around 10, if no light is shining on the sensor. With a light shining the number will go up to several hundred.

Adjust the two sensitivity trim pots on the interface board until this happens. After the infrared signal conditioner and analog to digital converter are working properly, set IR = 20, turn the robot on, and run the program. The robot will begin a series of scans in 7.5-degree increments until a light source is seen. At this time R2D2 sounds a musical note through the auxiliary cable, and it begins to track the light. R2D2 now moves toward the light. When he arrives at the light source, the image of the light on the detector becomes diffused, and the robot stops (once again to a musical accompaniment).

Incidentally, if you want to mount more sensors such as touch sensors, or sonar, the analog to digital converter design allows up to six analog signals to be read and converted simultaneously. ■

Program Listing 6 Continued

```
20 FOR I=1 TO 16
30 READ A(I),B(I)
40 NEXT I
100 DATA 1,4
101 DATA 3,90
102 DATA 1,2.5
103 DATA 3,270
104 DATA 1,4
105 DATA 3,90
106 DATA 1,2.5
107 DATA 3,270
108 DATA 1,4
109 DATA 3,90
110 DATA 1,2.5
111 DATA 3,270
112 DATA 1,4
113 DATA 3,90
114 DATA 1,2.5
115 DATA 3,270
116 DATA 1,4
120 DATA 3,270
200 FOR I=1 TO 16
210 A=A(I):B=B(I)
220 OUT 00,0
225 IF A=1 PRINT"I = ";I;" TRANSLATE ";B;" INCHES"
226 IF A=3 PRINT"I = ";I;" ROTATE HEAD ";B;" DEGREES"
227 IF A=1 THEN B=INT(K1*B)
230 IF A=3 THEN B=INT(K2*B)
231 OUT 255,4
232 FOR W=1 TO 2000:NEXT
233 OUT 255,0
235 FOR J=1 TO B
240 OUT 00,A
250 NEXT
260 OUT 00,0
270 NEXT
280 PRINT:PRINT"I AM NOW FINISHED WITH THE MOTION SEQUENCE"
285 PRINT
290 PRINT"* WHAT IS YOUR NEXT COMMAND ? *"
294 OUT 255,4
295 FOR W=1 TO 4000:NEXT
296 OUT 255,0
```


"I see Instant Software in your future..."
". . . . It's becoming very clear now . . . Your microcomputing life is going to be very exciting . . . Money! I see much money for you. Perhaps it is the money you will gain when Instant Software's new business applications guide your financial endeavors. I see travel . . . you will journey to distant worlds in distant times. You will build kingdoms from deserts and armies from slaves. Never will you be bored, adventurous one! Your programming burdens will lighten with new utilities—new tools. I see color! Many bright colors! I see new packaging for all these treasures . . . everything new! And . . . yes, a tall, handsome stranger who will guide you to these wonders. It will be your Instant Software dealer—a wise one indeed. All awaits you—all is yours for the asking. Soon . . . very soon!"

Get a glimpse of the future today—at your nearest Instant Software dealer.

ALABAMA
THE COMPUTER SHOP, Gadsden
ALASKA
COMPUTER TALK, Anchorage
JUNEAU ELECTRONICS, Juneau
WAPATCO, Anchorage
ARIZONA
COMPUTER STORE, Phoenix
MESA ELECTRONICS, Mesa
MILLET'S ELECTRONICS, Mesa
PERSONAL COMPUTER PLACE, Mesa
SIMUTEK, Tucson
SOFTWARE STATION, Tempe
TOY BOX, Sierra Vista
ARKANSAS
MICRO COMPUTER SYSTEMS, Hot Springs
CALIFORNIA
ADVANCE RADIO (R/S DEALER), Grass Valley
ADVANCED COMPUTER PRODUCTS, Santa Ana
ALLTRONICS, San Jose
AMCO ELECTRONIC SUPPLY, Azusa
ASAP COMPUTERS, Signal Hill
BYTE INDUSTRIES, Hayward
BYTE SHOP, Mountain View
CAPITOL COMPUTER SYSTEMS, Sacramento
COAST ELECTRONICS, Morro Bay
COMPUSOUND, Orange
COMPUTER MART OF CALIFORNIA, INC., Diamond Bar
COMPUTER PLUS, Sunnyvale
COMPUTER STORE, San Leandro
COMPUTER WORLD, Westminster
DEROSEAR APPLIANCE & REPAIR, Weaverville
DIMENSIONAL SOFTWARE, San Diego
EXATRON, Santa Clara
GAME-O-RAMA, Santa Barbara
GRAY MATTER, San Francisco
HOBBY WORLD ELECTRONICS, Northridge
HUNTINGTON COMPUTING, Corcoran
MALIBU MICROCOMPUTING, Malibu
MICRO SPECIALTIES, Daily City
MN&T INDUSTRIES, Lompoc
OPAMP/TECHNICAL BOOKS, Los Angeles
OPPORTUNITIES FOR LEARNING, Chatsworth
PC COMPUTERS, El Cerrito
Q I COMPUTERS, INC., Lawndale
RAC PRODUCTS, San Jose
RADIO SHACK, El Cajon
RADIO SHACK, Palm Springs
RADIO SHACK, San Diego
R&V SOUND (R/S DEALER), Fortuna
SALINAS HOBBY CENTER, Salinas
SHAVER RADIO, San Jose
SILVER SPUR ELECTRONICS, Chino
SOFTWARE PLUS, El Toro
STACEY'S BOOKSTORE, San Francisco
STRAWFLOWER ELECTRONICS (R/S DEALER), Half Moon Bay
THE COMPUTER STORE, Santa Monica
THE FEDERATED GROUP, Commerce
THE SOFTWARE STORE, Huntington Beach
WABASH APPLE, El Toro
WARRIORS, WIZARDS & ROBOTS, Garden Grove
WENNER BUSINESS SYSTEMS, Los Altos
COLORADO
APPARAT, Denver
FISTEL'S MICRO ELECTRONICS, Denver
POOR RICHARD'S CALCULATORS, Fort Collins
SOFTWARE GOURMET, Denver
CONNECTICUT
AMERICAN BUSINESS COMPUTERS, Groton
BYTE ME COMPUTER SHOP, New London
COMPUTER LAB, New London
INSTRUCTIONAL SYSTEMS COMPUTERS, Manchester
TECHNOLOGY SYSTEMS, Bethel
THE COMPUTER STORE, Stamford
DELAWARE
MICRO PRODUCTS, Wilmington
OMNIFAX, Wilmington
DISTRICT OF COLUMBIA
THE PROGRAM STORE, Washington, D.C.
FLORIDA
ADVENTURE INTERNATIONAL, Casselberry
ALL SYSTEMS GO, Winter Garden
AMF MICROCOMPUTER CENTER, Tampa
COMPUTER CENTER, West Palm
COMPUTER JUNCTION, Fort Lauderdale
COMPUTER WORLDS, Clearwater
COMPUTERLAND, Jacksonville
COMPUTERLAND, Sarasota
COMPUTERLAND, Tampa
COMPUTERLAND, West Palm Beach
HEATHKIT ELECTRONIC CENTER, Hialeah
H&H HOBBY SALES, Sarasota
MICROCOMP LTD., Miami
MICROCOMPUTER SYSTEMS INC., Tampa
MINI CONCEPTS, Holly Hill
SAV-A-BYTE, Jacksonville
SOUND IDEAS, Gainesville
SOUTH EAST MICRO DATA, Orlando

GEORGIA
ATLANTA COMPUTER MART, Atlanta
BAILEY'S COMPUTER SHOP, Augusta
DELTA DATA DYNAMICS, Tucker
ENERGY LOGIC, Columbus
FLEMING DRUG CO., Wrens
HAWAII
HONOLULU ELECTRONICS, Honolulu
MILLS ELECTRONICS, Lahaina
RADIO SHACK ASSOC. STORE, Honolulu
IDAHO
DENNIS STONE ENTERPRISES, Fruitland
ELECTRONIC SPECIALTIES, Boise
R&L DATA SYSTEMS, Idaho Falls
ILLINOIS
ALPINE COMPUTER CENTER, Rockford
BELL ELECTRONICS, Edinboro
CHICAGO MAIN NEWSTAND, Evanston
COMPREHENSIVE MICRO SYSTEMS, Chicago
COMPUTER JUNCTION, Elmhurst
COMPUTER STORE, Rockford
COMPUTERLAND, Niles
CREATIVE PROGRAMMING, Charleston
GARCIA & ASSOCIATES, Chicago
ILLINOIS CUSTOM COMPUTERS, Harrisburg
MAIN STREET COMPUTER CO., Decatur
MIDWEST MICRO COMPUTERS, Lombard
THE PILOT HOUSE, Gladstone
WALLACE COMPUTERS, Peoria
INDIANA
ABC HOBBY, Evansville
DIGITAL TECHNOLOGY, Lafayette
FALL CREEEK ELECTRONICS, Pendleton
GAME HUT, Bloomington
GAME PRESERVE, Indianapolis
PROFESSIONAL MICROCOMPUTER SOFTWARE, Muncie
SIMONTON LAKE DRUGS, Elkhart
THE BOARDROOM, Indianapolis
THE HAM SHACK, Evansville
IOWA
LENWOOD SYSTEMS, Center Point
MEMORY BANK INC., Bettendorf
SERNETT LEISURE CENTER, Carroll
KANSAS
AMATEUR RADIO EQUIPMENT, Wichita
CENTRAL KANSAS COMPUTERS, Herington
COMMUNICATIONS CENTER, Lincoln
GOSUB INTL, Wichita
HATCH COMPUTER CENTER, Alliance
HIGH TECHNOLOGY, Wichita
KENTUCKY
COMPUTER MAGIC, Louisville
PERRY'S COMPUTER, Bremen
LOUISIANA
ACME BOOK CO., Baton Rouge
COMPUTER SHOPPE, Metairie
MAINE
FRYEBURG COMPUTER CENTER, Fryeburg
MAINE MICRO SYSTEMS INC., Auburn
MID-MAINE COMPUTER COMPANY, Auburn
PORTSMOUTH COMPUTER CENTER, Portsmouth
MARYLAND
COMM CENTER, Laurel
PROGRAM STORE, Baltimore
WILLS COMPUTER STORE, Marlow Heights
MASSACHUSETTS
COMPUTER CITY, Charlestown
HI FI EXCHANGE, Northampton
LAND OF ELECTRONICS, Lynn
MARK GORDON COMPUTERS, Cambridge
OMNITEK SYSTEMS, Tewksbury
SOUND COMPANY, Springfield
STAR COMPUTING, Framingham
THE GAME SHOP, East Acton
TUFTS RADIO ELECTRONICS, Medford
MICHIGAN
ALL FOR LEARNING, W. Bloomfield
ALTERNATE SOURCE, Lansing
A.M. ELECTRONICS, Ann Arbor
COMIC KINGDOM, Detroit
COMPUTER CENTER, Garden City
COMPUTER CONNECTION, Farmington Hills
COMPUTER MART, Clawson
COMPUTER MART, Flint
COMPUTERLAND, Kentwood
COMPUTERLAND, Southfield
COMPUTRONIX, Midland
EIGHT BIT CORNER, Muskegon
FERRIS RADIO, Hazel Park
HOBBY HOUSE, Battle Creek
LEARNING CENTER LTD., Ann Arbor
MAIN SYSTEMS, INC., Flint
MID-MICHIGAN MEMORY, Dimondale
NEWMAN COMPUTER EXCHANGE, Ann Arbor
TRI-COUNTY ELECTRONICS & SOUND CENTER, Fenton
WEATHERWAX DRUGS, Brooklyn
WIZARD'S ARSENAL, East Lansing
YE OLDE TEACHERS SHOPPE, Ypsilanti

MINNESOTA
CODE ROOM, Eden Prairie
MINNESOTA SOFTWARE, White Bear Lake
RURAL AMERICA ENTERPRISES, Marshall
ZIM COMPUTERS, Brooklyn Center
MISSISSIPPI
C-COM, Jackson
DYER'S INC., West Point
SOFTWAREHOUSE, Jackson
MISSOURI
CENTURY NEXT COMPUTERS, Columbia
COMPUTER CENTER, Joplin
COMP-U-TRS, Florissant
CRC COMPUTERS, Joplin
D'S, Cameron
HOUSE OF COMPUTERS, Joplin
SOFTWARE SHACK, Belton
MONTANA
COMPUTER STORE, Billings
THE COMPUTER PLACE, Kalispell
NEBRASKA
APPLETREE SOFTWARE, Battle Creek
COMPUTERS WEST, Omaha
GREAT RACE HOBBY PLACE, Lincoln
LEZOTTE STUDIO, Ogallala
NEVADA
CENTURY 23, Las Vegas
HURLEY ELECTRONICS, Las Vegas
NEW HAMPSHIRE
BITSNBYTES COMPUTER CENTER, Concord
COMPUCRAFT, Keene
COVER CRAFT, Amherst
PAUL'S TV, Fremont
PORTSMOUTH COMPUTER CENTER, Portsmouth
RADIO SHACK ASSOC. STORE, Keene
STURDIVANT AND DUNN, Conway

NEW JERSEY
ABC'S TV SALES & SERVICE, Glassboro
ADELMAN'S STATIONERY CO., Union City
BARGAIN BROTHERS, West Trenton
CHANNEL 1 RADIO SHACK, Medford
COMPUTER FORUM, Redbank
COMPUTER MADNESS, Englishtown
COMPUTERWORLD OF WESTFIELD, Westfield
CROWLEY'S, Whitehouse Station
DAVE'S ELECTRONICS, INC., Pennsville
ELECTRONIC WORLD, Mantua
HOBBYMASTERS, Red Bank
J&J ELECTRONIC WORLD, Mantua
LASHEN ELECTRONICS, INC., Denville
MIDAS DATA SYSTEMS INC., Marlton
OMNIFAX, Cherry Hill
RADIO SHACK ASSOC. STORE, Moorestown
RADIOS UNLIMITED, Somerset
SOFTWARE CITY, River Edge

NEW MEXICO
AUTEL ELECTRONICS CO., Albuquerque
J&W ENTERPRISES, Clovis
MITCHELL MUSIC, Carlsbad
THOMAS E. CARR JEWELER, Alamogordo
WARGAMES WEST, Albuquerque

NEW YORK
ARISTO CRAFT DISTINCTIVE MINIATURES, New York
ASD HOME COMPUTER CENTER, Poughkeepsie
BERLINER COMPUTER CENTER, New Hyde Park
C. HABILD OF NEW DORP, Staten Island
COMPUTER CORNER, White Plains
COMPUTER RESOURCES, Williamsville
COMPUTER SHOP, Kingston
COMPUTERLAND, Carle Place
COMPUTERLAND OF NYC, New York
DIGIBYTE SYSTEMS, New York
80-MICROCOMPUTER SERVICES, Cohoes
FUTURE VISIONS COMPUTER STORE, Melville
PHANTOS RESEARCH, Camillus
PROGRAMS UNLIMITED, Jericho
MR. COMPUTER, Wappingers Falls
OMNIFAX, DeWitt
SOFTRON SYSTEMS, Rensselaer
WATERLOO HOBBIES, Mineola
WORLD OF COMPUTERS, Port Chester

NORTH CAROLINA
RAYBURN MICRO-ELECTRONICS, Sylva
SOUND MILL, Havelock

OHIO
ABACUS II, Toledo
ALTAIR SYSTEMS, INC., Dayton
ASTRO VIDEO ELECTRONICS, INC., Lancaster
CINCINNATI COMPUTER STORE, Cincinnati
COMPUTER STORE, Toledo
COMPUTERLAND, Columbus
COMPUTERLAND, Mayfield Heights
COMPUTERLAND, North Olmsted
COMPUTERLAND, Warren
JOBAR ENTERPRISES, Middlefield
MICROAGE, Columbus
MICRO COMPUTER CENTER, Centerville
MICRO ELECTRONICS INC., Columbus
MICRO-MINI COMPUTER WORLD, Columbus
TWENTY-FIRST CENTURY SHOP, Cincinnati
WANNA PLAY, Cincinnati
OKLAHOMA
COMPUTER STORE, INC., Tulsa
COMPUTER WORLD, Tulsa
PROFILE INC., Moore
RADIO SHACK ASSOC. STORE, Guymon
SOUNDS, ETC., Watonga
OREGON
COMPUTER FATHWAYS, Salem
L&R ELECTRONICS, Grant Pass
PIONEER ELECTRONICS, Sandy
TRS-80 PRODUCTS LTD., Portland

PENNSYLVANIA
ALLIED HOBBIES, Philadelphia
ARTCO ELECTRONICS, Kingston
BELL ELECTRONICS, Girard
COMPUTERLAND, Gibsonia
COMPUTERLAND OF HARRISBURG, Mechanicsburg
ERIE COMPUTER, Erie
JNE COMMUNICATIONS, Altoona
MAFEX ASSOCIATES, Johnstown
OMNIFAX, Feasterville
OMNIFAX, Philadelphia
PERSONAL COMPUTER CORP., Paoli
PITTSBURG COMPUTER STORE, Pittsburg
STEVENS RADIO SHACK DEALER, Phoenixville
ROUTE 30 ELECTRONICS, Latrobe
RUMPELSTILSKIN TOY SHOP, New Hope
TELEVISION PARTS COMPANY INC., New Brighton
PUERTO RICO
MICRO COMPUTER STORE, Caparra Terrace
RHODE ISLAND
COLONIAL ENTERPRISES, Foster
SOUTH CAROLINA
OMNI ELECTRONICS, Charleston
TENNESSEE
ACE MINI SYSTEMS, Clarksville
CHATTANOOGA COMPUTER CENTER, Chattanooga
COMPUTER WORLD, Nashville
COMPUTERLAB, Memphis
TEXAS
CODEDATA INC., Arlington
COMPUSHOP, Bellaire
COMPUSHOP/FM 1960W, Houston
COMPUSHOP/N FWY, Houston
COMPUTER 'N THINGS, Austin
COMPUTER PROGRAMMING ASSOC., Lubbock
COMPUTER PORT, Arlington
COMPUTER SALES & SERVICE, Fort Worth
COMPUTER SOLUTIONS, San Antonio
COMPUTERLAND OF SW HOUSTON, Houston
COMPUTERS BY O'NEILL, Lake Jackson
COMPUTERS TO GO, Austin
COMPUTEX, Webster
CORSAIR, Ft. Worth
GATEWAY ELECTRONICS, Houston
KA ELECTRONICS, Dallas
MAC'S TV, Fairfield
MARYMAC INDUSTRIES (R/S DEALER), Houston
PAN AMERICAN ELECTRONICS (R/S DEALER), Mission
R.L. COLE'S ELECTRONICS, San Antonio
ROY'S CB & ELECTRONICS, Aransas Pass
80 SOFTWARE, San Antonio
WAGHALTER BOOKS, INC., Houston
UTAH
COMPUTERLAND, Salt Lake City
FOOTHILL MODELERS, Salt Lake City
VIRGINIA
COMPUTER SOLUTIONS, Leesburg
COMPUTER WORKS, INC., Harrisonburg
HOME COMPUTER CENTER INC., Virginia Beach
LITTLE SOLDIER, Alexandria
SYSTEMS MARKETING, Arlington

WASHINGTON
AMERICAN MERCANTILE COMPANY, Seattle
COMPUTERLAND, Bellevue
COMPUTERLAND, Federal Way
COMPUTERLAND, Spokane
EMPIRE ELECTRONICS, Sunnyside
J.B. SALES, Snohomoish
LORDS, Port Angeles
UNIVERSITY VILLAGE MUSIC, Seattle
U.S.S. ENTERPRISE, Kirkland
WESTERN MICROCOMPUTER CENTER, Bellingham

WEST VIRGINIA
COMPUTER CORNER, Morgantown
COMPUTER STORE, Huntington
OHIO VALLEY ELECTRONIC SYSTEMS, Wheeling
SOUND & ELECTRONIC SPECIALTIES, Morgantown
WISCONSIN
BYTE SHOP, Milwaukee
COMPUTERLAND, Madison
COMPUTERLAND OF FOX RIVER VALLEY, Oshkosh
COMPUTER WORLD, Green Bay
MAGIC LANTERN COMPUTER, Madison
OMEGA MICROS, Milwaukee
PETTED MICROSYSTEMS, Milwaukee
S&O TV SALES, Monroe
WYOMING
COMPUTER CONCEPTS, Cheyenne

AUSTRALIA
DE FOREST SOFTWARE, Nunawading, Vic.
CANADA

Distributor:
MICRON DISTRIBUTING,
Toronto, Ont.

ALLIED COMPUTER CENTRE, Thunder Bay, Ontario
ARKON ELECTRONICS, Toronto, Ontario
AULL COMPUTER SYSTEMS, Victoria, B.C.
BITS & BYTES, Dartmouth, N.S.
CENTRAL DISTRIBUTORS LTD., Lachine, Quebec
COMPUCORNER, Smithers, B.C.
COMPUMART, Ottawa, Ontario
COMPUTER BARN, Sarnia, Ontario
COMPUTER CIRCUIT, London, Ontario
COMPUTER INNOVATIONS, Ottawa, Ontario
COMPUTER SHOP LTD., Calgary, Alberta
COMPUTER WORLD, Vancouver, B.C.
COMPUTERLAND OF BURLINGTON, Burlington, Ontario
CREATIVE COMPUTERS, Victoria, B.C.
DATATEC COMPUTER SYSTEMS LTD., Saskatoon, Sask.
ELECTRONICS 2001, Willowdale, Ontario
GALACTICA COMPUTERS LTD., Edmonton, Alberta
IRISCO DU QUEBEC, Ste. Foy, Quebec
LYONS LOGIC LTD., London, Ontario
MICROMATION, Toronto, Ontario
MICROSHACK, Saskatoon, Sask.
MICRO SHACK, Regina, Sask.
M&W COMPUTERS, Mississauga, Ontario
NIP & TUCK VARIETY, London, Ontario
OFFICE CENTRE, Kingston, Ontario
SAULT OFFICE MACHINES, Sault St. Marie, Ontario
STATUS COMPUTER SYSTEMS, St. Catherines, Ontario
TOTAL COMPUTER SYSTEMS, Ajax, Ontario
WEST WORLD COMPUTERS, Edmonton, Alberta
ITALY
BITS & BYTES, Milan
NEW ZEALAND
VISCOUNT ELECTRONICS, Palmerston North
NORWAY
A/S SORLUND, Vedavagen
SWEDEN
SENTEC AB, Jarfalla
UNITED KINGDOM
CALISTO COMPUTERS, Birmingham
THE SOFTWARE HOUSE, London
WEST GERMANY
MICROSTUFF, Frankfurt
NETHERLANDS & BELGIUM

Distributor:
SOFTWARE IMPORT BRABANT,
Eindhoven, Neth.

COMPUTER COLLECTIEF, Amsterdam
MICRO COMPUTING, Krommenie
MICRO DYNAMICS, Eindhoven
MUSICPRINT-CHIP, Arkel
OVEL GONNE, Drachten
R.A.L. MICROCOMPUTERS, The Hague

YOUR NAME COULD BE HERE. CALL: 1-800-258-5473

Find out what you're missing.

You heard the lady. Your microcomputing life *is* going to be very exciting. But don't leave it to chance—leave it to us! We have a brand new catalog full of packages and valuable coupons to help make it happen. It's free. . . if you fill out the coupon and mail it today. (Or make a photocopy and spare your magazine).

NAME: ______________________

ADDRESS: ______________________

CITY: ______________ **STATE:** ______ **ZIP:** ______

Instant Software Inc. ✓535

Peterborough, N.H. 03458 603-924-7296

Some ham software.

Morse Resource Part II

Allan S. Joffe W3KBM
1005 Twining Road
Dresher, PA 19025

In the programs discussed last month in part one of this article, we stored a message in a string. If we empty this string into an area in RAM that is free memory, the string can be refilled. We can then empty the refilled string into any contiguous free RAM area. This can be done at least four times, giving us 1000 or more characters. This area in RAM is a buffer, a storage area of convenience.

In Program Listing 1 line 5 clears enough string space to allow us to fill A$ to capacity. Line 10 sets the area in free RAM where our buffer area will exist. Line 20 gives us a clear screen at the start of the program, lists an instruction, and allows us to insert characters into A$. Lines 30, 40, 70, 90 and 100 store the contents of A$ into succeeding memory cells in the buffer area.

```
5    CLEAR 300
10   X = 24000:REM START OF RAM BUFFER AREA
20   CLS:INPUT "TYPE IN MESSAGE";A$
25   LN = LEN(A$)
30   FOR J = 1 TO LN
40   CODE$ = MID$(A$,J,1)
50   IF CODE$ = "*" GOTO 20:REM SUBSTRING TERMINATOR
60   IF CODE$ = "#" GOSUB 6000:REM END OF MESSAGE TERMINATOR
70   POKE X,ASC(CODE$)
90   X = X + 1
100 NEXT J

2000 FOR X = 24000 TO 25000:REM GIVES 1000 CHARACTER BUFFER AREA
2010 A = PEEK(X): PRINT CHR$(A);
2015 IF A = 32 FOR W = 1 TO 50:NEXT W:NEXT X:REM SEE TEXT
2020 REM ENTIRE CHARACTER DIRECTORY IS NOW ENTERED
2025 IF A = 65 THEN G$ = "SLE"
2030 IF A = 66 THEN G$ = " LSSSE"
2040 IF A = 67 THEN G$ = "LSLSE"
2045 IF A = 68 THEN G$ = "LSSE"
2050  .
       .      The balance of the character directory is entered
        .     with the pattern that is shown.
         .
          .
           .
            .
4000  LG = LEN(G$)
4010  FOR H = 1 TO LG
4020  XMT$ = MID$(G$,H,1)
4030  IF XMT$ = "S" GOSUB 5000
4040  IF XMT$ = "L" GOSUB 5020
4050  IF XMT$ = "E" GOSUB 5040
4080 NEXT H,X
4090 END
5000 OUT 255,4:FOR TM = 1 TO 20:NEXT TM: OUT 255,0:RETURN
5020 OUT 255,4:FOR TM = 1 TO 40:NEXT TM: OUT 255,0:RETURN
5040 FOR TM = 1 TO 20:NEXT TM:RETURN
6000 INPUT "PRESS ENTER TO TRANSMIT MESSAGE";Z$:RETURN
```

Program Listing 1.

Lines 50 and 60 are terminators. Line 50 has an asterisk as the terminator. The first character to be typed into the message is a quotation mark, which handles the situation peculiar to proper functioning of the comma. Assume that the message will exceed one string capacity (255 characters and spaces). If you type in approximately three and one-half lines of text, your average character and space count will approach the string limit.

The last character entered must be an asterisk. If you exceed the limit, simply backspace to a logical end point and insert the asterisk.

Now hit Enter and the program POKEs the ASCII value of each character of the string into the buffer. The program knows that all the meaningful characters have been stored when it hits the asterisk. The program now goes back to line 20 for a refill. Enter the next A$ with the rest of the message. Follow all procedures as with the first string.

At the end of the final string, the terminator (#) in line 60 is used. When the program encounters the #, it goes to line 6000 and lets you transmit the message by pressing Enter.

A Dual Function

Line 2015 serves a dual function. The first function takes care of any accidental spaces typed in prior to the terminator symbol in any given string. Without line 2015 the last character of such a line would repeat. This function is taken care of by the expression: IF A = 32 NEXT X.

The function generated by the timing loop using the variable W is a new function. It generates a between-words space that you can adjust by changing the value of the timing loop. ASCII 32 is the code value for a space, so a portion of line 2015 is a space detector.

The terminator character may be any character on the keyboard that is *not* included in your character list in the program.

Easy Repeat

Suppose that you have just sent a message. You heard it go out and watched the screen as each character was transmitted. If you hit Enter without entering a new message, and then hit the Enter key again, the

"If you have to send too many CQ messages... is your antenna still up after the last big blow?"

old message will once more be transmitted.

Here is the scenario: It's a dull day on 40 meter CW and you decide to call CQ. You load your general calling message, transmit it and tune across the band. Hmmmm, nobody came back to you. Repeat the message by merely hitting Enter, follow the direction on the screen, and hit Enter again. Your CQ goes out once more, again and again if needed. This a nice convenience.

By the way, if you have to send too many CQ messages... is your antenna still up after the last big blow?

Special Message Insertion

Insert the following line between lines 20 and 25:

IF A$ = "&" GOSUB 7000. Now insert the calling message in a subroutine in line 7000. 7000 A$ = "CQ CQ CQ DE W3KBM W3KBM K #":RETURN. Notice the use of the final terminator in the subroutine message.

When you run the program and wish to recall the special messasge, merely type in & and hit Enter. This is the case where you do not type in a terminator. If you did the pro-

```
5   DEFINT A-Z : DIM CODE$(100)
10  R = 24000:REM START OF BUFFER AREA IN RAM
15  CLS:PRINT "TYPE IN MESSAGE"
20  A$ = INKEY$:PRINT A$;
25  REM NOW WE ENTER CHARACTER LIST WHICH GETS INFORMATION INTO
        THE RAM BUFFER LOCATIONS
30  IF A$ = "A" POKE R,65: R=R+1
35  IF A$ ="B"  POKE R,66: R=R+1
40  IF A$ = "C" POKE R,67: R=R+1
45  IF A$ = "D" POKE R,68: R+R+1
50.
  .         Enter the rest of the character list following
   .         the indicated pattern. AFTER ENTERING THE COMPLETE
    .        CHARACTER LIST YOU THEN ENTER THE FOLLOWING LINES
     .       USING YOUR APPROPRIATE LINE NUMBERS.

      IF A$ = "#" POKE R,35: GOTO 1000
      IF A$ = ":" THEN R = R-1
      if A$ = " "POKE R,32:R=R+1
990 GOTO 20
1000 FOR X = 24000 TO 25000:REM THIS SETS BUFFER CAPACITY
1010 B = PEEK(X)
1015 IF B = 35 GOTO 10
1020 IF B = 32 THEN FOR W = 1 TO 10:NEXT W:REM Between word spacing
1025 REM NOW WE HAVE TO INSERT A CHARACTER LIST TO GET THE
        INFORMATION "OUT" OF THE BUFFER
1030 CODE$(65) = "SLE"
1035 CODE$(66) = "LSSSE"
1040 CODE$(67) = "LSLSE"
1045 CODE$(68) = "LSSE"
1050 CODE$(69) = "SE"
1055.
   .        The balance of this code character list is entered
    .       following this same pattern. Depending on how close
     .      you space line numbers, it may be necessary to
      .     change following line numbers to fit your style.
       .
2000 LN = LEN(CODE$(B))
2010 FOR H = 1 TO LN
2020 XMT$ = MID$(CODE$(B),H,1)
2030 IF XMT$ = "S" GOSUB 5000
2040 IF XMT$ = "L" GOSUB 5020
2050 IF XMT$ = "E" GOSUB 5040
2060 NEXT H,X
2070 END

   The three subroutines are identical to those of Listing # ONE
so merely duplicate them at this point.
```

Program Listing 2.

See List of Advertisers on page 354

"The important thing is getting the message through."

Program Listing

```
50 REM THIS PROGRAM WAS DEVELOPED BY ALLAN S.JOFFE.  PART OF IT
WAS USED AS A DEMO PROGRAM CALLED MORSE RESOURCE WHICH WAS THEN
 SUBMITTED TO 80 MICROCOMPUTING....AUGUST 1980                 *
**
100 REM CW PGM FOR LONG MESSAGES.AFTER EACH THREE LINES OF TEXT
TYPE IN AN * AND WHEN PROMPT RETURNS CONTINUE TYPING NEXT 3 LINE
S.THEN INSERT ANOTHER * AND CONTINUE.  AT END OF FINAL LINE  INS
ERT AN # WHICH ACTS AS A TERMINATOR FOR MESSAGE.              ***
*
120 REM SO THAT A COMMA IN TEXT WILL NOT GIVE EXTRAS IGNORED MSG
 FIRST CHARACTER OF EACH FIRST LINE OF TEXT MUST BE A "       ***

140 REM LATEST VERSION WITH ALL FEATURES***AND ADDING SPEED
CONTROL AT THE BEGINNING OF THE PROGRAM  8/1/80            *****

145 CLS:PRINT "INSTRUCTIONS---FIRST CHARACTER OF EACH MESSAGE MU
ST BE A QUOTE  SIGN. THEN TYPE IN MESSAGE.  IF MESSAGE IS OVER T
HREE FULL LINES OF TEXT, INSERT AN ASTERISK * AT END OF LINE AND
 PRESS ENTER"
147 PRINT:PRINT:PRINT"INSERT NEXT SECTION OF MESSAGE STARTING  A
GAIN WITH A QUOTE SIGN IF THIS IS FINAL PART OF MESSAGE END   WI
TH A # TERMINATOR     CHARACTER ELSE USE AN ASTERISK  TO PUT IN
NEXT PART OF MESSAGE"
150 PRINT:PRINT:PRINT "REPEAT IF NECESSARY.  FINAL CHARACTER OF
YOUR TOTAL MESSAGE MUSTBE THE FINAL TERMINATOR #"
155 INPUT "TO START PROGRAM HIT ENTER";N
160 CLS:CLEAR 300:DEFINT A-Z
200 INPUT "SET SPEED BY TYPING 10 OR 15 OR 20";L:IF L=10 THEN G=
30  ELSE IF L= 15 THEN G=20 ELSE IF L=20 THEN G=10
210 IF L= 10 OR L=15 OR L=20 GOTO 240 ELSE PRINT "YOU HAVE ENTER
ED AN INCORRECT SPEED":FOR I = 1 TO 500:NEXT:GOTO 200:REM THIS
PROTECTS AGAINST ERROR IN SPEED CHOICE
240 X= 24000:REM START OF RAM BUFFER AREA
260 CLS:INPUT "TYPE IN MESSAGE";A$
280 IF A$="&" GOSUB 1660:REM EXAMPLE OF CANNED MESSAGE INSERTION
300 LN= LEN(A$)
320 FOR J= 1 TO LN
340 CODE$= MID$(A$,J,1)
360 IF CODE$="*" GOTO 260
380 IF CODE$="#" GOSUB 1640
400 POKE X,ASC(CODE$)
420 X=X+1
440 NEXT J
480 FOR X = 24000 TO 25000:REM BUFFER CAPACITY 1000 CHARACTERS
500 A= PEEK(X):PRINT CHR$(A);
520 IF A< 64 GOTO 1080:REM SPEEDS UP PGM EXECUTION
540 REM FIRST PART OF LINE 1080 CONTROLS REPEAT OF LAST CHAR IF
IT IS FOLLOWED BY A TYPED IN SPACE. MIDDLE DELAY LOOP CONTROLS
SPACE BETWEEN WORDS IN MSG.
560 IF A= 65 THEN G$="SLE"
580 IF A=66 THEN G$="LSSSE"
600 IF A = 67 THEN G$="LSLSE"
620 IF A = 68 THEN G$="LSSE"
640 IF A = 69 THEN G$="SE"
660 IF A= 70 THEN G$= "SSLSE"
680 IF A = 71 THEN G$="LLSE"
700 IF A = 72 THEN G$= "SSSSE"
720 IF A = 73 THEN G$= "SSE"
740 IF A = 74 THEN G$= "SLLLE"
760 IF A = 75 THEN G$= "LSLE"
780 IF A = 76 THEN G$= "SLSSE"
800 IF A = 77 THEN G$= "LLE"
820 IF A = 78 THEN G$ = "LSE"
840 IF A = 79 THEN G$= "LLLE"
860 IF A = 80  THEN G$= "SLLSE"
880 IF A = 81 THEN G$= "LLSLE"
900 IF A = 82 THEN G$= "SLSE"
920 IF A = 83 THEN G$ = "SSSE"
940 IF A = 84 THEN G$ = "LE"
960 IF A = 85 THEN G$= "SSLE"
980 IF A = 86 THEN G$= "SSSLE"
```

Program continues

gram would operate on the &.

Now proceed normally and your special message will be sent. You can retransmit the special message once it is in the buffer by hitting Enter, following directions, and hitting Enter once again.

By restructuring the line numbers it is possible to insert a whole series of canned messages that may be recalled by entering the calling character of that particular message.'

Speed

Key in this short program:

```
10 FOR X = TO 1000
20 PRINT X;
30 NEXT X
```

Run this and time the program from start to finish. It runs in about 16 seconds.

Now run the same program with the addition of a new line 5: 5 DEFINT X. Run the program again. It should take about ten seconds, 35 percent faster than the first time.

Line 5 defines the variables used so that they are treated as integers by the computer. This makes less processing work for the Basic interpreter and allows it to run the program faster.

Speed is nice but the important thing is getting the message through. Sometimes speed is incompatible with that goal.

Inkey$ Meets the Buffer

The program using INKEY$ can also incorporate the buffer feature. One problem with the instantaneous version of the program was that of error transmission: If you hit the wrong key, that character is instantly sent. Program Listing 2 shows the approach to INKEY$ and a + buffer.

DEFINT is used to speed program execution. The # is the end-of-message terminator. As soon as you enter #, the message is sent. This gives you control of transmission.

The program sets the buffer size at about 1000 characters, but you may set the size to your needs.

The line that reads IF A$ = ":"THEN R = R − 1 allows you to correct a typing error. When you type in each character, the next character is entered in the next numerical buffer location by the R = R + R expression that follows each entry in the top of the program character list. If you mistype a character, you can wipe it off the screen by using the backspace arrow, but this will not remove the erroneous character from the buffer memory cell. If you enter a colon after you see an improper character entry, the program will reduce the R counter to one. Now if you type in the proper character, your transmission will be correct. ■

Program continued

```
1000 IF A = 87 THEN G$= "SLLE"
1020 IF A = 88 THEN G$= "LSSLE"
1040 IF A = 89 THEN G$= "LSLLE"
1060 IF A = 90 THEN G$= "LLSSE"
1080 IF A= 32 THEN FOR W = 1 TO 50:NEXT W:NEXT X
1100 IF A = 48 THEN G$="LLLLLE"
1120 IF A = 35 THEN X= 24000:GOTO 260
1140 IF A = 49 THEN G$= "SLLLLE"
1160 IF A = 50 THEN G$= "SSLLLE"
1180 IF A = 51 THEN G$= "SSSLLE"
1200 IF A = 52 THEN G$= "SSSSLE"
1220 IF A = 53 THEN G$= "SSSSSE"
1240 IF A = 54 THEN G$= "LSSSSE"
1260 IF A = 55 THEN G$= "LLSSSE"
1280 IF A = 56 THEN G$= "LLLSSE"

1300 IF A = 57 THEN G$= "LLLLSE"
1320 IF A = 44 THEN G$="LLSSLLE":REM COMMA
1340 IF A = 46 THEN G$="SLSLSLE":REM PERIOD
1360 IF A = 47 THEN G$= "LSSLSE":REM /
1380 IF A = 63 THEN G$= "SSLLSSE":REM ?
1400 LG = LEN(G$)
1420 FOR H= 1 TO LG
1440 XMT$=MID$(G$,H,1)
1460 IF XMT$ ="S" GOSUB 1580

1480 IF XMT$="L" GOSUB 1600
1500 IF XMT$="E" GOSUB 1620
1520 NEXT H,X
1540 END
1580 OUT 255,4:FOR TM = 1 TO G :NEXT TM:OUT 255,0:RETURN
1600 OUT 255,4:FOR  TM= 1 TO 3*G:NEXT TM:OUT 255,0:RETURN
1620 FOR TM = 1 TO G :NEXT TM:RETURN
1640 INPUT "PRESS ENTER TO TRANSMIT MESSAGE";Z$:RETURN
1660 A$ ="CQ CQ CQ CQ DE W3KBM W3KBM W3KBM K #":RETURN:REM  NOTE
 USE OF TERMINATOR IN THIS MESSAGE  *** *** *** ***
```

How can we keep them down on the farm after they've got their TRS-80?

Spanning the Electronic Nation

Dennis Bathory Kitsz
Roxbury, VT 05669

The Microconnection
Microperipheral Corp.
Mercer Island, WA
$249

Lynx
Emtrol Systems, Inc.
Lancaster, PA
$249

Telephone Interface II
Radio Shack
Ft. Worth, TX
$250 (RS232 board at $100 and Expansion Interface at $299 also required.)

Be warned: The world of communications is large; the TRS-80 is small. But once you've brought one to the other, your data can travel the breadth of the electronic nation. You won't stay down on the farm for long after that.

The way the TRS-80 manipulates information is not much different than the way other computers do it, and so it can easily communicate with other machines. By converting data to a continuous stream of pulses, and using those pulses to change the pitch of audio tones, a powerful telephone connection to other computers is possible. Streams of data are also used to communicate directly with what are known as serial peripheral devices, including printers.

This communication process has become relatively standardized over the years. For example, a computer talks to a Teletype printer at 110 bits of information per second. This is called the baud rate. Standard baud rates run from a low of the Teletype's 110 (there are lower nonstandard rates) to a high of 19,200. Telephone communication over ordinary voice lines is conducted at 300 baud; special private (and more expensive) data lines can handle rates from 600 baud and up.

Other parts of data communications "protocol," as it is known, are the character code itself (usually the familiar ASCII, although there are obsolete formats and IBM's own EBCDIC), the number of start, stop and data bits, and a verification bit known as parity.

The TRS-80 Model I is not outfitted with communications capabilities, but several companies, including Radio Shack, offer special hardware for serial transfer of information. These hardware additions come with a variety of capabilities and support. They are called *modems*, which stands for modulator/demodulator.

The Telephone Interface II modem; telephone handset cradles in the rubber cups.

The Telephone Interface II from Radio Shack is an acoustic device; that is, the telephone handset fits into a pair of rubber cups which block out unwanted sounds and permit the Interface's speaker and microphone to join closely to the telephone's microphone and earphone. The Interface requires RS-232C input. Simply stated, RS-232C is an industry standard definition of communications hardware according to voltages and polarities. The Radio Shack RS-232 board fits into the TRS-80 expansion box.

The total hardware required to communicate with Radio Shack's hardware is the Ex-

"Your data can travel the breadth of the electronic nation. You won't stay down on the farm for long after that."

pansion Interface, RS-232 board, and telephone interface.

The Microconnection, manufactured by the Microperipheral Corporation, is a direct-connect device. It comes with a modular plug to snap into a telephone outlet, rather than using audio coupling to the handset. It plugs directly into the edge card connector of either the keyboard unit or the expansion interface, and is complete as sold.

The Lynx, made by Emtrol Systems, Inc., is also a complete direct-connect communications box. Like The Microconnection, it is supplied with a cord and modular plug, and fastens to the keyboard or expansion interface.

All three units come with familiar black-box power supplies.

Advantages and Disadvantages

None of these communications attachments is perfect, but for simple telephone applications, all are useful. Beyond that, the questions become more complicated.

Most versatile of the three is the Radio Shack combination. It provides a full range of standard baud rates and other communications parameters, as well as the best documentation. It has the largest body of software (Radio Shack's and others') among the three units.

Unfortunately, it's expensive. An expansion interface is required to support an RS-232 board, and that in turn is required to run the Telephone Interface. Simple software is provided, but full-featured communication packages start at $50. The total tab can be higher than $650, not including the telephone bills.

Next on the list is The Microconnection. Variable protocols are provided, but the baud rate is limited to a standard telephone rate of 300. It is the easiest to get up and running, and is provided with an excellent dumb terminal program and fine instructions. Direct connection to the telephone line requires that you formally notify the phone company in some areas. The Microconnection also provides a voice input for use in ham transmissions and other voice/data combinations, and a standard RS-232 output jack is included for talking to 300-baud serial printers, etc.

The available software pool for The Microconnection was initially small, limited to that sold by its manufacturer. Their more sophisticated software package, which is an additional $50, has some bugs and is far from crash-free. Now, however, other commercial communications software is becoming available in forms modified for use by The Microconnection (sometimes at an additional charge). All told, the device and software is an attractive alternative if your goal is straightforward telephone communication.

Finally, the Lynx also provides variable protocols and the single telephone rate of 300 baud. It too connects easily, but the instructions are poorly written and the program was fragile and crashed often during use. It is less sensitive to signals than either The Microconnection or Telephone Interface II, resulting in some garbled or lost data. It has no RS-232 output, limiting it exclusively to telephone use.

Software written for the Radio Shack RS-232/Telephone Interface will generally work with the Lynx, because the Lynx is arranged to use the same output port number as the Radio Shack board. This results in an interesting problem—the Radio Shack board must be disconnected if the Lynx is to be used. Otherwise, confused signals and computer bus conflict will be the result, since both use ports 232, 233, 234, and 235. This means that if for any reason you have decided not to use the acoustically coupled Telephone Interface, but have nonetheless installed the Radio Shack RS-232 to drive a serial printer, you must not use it simultaneously nor even have it connected when the Lynx is in place. This can be a serious drawback; keep a screwdriver handy.

There is little difference in performance between the direct-coupled Microconnection and the acoustical Telephone Interface II. The sensitivity of the latter is quite good, and the rubber earcups protect the unit from extraneous sounds. Striking or bouncing the unit (or tripping over the phone cord, my favorite hobby) will cause some minor data loss.

The Microconnection discriminates very well between noise and signals, giving error-free data with most reasonable telephone situations.

A Tale of Two Cities

The only problem I had with the unit might be unique, but if you live in an area where the quality of telephone service is not high, this tale will be of interest. The Microconnection, as with all devices made to attach directly to the phone lines, has an FCC serial number which must be provided to the telephone company by the user along with notice that the communications device is in place.

Not wanting to get involved with our telephone company, which is a private, strictly one-horse system, I connected the device without notifying them. Alas, the send/receive situation was impossible. A phone call to the modem's manufacturer resulted in a complete and courteous walk-through on how to increase the device's sensitivity, as well as a 3000-mile measurement of the signal loss from tiny Roxbury to bustling Seattle. It looked bad. Even with a tweaked Microconnection, the data was garbled much of the time.

A hesitant phone call to the telephone company resulted in two service calls and a stiff lecture about the required notification due them of the device and its serial number. But $20 worth of useless data calls later, the modem was still not doing the job. The Microperipheral Corporation was very helpful, but claimed the telephone company was at fault; the phone company and its Neanderthal engineers wanted to take The Microconnection for examination, claiming it must be at fault. Nothing doing, said I to the latter request.

For some days there was a stalemate, until the Telephone Interface II arrived. Oddly,

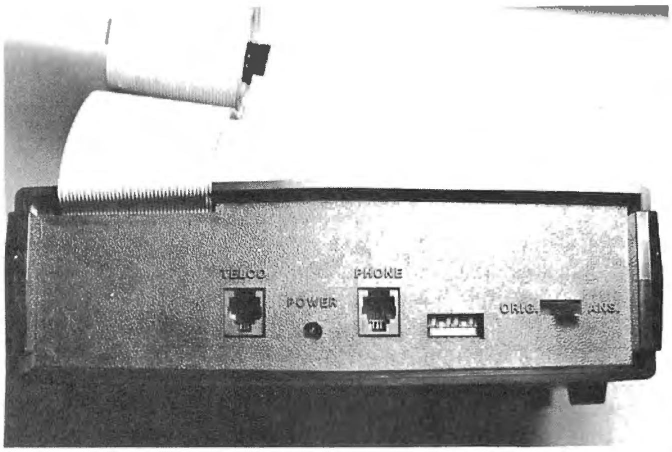
The Connection compared in size to the Exatron Stringy-Floppy.

"But $20 worth of useless data calls later, the modem was still not doing the job."

it was more sensitive than The Microconnection under these adverse conditions, but still the lost data was more than I was prepared to expect.

Finally the phone company admitted the truth: In a typical economy move, they had bought what they called "loading coils," at surplus, to balance the lines between Roxbury and the home office eight miles to the north. These had not been installed, they said, because no brackets came with them, and both the coils and brackets were obsolete. Perhaps in a year or two the situation would be corrected.

The Public Service Commission then re-



The back panel of the Lynx has telephone connections, user programming switches, and originate answer switch.

The front panel of the Lynx indicates carrier, and has voice data switch.

"My monthly bill charges me an extra fifty cents for 'Miscellaneous Business Equipment.' "

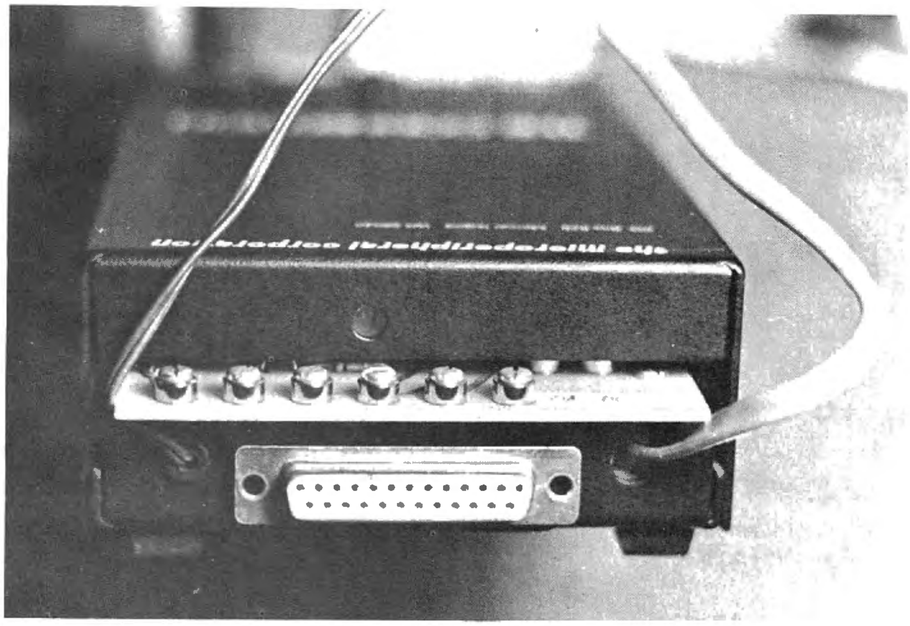
ceived a polite note. A few days later, phone company engineers and installers were standing outside my home in light snow, pondering the problem.

At last those lines (the first private lines in Roxbury, a fact of which I was very proud) were switched with an older (and properly balanced) party-line set. All four party lines had to be dedicated to my single telephone, because there were no other options for the company. I received credit for the unsuccessful data calls—but I also received an installation bill for The Microconnection, even though I did the installation myself. My monthly bill charges me an extra fifty cents for "Miscellaneous Business Equipment."

A writer from Canada tells me installation of non-phone-company equipment is even harder there, and he has no desire to fight for the multiple permits needed.

Back to the Evaluation

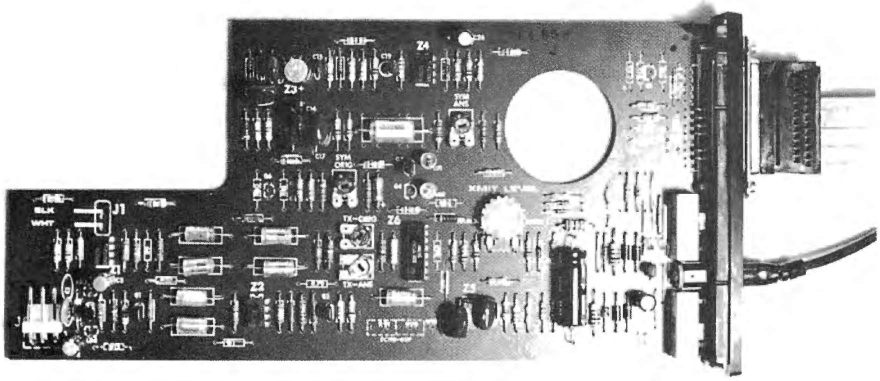
By this time, the Lynx had arrived. With the new lines in operation, I expected no problems, but data was lost on long dis-



The back panel of The Connection has microphone connecting screws, RS-232 output, power and telephone cables.

✓See List of Advertisers on page 354

"The Microconnection is very compact, and created on two piggyback boards inside a small case."



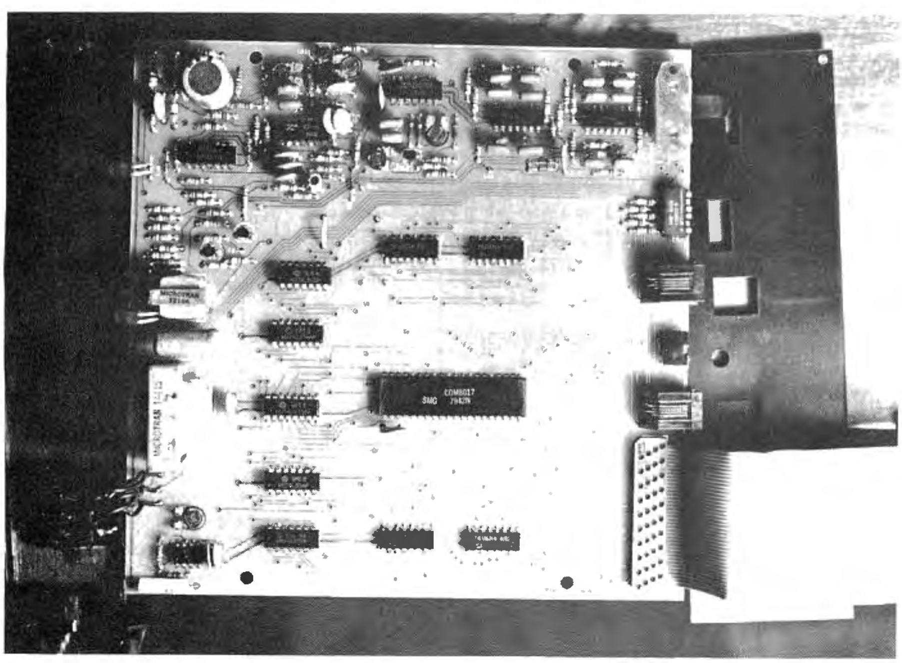
Inside the Telephone Interface II: holes and slot are for microphone and speaker, disconnected in this photo.

tance calls. The Microconnection and Telephone Interface II continued to work very well, so I conclude that the Lynx unit I received for evaluation was not as sensitive or discriminating in its signal reception. As evidenced by comments by those on the receiving end, though, all three units transmit exceptionally well, with no data loss.

There are two potential problems with data transmission, one which pertains to all modems, the other exclusively to the Telephone Interface II. The first difficulty is the way in which some long-distance telephone calls handle the audio. One side of the line dominates the signal, blanking out the audio from the other side. This makes two-way transmission impossible, and if the program being used depends on sensing a "carrier" tone at the far end to operate, it will freeze or report an error. The only solution to this is to dial again and hope to get a line capable of simultaneous transmission of two signals. I have had this problem mostly during peak calling periods.

The other problem is that the acoustic connection may overload newer electronic telephones, or the operation of the computer may interfere with the expensive cordless telephones being sold. A Radio Shack manager reports that the Telephone Interface II simply will not work with the electronic telephones in his area, where the phone company is actively replacing its older electromechanical devices with sophisticated electronic ones.

As for the cordless phones, the manager reports that, modem or not, even the presence of an operating TRS-80 (which emits wide-band radio-frequency interference) will blot out the operation of any cordless phone in the vicinity. I have not had a chance to explore this, but recommend that readers with recent phone installations get an assurance from their Radio Shack manager that the equipment can be returned before purchasing the complete RS-232/Telephone Interface setup.



The Lynx modem board shows loosely-spaced digital circuitry at top, compact audio circuitry at bottom.

Inside

The boards inside the three devices show clean work and high quality components. The Microconnection is very compact, and created on two piggyback boards inside a small case. Its circuit board is fiberglass, and the adjustment controls are stable, precision devices. No circuit diagram is provided with The Microconnection, making user adjustment (which is actually illegal with direct-connect phone devices anyway) impossible. Documentation of the hardware is weak, only briefly commenting on the use of the modem with serial devices and in voice/data operation. User programming information is scanty, though the company will send it on request.

The Telephone Interface II is also compactly designed, and all parts on its board are clearly labeled—a joy when troubleshooting. The case is slim (only an inch thick), and color-coordinated with the rest of the TRS-80 system. A full circuit diagram, an extensive manual, and excellent descriptions of user programming are provided. Once again, Radio Shack provides the star documentation of the lot.

My review copy of the Lynx is the only one of the three using a phenolic paper board, and its parts layout is not compact. Although the Lynx performs well, the wide spacing between parts, combined with a plastic case and the inexpensive circuit board material, make it a candidate for errors, and may explain the slight lack of reliability of this unit. (Later Lynx units have been produced with fiberglass boards, and reports indicate higher reliability.) The user documentation consists of a few illegible pages and a very brief description of the unit's operation. From my point of view, this documentation is entirely unsatisfactory.

One feature common to all the units is the originate/answer option. When two modems communicate, each must transmit at

. . .And my dad says GRAFTRAX 80 does better graphics than anybody.

Epson.

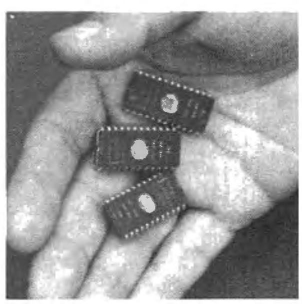
You might say it's the head of its class. GRAFTRAX 80 gives you the highest resolution dot matrix in the world — up to 120 dots per inch horizontally and 216 dots per inch vertically. Which makes our MX-80 and MX-80 F/T work more like a plotter than a printer.

And that's just for starters.

GRAFTRAX 80 more than doubles the features available in an Epson MX-80 or MX-80 F/T. For instance, we've added nice little touches like an italic character set — which doubles the number of font types to 24. And with GRAFTRAX 80, you can change style, size and density anywhere in the line. And backspace the head. And use block graphics with *any* computer. And redefine all escape codes under software control.

Up to now, you couldn't get some of these features at *any* price. With GRAFTRAX 80, you get 'em all. And more. Lots more.

Best of all, we haven't forgotten all those people who already own an MX-80 or MX-80 F/T. You own the printer that set the standards all the others are following. And you can install GRAFTRAX 80 PROMs yourself, for less than $100. And make the best even better.



You can see why we say GRAFTRAX 80 is the head of its class. There's nothing else *in* its class.

404

EPSON
EPSON AMERICA, INC.

3415 Kashiwa Street • Torrance, California 90505 • (213) 539-9140

"The Lynx, weakest of the three, still shows good circuit design and uses quality parts."

a different pitch. The custom is to call one the originator of the call, the other the answer side of the call. The Microconnection did not have this feature on its first model (reviewed for this article), but the present version contains an answer mode.

In sum, the units are well designed, using parts of high quality. Only the Radio Shack unit uses crystal control, although The Microconnection provides signals well within standard tolerance through use of precision resistors, capacitors, and integrated circuits. The Lynx, weakest of the three, still shows good circuit design and uses quality parts.

What to Do with Them

All these modems provide access to telephone communications. Among the most popular telephone uses is connection to a large program and data base, such as The Source or Micronet. These services, for a fee, provide an enormous repertoire of programs and information, including stock reports, powerful text editing programs, and substantial user storage.

Using one of these services is like having a terminal on a large, powerful, time-sharing computer—which is exactly the case. The full computational and storage power of the machines is available to users of home computers like the TRS-80, Apple, etc.; by using a modem and a standard protocol they are all capable of talking the same language. A TRS-80 becomes as powerful as a Control Data terminal.

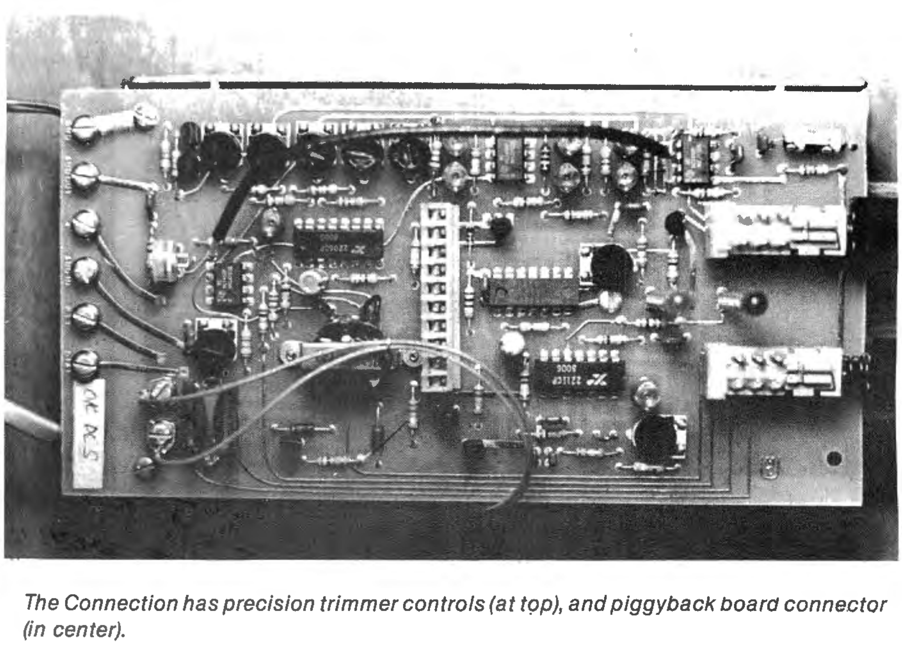
Other uses of the modems are to access free computer bulletin boards (a listing of over 250 is provided with The Microconnection); exchange of original programs and information; and accessing local time-shared computers. Local computers can provide a data base used by regional companies, meaning that engineers, scientists, sociologists, and so forth, can continue their normal work at home, far from their usual terminals.

Banks now permit (and some are actively encouraging) banking at home; financial institutions will feed updated Wall Street reports right to your living room or office. Western Union accepts mailgrams directly from the TRS-80 into a toll-free number.

The future is being demonstrated in France: A program is being set up to provide all telephone customers with a data terminal to replace directory assistance. The telephone company suggests it will be faster and cost less than the present human operators.

Modems capable of interfacing to the TRS-80 continue to appear. Each should be capable of error-free data transmission and reception, flexible user programming, and should be supported by a body of quality software. The Telephone Interface II, The Microconnection, and the Lynx are all up to this challenge.

The Telephone Interface II, a Radio Shack product, can be expected to have continued support from both that company and independent software authors. The Microconnection already has fairly good support available for it, including modifications to Radio Shack and other communications packages. The Lynx, along with its other weaknesses, has little more than manufacturer support at this writing, although most communication programs written for the Radio Shack modem will work with it. ■



The Connection has precision trimmer controls (at top), and piggyback board connector (in center).

TRS-80 (TM)

MAKE A SOUND INVESTMENT IN YOUR MODEL I OR MODEL III LIBRARY

Why did Creative Computing call our MICROCOSM I "a worthwhile purchase"? What caused 80-US to say it "is exceptional"? Simple: BASICS AND BEYOND has produced consistently high-quality, creative, bug-free packages, each focusing on educational, practical and entertaining programs for your Model I or Model III TRS-80 ™ microcomputer. MICROCOSM I with 30 programs and MICROCOSM II with 20 programs, are available for only $24.95 each.

With MICROCOSM III, BASICS AND BEYOND has broken the sound barrier. It includes "Seige", a medieval clash of arms; a serenade from "Computer Organ"; "Pinball", which zings and bongs as it ricochets from flipper to bonuses; step by step learning with "Long Division"; right wrong buzzers; and 16 more. MICROCOSM III is a 20 program package with sound for $24.95. This package available for Model I only.

(914)962-2355

BASICS & BEYOND, inc.

BOX 10b • AMAWALK, N.Y. 10501

No charge for postage or handling. N.Y. residents add 5% sales tax. TRS-80 is a trademark of Radio Shack Division of Tandy Corp. Master Charge and Visa accepted

49

TRS 80™*

A MAJOR BREAKTHRU FOR

Electronic Engineers
Electronic Hobbyists
Electronic Students
Ham Operators

A General DC-AC (steady state) Analysis of Any Circuit

Will Analyze and Compute:

- Node, Branch, Element, Voltages
- Node, Branch, Element, Currents
- Branch Power Dissipation
- Magnitude and phase values and complete frequency response with graphic display
- Modify any element in circuit for desired results

A complete operational manual supplied comparable to I.B.M.'s E.C.A.P.™ Program
*A.C. Analysis Program *$149.95*
*D.C. Analysis Program *$89.95*

To Order Write:
G & L Software Enterp.
2304 N. 1st. Street
Upland, CA 91786

VISA · master charge

Do Not Send Cash in the Mail

TRS 80™ is a trademark of the Tandy Corp., E.C.A.P. is a trademark of International Business Machines, Inc.

39

THE TWENTY FIRST CENTURY IS HERE!

PUBLICATIONS for the TRS-80[1] Model III

Mystery of the ROM **Mystery of the DISKETTE**
Mystery of the DOS **$19.95 each**

Must additions to the computer library of the serious programmer as well as the casually interested. Written in a technical, yet easy-to-read and understandable style.

Each book includes:
- detailed tables of contents and indexes, alphabetical and numerical tables with numerous cross references.
- register setup going into routines.
- register results coming out of routines.
- detailed explanations on all routines, structures, codes, etc.; and not just a few ambiguous comments.
- Numerous summaries for ease of book use.

Special . . . all three $49.95

SORT/CMD
- multi-key (to 14 fields); Multi-tag (to 14 more fields)
- sort any combination of variable type
- integrates easily into any existing program

$19.95 each *Disk or non-disk; Model I or III; 32K or 48K*

PATCHES
to allow operation of many Model I programs on the Model III:
- LM OFFSET/CMD, Superzap/CMD[2]
- RSM 2D/CMD 10K, 32K, 48K[3]
- Big Five Software Games (all games)[4]

$5.95 each Many more . . . send for brochure!

DISK DIRECTORY INDEX PROGRAM *(Model III only)*
- Maintain a master directory listing and file all diskettes.
- Index size limited only by diskette space available and number of diskettes on which the index will be maintained.
- Summary by file name, diskette, or extension.

Direct/BAS $14.95
Direct/CMD $24.95 *(enhanced, more powerful, faster)*

Send check or money order (Kansas residents add 3% sales tax) to: **Twenty-First Century Software**
1607 North Cochran Hutchinson, Kansas 67501
*Call **316 663-1047** for additional information.*

TWENTY FIRST CENTURY SOFTWARE

18

Registered trademarks or products of [1]Tandy Corp., [2]Apparat Inc., [3]Small Systems Software, [4]Big Five Software Company

See List of Advertisers on page 354

Why most compilers fail and what can be done about it.

A Macroprocessor for Basic—Part III

J. Alan Olmstead
J. Olmstead Financial Engineering Systems
3843 West St., Moritz Lane
Phoenix, AZ 85023

Computer optimization efforts cannot stop at producing Basic programs faster and with less programming manpower. Programming time may in fact be the least expensive portion of computer operations. Programming is a capital investment rather than an operating expense—the dollar value representing programming time is spent only once; modifications and extensions to the original program are merely add-ons to that original effort. But the program is run continuously; any inefficiencies built into it, or built into the computer or programming language used, are expenses incurred repeatedly for the program's life.

Basic is the easiest high-level language to learn and to use ever designed. But it exacts a high price for that convenience every time a Basic program runs.

Basic Inefficiencies

The first thing to remember about Basic is that it doesn't exist. Basic is a syntactical concept that is applied in as many different ways as the various writers of Basic interpreter programs. Whether fixed in ROM or loaded from disk, the Basic *interpreter* runs the computer, never the Basic program. Accordingly, one must always think and speak in terms of a particular interpreter. Several of the most common contemporary interpreters include the Northstar Horizon, IBM's late 5100 series, and Microsoft's TRS-80 Model I Basic, licensed to Radio Shack.

Every interpreter has its own characteristics and bugs, but all share underlying similarities. Individual characteristics do not promote an understanding of the problems involved in optimizing a computer programmed in Basic.

Basic programs are not really programs at all, but rather, data files. A real computer program takes control of the computer's registers and is responsible for their successful manipulation. Basic programs have no way to take that control, which is restricted to interpreters, assemblers and compilers.

The Interpreter

A Basic program, considered as a data file, is a variable-length, record-sequential, memory-resident data file. In any sequential file, the first record is read first, the second is read second, and so on. The first line in a Basic program is always the start of the program, whereas an Assembly language program can begin execution at any chosen address.

In addition to the Basic command line, there must be next-line pointers to help distinguish between the logical and the physical next line. Table 1 illustrates the difference between the logical next line versus the physical next line pointers.

In Method 1, subfield a is the line number of the Basic command line, compressed into machine code and composing two hexadecimal bytes of memory. Subfield b is the Basic command line proper, with the Basic command words compressed into single-byte codes between decimal 128-255 for space economy. Subfield c is the end-of-record mark. Subfield d is the next-record-location pointer.

In Method 2, subfield a is the line number, but subfield b is the length of the current Basic command line. It is shown here as two bytes, but is frequently restricted to one byte and, therefore, a maximum line length of 255 characters. Fields c and d are the same as in Method 1.

When the program is first loaded into memory, the lines are usually physically and logically sequential from low memory to high memory. But, if a line should be edited and accordingly made shorter or longer, it may no longer fit. If a new line is inserted, it definitely will not fit—interpreters do not shift the rest of the program upwards to make room. Instead, they append the new line onto the end of the program and adjust the next-record pointers of the previous and next lines to reflect the change. In some Basics, a periodic recompression of memory may take place. In others, when the end of memory is reached the interpreter flashes an ?OM error message. However, after saving and reloading the program, programming may continue for many more new lines.

After editing, the physical lines are no longer logically sequential. In some Basics, the lines may not be physically reordered even after saving the program, unless the ASCII mode is requested, but many systems do not offer the ASCII option. The program remains a mixed-up hodge-podge to be sorted by the interpreter.

The physical and logical sequence is the same at load time for most lines; those containing GOTO, GOSUB, Then, Else, Resume and Return are exceptions, however. When the interpreter encounters these commands, it must abandon the next-record pointers and use some other method of locating the logical next record. Two methods used to accomplish this are the sequential and the "tree-structured" search for the named line number.

The sequential search is the most unbelievable because it causes the interpreter program to begin at the physical beginning of the program and search in physical record sequence until the named record is found. Then program execution begins again. Even at high speeds, this method imposes such a large time penalty on the user that he is forced to revert to bad programming practices for economy.

The use of independent subroutines called by GOSUB commands saves much valuable space and programming time during program debugging. Since the average commercial applications program will contain between 30-90 such subroutines, the sequential subroutine search procedure can reduce program execution speed by 75 percent.

Incredible as it may seem, the TRS-80 Model I disk drives, with a transfer rate of only 12,500 bytes per second, are faster than main memory, which is measured in microseconds. This is true only under the Basic interpreter, and particularly during

"Basic is the easiest high-level language to learn and to use ever designed."

execution of programs containing large numbers of subroutines.

A "tree-structured" search for the first line of a subroutine is somewhat faster, but not enough to make the point inapplicable. For example, if the largest magnitude of line number was 9999, a table of addresses of the even-thousand line numbers would take only 20 bytes. If each of those line pointers in turn pointed to a table of addresses of the even one-hundred line numbers, the total size of the table would still be only 220 bytes, but the interpreter would always be within an average of five lines of the desired line number (assuming a line number increment of ten) after two table look-ups. The tree-structured search technique is vastly superior to the straight sequential search, but it is not fast enough to make a significant difference when attempting to optimize the computer's performance overall.

These two illustrations identify significant sources of inefficiency in Basic programs. The need for record management induces an inefficiency that accumulates during program execution. Since subroutines are executed repeatedly, the next-record lookup inefficiency tends to accumulate in geometric magnitudes.

A compiler locates a source language record and generates from it the machine executable code. If the resulting program passes through a given line any number of times, the line is merely executed; compilation is performed only once. The interpreter is made up of two general groups of subroutines: the "parsing" logic and the execution logic. The term *parsing* refers to the structural analysis of a series of symbols that collectively convey a complex concept. The Basic parser breaks the command line down to determine which of its precoded subroutines should be executed in what order. This process is quite time consuming; each line must be reparsed every time it is executed.

Not only math applications suffer these inefficiencies; they are quite common in any byte-manipulating applications, such as word processing.

How Data Is Stored

An Assembly language programmer usually maps out a section of memory for data. During program execution, the data remains in that location. Most Basic interpreters follow this pattern. Both explicit and implicit dimensioning cause a section of memory to be mapped out. For this reason, an implicitly dimensioned variable cannot be dimensioned again to a longer length.

When a Basic program names data, it does so by assigning an ASCII name, such as A$ or A. When the data name is encountered in the Basic program, the interpreter consults a data name table for a match and inserts new names if none is found. Along with the name, the interpreter stores the length of the data and the location in memory where storage space has been found. If the data name is merely being DIMensioned, all the data-name table information is set up except the data address, because there is no data to address until an assignment (A$ = "DATA") is made.

Numeric data is handled differently; for integer data, for example, the address entry in the table is the same number of bytes as would be needed to store the data itself. The functions are interchanged and the address portion of the table is initialized to

"The need for record management induces an inefficiency that accumulates during program execution."

zeros, which means that the data name is equal to the number zero. When a value is assigned ("A = 23"), the address portion of the table is changed to reflect the value, not an address. When a string data assignment is made, memory space is found, the data is stored, and the address bytes are updated to reflect the address of the data.

A particularly good design job is reflected in Microsoft's use of the space taken up by the data literal as the same space for the data. Thus, the only result of the command A$ = "DATA" is that the table entry of A$ is updated to reflect the location of the D in DATA. The method is highly efficient for small memory-sized computers.

Time-dependent functions if possible at all are possible only by rigging the TRS-80 with cute tricks to get by.

Typists must constantly watch the screen to ascertain that their keystrokes are not being ignored by the interpreter's arbitrary decision to value space over time. If control is ever returned to the interpreter, the communications service routine will usually not get it back in time to service the I/O port.

The TRS-80 Model I computer has outgrown the Basic interpreter.

Today nearly every conceivable peripheral device is being interfaced to the world's most successful computer. In most cases, however, special Assembly language software must be acquired to access the peripheral through the USR function.

If the trend away from Basic (while pretending to be programming in Basic) continues, sooner or later someone must ask, why bother with it at all? There is one very good reason: For commercial and business-oriented applications, it is a great language to use, delivering cost-per-line economy not possible with any other language. To improve the power of Basic while remaining as simple as Basic, we need a Basic compiler: a program that will accept Basic—even interpreter Basic—as a source, and create a pure Assembly language program from it.

Why Basic Compilers Fail

Even a working Basic compiler fails if it restricts the programmer's capabilities to those functions that are possible only in Basic. There are specific reasons for compiler failure when their functions are restricted to translation of Basic. All the characteristics of Basic which we have identified as inefficient derived because a Basic program is a data file rather than a real program; and that fact continues to dominate any version of Basic interpreter. To the extent that a compiler attempts to emulate an interpreter, the chances of successful operation of the compiled output are so slim as to warrant the prediction that the compiler must, inevitably, fail.

What is meant by the idea of emulating a compiler, versus being a real compiler? Output from a compiler is a computer program in every sense; it takes control of the computer's registers, and the responsibility for their successful manipulation. The registers may contain only valid memory addresses, modifiers of memory addresses, and one or two bytes of data currently being manipulated.

When an Assembly language program says "GOTO 1000," it abruptly alters the contents of the program counter, which is the register pointing to the address of the next operation command to be executed. When the program says 'A$ = "DATA" ', it is identifying the left-most byte addresses of the sending and receiving fields in preparation for an extended move. These are precise commands, meaning exactly correct, every time.

That a compiler works at all has always been a small miracle. The reason the Basic interpreter is able to accomplish program execution tasks from such general programming statements is that it enjoys the advantage of having both program instruction and user's data at the same time. If program instruction is not precise, the data can be examined to fill in the gaps. By contrast, compilers never have the user's data to provide clarity. Given the precision of the Assembly language program, detailed to

"In the context of compilers, optimized code is a term meaning the opposite of what it says."

the last bit of the last byte, the chance of failure is a statistical certainty if the compiler attempts to treat the Basic program in the same manner as did the interpreter.

The Basic interpreter language available to the programmer contains no command words or symbols for conveying more precise information. The compiler can do only two things: It can optimize its code, or it can simply take a stab and hope everything comes out all right.

In the context of compilers, optimized code is a term meaning the opposite of what it says. The optimizing compiler throws in everything it can get hold of in order to be prepared for the coming onslaught of user's data. It is not unusual for a compiler manual to state minimum equipment configuration requirements of at least 128K.

Some of this kind of optimization is inevitable in all compilers (the Jofes MetaBasic Compiler also has its share of it). As long as the programmer's source code remains insufficiently precise, the only alternative is to include partial coding and hope it will be adequate.

Two examples of this include numeric and string data. With an interpreter, you can define two integers, then divide them, and still end up with an integer result. The interpreter does not tell the programmer that it sneaked in an intermediate stage of single or double-precision format in order to complete the division stage of the computation successfully. After compilation, the program fails.

It fails because the compiler emulates the interpreter on the assumption that the evaluation of the arguments would take place when the data was presented. The compiler did not maintain a table of data types and consult that table for error-checking when the division command was encountered. As a result, the wrong code was included in the compiled output.

With string data, the Basic interpreter program could perform input and output operations to and from data arrays at will. After compilation, the results are "unspecified," meaning, "Gee, guys, we don't know what it's going to do!" When the Print or LPRINT command is encountered by the compiler, it calls the wrong kind of code, to handle the transfer of array data instead of string data.

These problems are not insurmountable, but it is unlikely that the many ways to correct the problem will be contained in a single compiler program within a 48K user system, nor will they be discovered without extensive field testing.

How could a redesign of compilers proceed? The first step is to recognize the inherent limitations of any compiler and not try to be all things to all people. More space in the compiler should be spent performing diagnostic analysis and using listing notes to indicate where potential problems may be shaping up. This space can be created by breaking the compiler up into several programs, each segment handling analysis of the same kind, then automatically chaining to the next segment without operator intervention.

The second step should deal with the problem of lack of precision by giving the programmer the necessary tools by modifying the Basic language itself. A small example would be initializing the number of decimal points in precision variables: "A DEFBCD 3(10,10)" means "define a numeric variable called "A" in binary coded decimal format, but initialize it with three decimal positions; and make it an array of 10 vertical and 10 horizontal elements."

MetaBasic

A more precise (and vastly more flexible) Basic is what the Jofes compiler calls MetaBasic. It is an intermediary into Assembly language, a half-way house for programmers.

First, it may be used in the original program instead of conventional Basic. Some of the commands are the same (GOTO, GOSUB), some are similar (Open, Close) and some are completely different. For example, the MetaBasic command 'FLOOD A$ WITH 255 " " is immediately obvious as an alternative to 'A$ = STRING$(255," ")'.

Second, in programs that already exist for which manual rewriting would be a labor-intensive task, the MetaBasic Precompiler converts interpreter Basic into MetaBasic automatically. In the process, it shows the default conditions it added as express arguments. By reading the printed listing, which corresponds on a line-number-tag basis to the original, the programmer can see immediately where the compiler may have gone wrong in its interpretation of the intended result.

Finally, for the growing body of programmers who use Basic for its economy and not because it is their only language, MetaBasic will accept the full line of machine language commands.

Compilers fail because they try to do too much and end up delivering too little. Perhaps it would be more accurate to say that they fail because they try to do too much of the wrong kind of functions. By changing priorities on what a compiler should be expected to do, a Basic compiler can in fact deliver far more than was previously anticipated. ■

Next month: more on MetaBasic.

TRS-80® DISCOUNT SOFTWARE TRS-80®

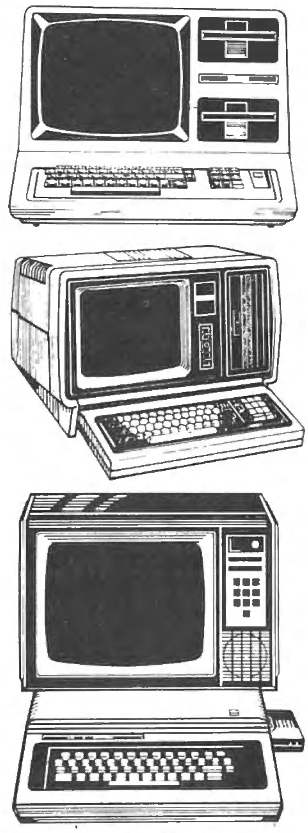
MODEL III SOFTWARE

26-1563 Scripsit I & III.........$79.00
26-1566 Visicalc I............$83.00
26-1567 Visicalc III...........$83.00
26-1552 General Ledger......$90.00
26-1553 Inventory Ctl.........$90.00
26-1556 Payroll.............$180.00
26-1555 Accts. Receivable....$135.00
26-1554 Accts. Payable......$135.00
26-1558 Bus. Mail List.........$90.00
26-2201 Fortran I.............$90.00
26-1562 Profile..............$72.00

Plus many more. . . . .

COLOR SOFTWARE

26-3050 Chess...............$36.00
26-3051 Quaser..............$36.00
26-3052 Pinball..............$27.00
26-3053 Football.............$36.00
26-3055 Checkers............$27.00
26-3057 Dino Wars...........$36.00
26-3059 Backgammon.......$27.00
26-3152 Typing Tutor.........$27.00



MODEL II SOFTWARE

26-4530 Scripsit II...........$265.00
26-4511 Visicalc II...........$265.00
26-4512 Profile II............$162.00
26-4501 General Ledger.....$180.00
26-4502 Inventory Mgmt.....$180.00
26-4503 Payroll.............$360.00
26-4504 Accts Receivable....$270.00
26-4505 Accts Payable.......$270.00
26-4507 Mail List II..........$108.00
26-4701 Fortran.............$270.00
26-4703 Cobol..............$270.00

Plus many more. . . . .

POCKET SOFTWARE

26-3510 Real Estate...........$22.50
26-3511 Civil Eng.............$22.50
26-3513 Aviation.............$22.50
26-3516 Bus. Statistics........$18.00
26-3517 Bus. Financial........$18.00
26-3518 Per. Financial........$18.00
26-3514 Math Drill............$13.50
26-3515 Games..............$13.50

ORDER TOLL FREE 1-800-841-0860

CALL US - SAVE MONEY

We carry the full line of TRS-80 Computers. All TRS-80 software, furniture, and accessories sold at discount. We stock most items to assure you fast delivery and save you money.

WRITE FOR FREE CATALOG

MICRO MANAGEMENT SYSTEMS, INC. ✓72

No Taxes on Out Of State Shipments

Immediate Shipment From Stock on Most Items

DOWNTOWN PLAZA SHOPPING CENTER
DEPT. # 4A
115 C. SECOND AVE. S.W.
CAIRO, GEORGIA 31728
(912) 377-7120 Ga. Phone No.

TRS-80 is registered trademark of the Tandy Corp.

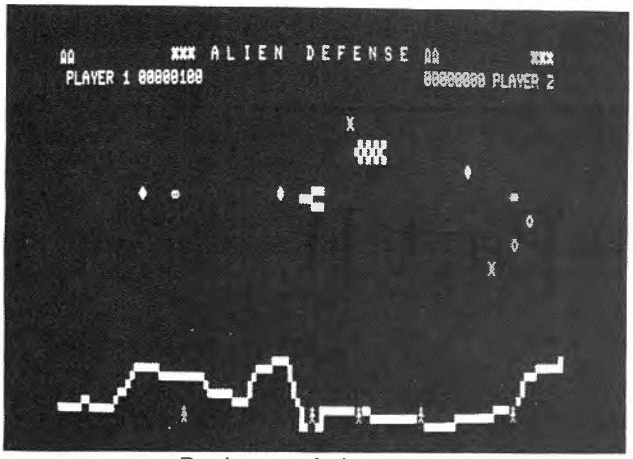
Thousands of Satisfied Customers

VISA And Mastercharge Welcomed

✓See List of Advertisers on page 354

ALIEN DEFENSE

MOD III with Sound



By Larry Ashmun

IT'S HERE!

"It's more than just another machine language arcade game."

IT'S AN EXPERIENCE

Super Fast
1 or 2 Player
MOD III ONLY

Tape version **$19.95**
Disk Version . . . . . **Coming Soon**

SEE OUR EXPANDED AD
IN 80 US JOURNAL
for more product information.

Larry Ashmun Is Back . . .
and Brings You . . .

SUPER VADERS

With Sound - Mod I & Mod III
©1981 Soft Sector Marketing, Inc.

The third in the evolution of the most popular space invaders games for Tandy machines. Now for 1 or 2 players, 10 levels of play.

Cassette (goes to disk) **$19.95**

Ask for upgrade information for TRS-Super Invaders or Invaders-Plus.

MOD III ROM COMMENTED

©1981 Soft Sector Marketing, Inc.

Only $22.50
+$2.50 Shipping & Handling

Not just a rehash of old information, but detailed comments on the ROMS in the latest machine from Tandy.

Now in its 3rd printing!
Over 150 Pages!

TAPE COPY 2

©1981 Soft Sector Marketing, Inc.

This program will load most any TRS-80 500 Baud system tape (standard) Mod I speed) and load it into memory and save it at either 500 or 1500 Baud on the Mod III. NO KNOWLEDGE OF MACHINE LANGUAGE NEEDED. Now it gives you a way to back up a machine language program that loads at the lower speed and makes cassette loading into your new Mod III a much faster and more reliable process. Works with Mod I* & Mod III.

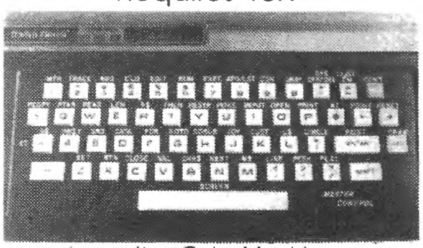
Only **$14.95**

*Mod I loads and saves at 500 Baud only

MASTER CONTROL

For the Color Machine

©1981 Soft Sector Marketing, Inc.
Requires 16K



Ask about our other Color Machine programs!

FEATURES:

1. 50 preprogrammed command keys, standard and extended commands.
2. Direct control of motor, trace, and audio.
3. Relocatable machine code.
4. Automatic line numbering, starting point and increment alterable.
5. Programmable custom key.
6. Direct Run button.
7. Keyboard overlay for easy program use.
8. Easy entry of entire commands into the computer and display.
9. Full instruction manual.

Cassette **$24.95**

Outside Michigan
Order Line Only
800-521-6504

SSM SOFT SECTOR MARKETING, INCORPORATED
6250 Middlebelt
Garden City, Michigan 48135

Questions &
Michigan Orders
313-425-4020

✓434

Which one is best for you?

All About Sorts—Part II

Len Gorney
Box 91 R.D. 5
Clarks Summit, PA 18411

The Shell sort technique was initially developed by Shell and later expanded by R.M. Frank and R.B. Lazarus (see bibliography). This is an interchange sorting technique where several passes through the original list of items reduces it to a more nearly sorted list. The final pass usually reverts to that of a bubble sort, since it is the bubble sort which can most quickly handle nearly sorted items.

A particularly vexing problem connected with the Shell sort involves the parameter which determines the distance (D) between items to be compared. In our previous sorting examples, we always compared adjacent items or a single item with every other item in the list.

The Shell sort, on the other hand, chooses some distance (D) between items, and these two items are then compared. To avoid becoming involved with the mathematics, we will simply calculate the initial value of the distance parameter as being equal to one-half the number of items to be sorted; that is, D = N/2. (Lorin offers a readable examination of the Shell sort, as well as a method for calculating the distance parameter.)

To accurately determine the maximum, average and minimum number of comparisons and exchanges involved in a Shell sort, a lot of information must be available regarding the distance parameter value. It is this single parameter which determines the overall effectiveness of the Shell sort.

Program Listing 1 and Table 1 are the programmed routine and sample outputs for our Shell sort routine. Tracing the items as they are sorted may clear any misconceptions regarding the actual operation of this technique.

Quick Sort

One reason for the slowness of the bubble sort is the many comparisons and exchanges made between adjacent items. To speed up the sorting process, the distance between compared items should be increased to a value whereby comparisons and exchanges are made between items some distance apart. The quick sort procedure uses this technique to offer a speedier sorting algorithm. The distance parameter (D) is also used by the Shell sort.

The quick sort routine begins with a simple operation and proceeds by using stacks to sort the items. Let us begin with the selection of what is referred to as the pivot value of the quick sort sorting technique.

The quick sort routine initially estimates the median value of the items to be sorted. Based on the median value, the list is broken into two groups; one group contains those keys which are less than the median value while the other group consists of those keys which are greater than the median value. The median value is placed between two groups.

For example, the following list is initially broken into two groups by taking the first key value as the median of the keys contained in the list. This list—5 2 7 8 1 3 7—will be used as our list of items to be sorted. The assumed median value will be the value given as the first key in the list, the value five. The original list is now broken into two groups: the left group containing keys less than the assumed median; and the right group consisting of keys greater than the median, 2 1 3 and 7 8 7.

Inserting the assumed median value between the two groups gives us the following list: 2 1 3 5 7 8 7. Each group is now sorted separately by the quick sort technique.

The method used by the quick sort to make a choice of a median value runs the gamut from a random number generator, calculating the actual median value of the keys, estimating the median of the list, or simply choosing the first (or any) key as the assumed median value of the keys. We will employ this last method as it is the simplest.

After a median or pivot value is chosen, the remaining keys are broken into two groups. Keys less than the pivot are placed in locations beginning with the first location of this list, while keys greater than the median are placed in locations beginning with the Nth location. At the end of this initial distribution phase, the last remaining position is the location for the chosen pivot value. The pivot value chosen determines the subsequent efficiency of the algorithm. If the worst case is chosen (the smallest or the largest key in the list), the sizes of the two groups would be equal to zero and N − 1. If the pivot value is, in fact, the actual median value of the keys in the list, the two

ORIGINAL ITEMS	8 0 9 4 3 5 7 2 6 1
PASS #1 ITEMS	5 0 9 4 3 8 7 2 6 1
PASS #2 ITEMS	5 0 9 4 3 8 7 2 6 1
PASS #3 ITEMS	5 0 2 4 3 8 7 9 6 1
PASS #4 ITEMS	5 0 2 4 3 8 7 9 6 1
PASS #5 ITEMS	5 0 2 4 1 8 7 9 6 3
PASS #6 ITEMS	2 0 5 4 1 8 7 9 6 3
PASS #7 ITEMS	2 0 5 4 1 8 7 9 6 3
PASS #8 ITEMS	1 0 2 4 5 8 7 9 6 3
PASS #9 ITEMS	1 0 2 4 5 8 7 9 6 3
PASS #10 ITEMS	1 0 2 4 5 8 7 9 6 3
PASS #11 ITEMS	1 0 2 4 5 8 7 9 6 3
PASS #12 ITEMS	1 0 2 4 5 8 6 9 7 3
PASS #13 ITEMS	1 0 2 3 5 4 6 8 7 9
PASS #14 ITEMS	0 1 2 3 5 4 6 8 7 9
PASS #15 ITEMS	0 1 2 3 5 4 6 8 7 9
PASS #16 ITEMS	0 1 2 3 5 4 6 8 7 9
PASS #17 ITEMS	0 1 2 3 5 4 6 8 7 9
PASS #18 ITEMS	0 1 2 3 4 5 6 8 7 9
PASS #19 ITEMS	0 1 2 3 4 5 6 8 7 9
PASS #20 ITEMS	0 1 2 3 4 5 6 8 7 9
PASS #21 ITEMS	0 1 2 3 4 5 6 7 8 9
PASS #22 ITEMS	0 1 2 3 4 5 6 7 8 9
SORTED ITEMS	0 1 2 3 4 5 6 7 8 9

COMPARISONS = 29 EXCHANGES = 13

Table 1. Shell Sort Sample Output

ORIGINAL ITEMS	8 0 9 4 3 5 7 2 6 1
PASS #1 ITEMS	8 0 9 4 3 5 7 2 6 1
PASS #2 ITEMS	1 0 6 4 3 5 7 2 8 9
PASS #3 ITEMS	0 1 6 4 3 5 7 2 8 9
PASS #4 ITEMS	0 1 2 4 3 5 6 7 8 9
PASS #5 ITEMS	0 1 2 4 3 5 6 7 8 9
PASS #6 ITEMS	0 1 2 3 4 5 6 7 8 9
SORTED ITEMS	0 1 2 3 4 5 6 7 8 9

COMPARISONS = 33 EXCHANGES = 12

Table 2. Quick Sort Sample Output

"When the size of the items becomes prohibitively large . . . rather than physically move the entire item to a sorted list, the linked list sorts provide pointers to the sorted keys of the items."

groups will be equal in size. Our choice should cause the size of the groups to fall somewhere between these two extremes.

After completing the initial distribution phase, a stack structure saves (pushes) a group of keys as subsequent sorting phases are processed. Eventually these stacked groups are unstacked (POPped) and processed accordingly. When the stack is finally emptied of all the keys, the list is in its final (sorted) condition.

Program Listing 2 is the programmed routine which accomplishes the quick sort; Table 2 represents sample output as the process sorts the items.

Linked List Sorts

Our previous sorting techniques arranged the items in the list by using simple keys and moving the entire item. In the examples, the items consisted solely of the key. This simple assumption cannot always be made, however. For example, the usual key in a mailing label sort is often the zip code: the keys (zip codes) are compared and the entire item (name, address, city, state and zip code) must be moved into its proper location in the sorted list. Obviously, moving this much data takes quite a bit of time.

When large items must be moved, another efficiency measure is the time required not only to sort the items but also to move them. When the size of the items becomes prohibitively large, a more efficient sorting technique is usually required to save time.

Rather than physically move the entire item to a sorted list, the linked list sorts provide pointers to the sorted keys of the items. To produce a sorted output list, the programmed routine (usually the mainline program) simply uses an array which contains pointers to the keys of interest. These pointers are sorted so they point to the respective keys of the items.

```
1000 DIM A(10)
1010 N = 10
1020 FOR X = 1 TO 4
1030   CLS:
       PRINT"ORIGINAL ITEMS    ";
1040   FOR K = 1 TO N
1050     READ A(K):
         PRINT A(K);
1060   NEXT K
1070   GOSUB 1170
1080   C = 0:
       E = 0:
       GOSUB 1180
1090  PRINT:
       PRINT"SORTED ITEMS      ";
1100   FOR K = 1 TO N
1110     PRINT A(K);
1120   NEXT K
1130   PRINT:
       PRINT"COMPARISONS =";C;"EXCHANGES =";E:
       GOSUB 1170
1140 NEXT X
1150 END
1160 DATA 0, 1, 2, 3, 4, 5, 6, 7, 8, 9:
     DATA 9, 8, 7, 6, 5, 4, 3, 2, 1, 0:
     DATA 8, 0, 9, 4, 3, 5, 7, 2, 6, 1:
     DATA -7, 3, -9, -7, 0, 6, 6, 4, 5, -1
1170 FOR T = 1 TO 1000:
     NEXT T:
     RETURN
1180 '
        SHELLSORT.

1190 '
        DISTANCE PARAMETER (D) DETERMINATION.

1200 D = INT ( N / 2 ):
     PP = 0
1210 FOR P = 1 TO N-D
1220 '
        COMPARE THE J(TH) ITEM AGAINST THE ITEM D DISTANCE AWAY.

1230   J = P:
       EE = 0
1240   C = C + 1:
       IF A(J) <= A(J+D) THEN 1290
1200 D = INT ( N / 2 ):
     PP = 0
1210 FOR P = 1 TO N-D
1220 '
        COMPARE THE J(TH) ITEM AGAINST THE ITEM D DISTANCE AWAY.

1230   J = P:
       EE = 0
1240   C = C + 1:
       IF A(J) <= A(J+D) THEN 1290
1250 '
        EXCHANGE ITEMS.

1260     T = A(J):
         A(J) = A(J+D):
         A(J+D) = T:
         J = J - D:
         E = E + 1:
         EE = E
1270 '
        LOOP BACK TO COMPARE AS LONG AS POINTER J IS VALID.
1280     IF J > 0 THEN 1240
1290 '
        TAKE SNAPSHOTS.

1300       PP = PP + 1:
           PRINT:
           PRINT"PASS #";PP;"ITEMS    ";
1310       FOR KK = 1 TO N
1320         PRINT A(KK);
1330       NEXT KK
1340       GOSUB 1170
1350 NEXT P
1360 D = INT ( D / 2 ):
     IF EE = 0 THEN 1380
1370 IF D > 0 THEN 1210
1380 RETURN
1390 END
```

Listing 1. Shell Sort Routine

✓See List of Advertisers on page 354

"How do we get from the initial condition of the pointer array to its final condition? The answer involves a bit of magic as well as a good deal of coding."

List of items to be sorted	9	−7	3	0	−7
Pointer array	1	2	3	4	5

Fig. 1. Linked List Sort Item List

List of items to be sorted	9	−7	3	0	−7
Pointer array	2	5	4	3	1

Fig. 2. Linked List Sort Final Array

Assume that Fig. 1 represents the list of items we wish to sort and the array which holds the pointers to this list of items. (We will use a short list of items and short item length to demonstate the mechanics of a linked list sort.) The original list of items will remain in the same order throughout the sorting process. The array which holds the pointers will change from its initial condition to its final (sorted) condition during the sort. The values contained in the pointer array represent the subscript into the list of items to be sorted.

In its initial condition, the pointer array assumes that the list of items to be sorted is in sorted order. In other words, P(1) = 1, which means that the value associated with P(1) points to the subscript which is the smallest item in the list: If P(1) = 1, then A(P(1)) = 9. The second value in the pointer array, P(2) = 2, points to the next smallest item in the list to be sorted; that is, A(P(2)) = −7. The values in the pointer array do not represent a sorted arrangement of items to be sorted. The initial condition of the pointer array simply assumes that the list of items to be sorted is sorted.

Fig. 2 represents the final condition of the pointer array and list of items to be sorted. Using the pointer array value as the subscript value into the array of items to be sorted, we may make the following judgement:

A(P(1)) = A(2) = −7
A(P(2)) = A(5) = −7
A(P(3)) = A(4) = 0
A(P(4)) = A(3) = 3
A(P(5)) = A(1) = 9

How do we get from the initial condition of the pointer array to its final condition? The answer involves a bit of magic as well as a good deal of coding.

Our primary examination of linked list sorts will revolve about two techniques. The first uses the technique just described, while the second uses a somewhat different approach.

Vector Sort

The first linked list sort is called a vector sort, and uses the technique of providing pointers to the items in the list to be sorted. Program Listing 3 consists of a mainline program and the vector sort subroutine which produces the sample output in Table

```
1000 DIM A(10), MI(10), MX(10)
1010 N = 10
1020 FOR X = 1 TO 4
1030   CLS:
       PRINT"ORIGINAL ITEMS    ";
1040   FOR K = 1 TO N
1050     READ A(K):
         PRINT A(K);:
         MI(K) = 0:
         MX(K) = 0
1060   NEXT K
1070   GOSUB 1170:
       PRINT
1080   C = 0:
       E = 0:
       GOSUB 1180
1090   PRINT:
       PRINT"SORTED ITEMS      ";
1100   FOR K = 1 TO N
1110     PRINT A(K);
1120   NEXT K
1130   PRINT:
       PRINT"COMPARISONS =";C;"EXCHANGES =";E:
       GOSUB 1170
1140 NEXT X
1150 END
1160 DATA 0, 1, 2, 3, 4, 5, 6, 7, 8, 9:

     DATA 9, 8, 7, 6, 5, 4, 3, 2, 1, 0:
     DATA 8, 0, 9, 4, 3, 5, 7, 2, 6, 1:
     DATA -7, 3, -9, -7, 0, 6, 6, 4, 5, -1
1170 FOR T = 1 TO 1000:
     NEXT T:
     RETURN
1180 '
        QUICKSORT.

1190 BP = N:
     TP = 1:
     X = 0:
     Y = 0:
     P = 0
1200 GOSUB 1510:
     GOSUB 1520
1210 '
        TAKE SNAPSHOTS.

1220 P = P + 1
1230 PRINT"PASS #";P;"ITEMS   ";
1240 FOR KK = 1 TO N
1250  PRINT A(KK);
1260 NEXT KK
1270 GOSUB 1170
1280 PRINT
1290 IF TP >= BP THEN 1480
1300   NM = A(TP)
1310   IF TP = BP THEN 1360
1320     C = C + 1
1330     IF A(BP) < NM THEN A(TP) = A(BP):
                            E = E + 1:
                            GOTO 1340
                       ELSE BP = BP -1:
                            GOTO 1310
1340     IF TP = BP THEN 1360
                    ELSE C = C + 1
1350       IF A(TP) > NM THEN A(BP) = A(TP):
                              E = E + 1:
                              GOTO 1310
                         ELSE TP = TP + 1:
                              GOTO 1340
1360 A(TP) = NM:
     E = E + 1:
     NB = TP - 1:
     NT = TP + 1
1370 GOSUB 1530:
     F = NB - TP:
     GOSUB 1540:
     NX = BP - NT
1380 IF F > NX THEN 1430
1390   M = TP:
       TP = NT
1400 IF TP < BP THEN GOSUB 1510:
                     GOSUB 1520
1410 BP = NB:
     TP = M
1420 IF TP >= BP THEN 1500
                 ELSE 1470
1430 LS = BP:

     BP = NB
1440 GOSUB 1510:
     GOSUB 1520
1450 TP = NT:
     BP = LS
1460 IF TP >= BP THEN 1500
1470   GOSUB 1510:
       GOSUB 1520:
       GOTO 1210
1480 GOSUB 1530:
     GOSUB 1540
1490 IF Y = 0 THEN RETURN
1500   GOSUB 1530:
       GOSUB 1540:
       GOTO 1200
1510 IF X = N THEN PRINT"WRONG WAY":
                   STOP
              ELSE X = X + 1:
                   MI(X) = TP:
                   RETURN
1520 IF Y = N THEN PRINT"WRONG WAY":
                   STOP
              ELSE Y = Y + 1:
                   MX(Y) = BP:
                   RETURN
1530 IF X = 0 THEN RETURN
              ELSE TP = MI(X):
                   X = X - 1:
                   RETURN
1540 IF Y = 0 THEN RETURN
              ELSE BP = MX(Y):
                   Y = Y - 1:
                   RETURN
1550 END
```

Listing 2. Quick Sort Routine

The merge sort requires more storage space than the vector sort, but it takes less time to execute.

3. The vector sort requires extra storage for the pointer array (P) equal to the number of items in the list to be sorted.

The vector sort proceeds generally in the same manner as our previously examined sorting techniques, the major difference being that no exchanges are made within the list of items to be sorted. The pointer array is adjusted as the positions of the keys are determined relative to the pointer array. Table 3 can be traced to see the operation of this routine.

Merge Sort

The second linked list sort is a type of merge sort which requires more extra storage than the vector sort. This disadvantage is somewhat diluted in that the merge sort takes less time to execute than the vector sort. The determining variable in the selection of either of these two linked list sorts is time versus storage. It is up to the user to decide which linked list sort to use for the particular application at hand.

Fig. 3 shows the final condition of the variable used in the merge sort. Follow the pointers to determine the order of the items to be sorted.

The variable S2 points to the subscript of the items to be sorted where the smallest items resides. According to Fig. 9, the variable S2 is equal to 2, therefore, A(S2) = A(2) = −7. To determine the next sorted item, use the pointer array (P) in the following manner: Since A(S2) = A(2), the next smallest item will be at P(2). The subscript of the array A is used to gain access to the value stored in the adjacent pointer array P. P(2) = 5, therefore, A(P(2)) = A(5) = −7, the next smallest of the items to be sorted. Subsequent items can be found by the following:

```
A(S2)   = A(2) = -7
A(P(2)) = A(5) = -7
A(P(5)) = A(4) =  0
A(P(4)) = A(3) =  3
A(P(3)) = A(1) =  9
```

The last condition (P(1) = 0) means that the list of items to be sorted is exhausted, since no item resides in A(P(1)) = A(0).

Listing 4 is mainline program and the merge sort subroutine for the second linked list sort.

We have only scratched the surface of sorting techniques; use the bibliography if you want to gain a deeper understanding of these and other sorting routines available to the microcomputerist.

The routines in this article were successfully run on a TRS-80 Level II with 16K memory. The memory requirement is not important, since the sorting lists contain only 10 items. Modifying line numbers 1000 and 1010 will allow for larger lists to be sorted:

```
1000 DIM A(x), S(x), MI(x), MX(x)
1010 N = x
```

where x is equal to the size of the list of items to be sorted.

List of Items to be sorted	9	−7	3	0	−7
Pointer array	0	5	1	3	4
Variable Starter (S2) = 2					

Fig. 3. Merge Sort Array Final Condition

```
ORIGINAL ITEMS   8 0 9 4 3 5 7 2 6 1
PASS #1 ITEMS    8 0 9 4 3 5 7 2 6 1
PASS #2 ITEMS    0 8 9 4 3 5 7 2 6 1
PASS #3 ITEMS    0 4 8 9 3 5 7 2 6 1
PASS #4 ITEMS    0 3 4 8 9 5 7 2 6 1
PASS #5 ITEMS    0 3 4 5 8 9 7 2 6 1
PASS #6 ITEMS    0 3 4 5 7 8 9 2 6 1
PASS #7 ITEMS    0 2 3 4 5 7 8 9 6 1
PASS #8 ITEMS    0 2 3 4 5 6 7 8 9 1
PASS #9 ITEMS    0 1 2 3 4 5 6 7 8 9
SORTED ITEMS     0 1 2 3 4 5 6 7 8 9
COMPARISONS = 35 EXCHANGES = 27
```

Table 3. Vector Sort Sample Output

```
ORIGINAL ITEMS   8 0 9 4 3 5 7 2 6 1
PASS #2 ITEMS    0 8
PASS #3 ITEMS    0 8 9
PASS #4 ITEMS    0 4 8 9
PASS #5 ITEMS    0 3 4 8 9
PASS #6 ITEMS    0 3 4 5 8 9
PASS #7 ITEMS    0 3 4 5 7 8 9
PASS #8 ITEMS    0 2 3 4 5 7 8 9
PASS #9 ITEMS    0 2 3 4 5 6 7 8 9
PASS #10 ITEMS   0 1 2 3 4 5 6 7 8 9
SORTED ITEMS     0 1 2 3 4 5 6 7 8 9
COMPARISONS = 26 EXCHANGES = 9
```

Table 4. Merge Sort Output Sample

✓See List of Advertisers on page 354

"Since the mainline program presented uses only four sets of data . . . the program is quite useless except for illustrative purposes."

Line numbers 1020 through 1060 represent the common mainline program of the routines. Since the mainline program as presented uses only four sets of data as input to the items to be sorted, the program is quite useless except for illustrative purposes. The sorting subroutines are completely independent of any mainline program except for the passing of the following arguments:

- The array A which holds the items to be sorted.
- The variable N which is the number of items to be sorted.
- The variable C which is the number of comparisons.
- The variable E which is the number of exchanges.

Variables C and E may be deleted from the subroutines without affecting the operation of the subroutines. Thoroughly reading the code should provide you with the information required for modifications to the sorting subroutines. ■

Bibliography

Bell, D. A. *"The Principles of Sorting."* Computer Journal, *1, 1, 71-77.*

Frank, R. M. and R. B. Lazarus *"A Highspeed Sorting Procedure."* Communications of the ACM, *3, 1, 20-23.*

Hibbard, T. N. "An Empirical Study of Minimal-storage Sorting." Communications of the ACM, *6, 5, 206-313.*

Hoare, C. A. R. "Quicksort." Computer Journal, *5 (1962), 10-15.*

IBM Corporation, "Sorting Techniques." C20-1639 (1965).

Knuth, D. E. The Art of Computer Programming*: Volume 3,* Sorting and Searching. *Reading, Massachusetts: Addison-Wesley, 1973.*

Lorin, H. Sorting and Sort Systems*. Reading, Massachusetts: Addison-Wesley, 1975.*

Martin, W. A. "Sorting." ACM Computing Surveys, *3, 4, 147-174.*

Shell, D. L. "A High-speed Sorting Procedure." Communications of the ACM, *2, 7, 30-33.*

Van Emden, M. H. "Increasing the Efficiency of Quicksort." Communications of the ACM, *13, 9, 563-567.*


```
1000 DIM A(10), V(10)
1010 N = 10
1020 FOR X = 1 TO 4
1030   CLS:
       PRINT"ORIGINAL ITEMS   ";
1040   FOR K = 1 TO N
1050    READ A(K):
         PRINT A(K);:
         V(K) = K
1060   NEXT K
1070  GOSUB 1170
1080   C = 0:
       E = 0:
       GOSUB 1180
1090   PRINT:
       PRINT"SORTED ITEMS     ";
1100   FOR K = 1 TO N
1110    PRINT A(V(K));
1120   NEXT K
1130   PRINT:
       PRINT"COMPARISONS =";C;"EXCHANGES =";E:
       GOSUB 1170
1140 NEXT X
1150 END
1160 DATA 0, 1, 2, 3, 4, 5, 6, 7, 8, 9:
     DATA 9, 8, 7, 6, 5, 4, 3, 2, 1, 0:
     DATA 8, 0, 9, 4, 3, 5, 7, 2, 6, 1:
     DATA -7, 3, -9, -7, 0, 6, 6, 4, 5, -1
1170 FOR T = 1 TO 1000:
     NEXT T:
     RETURN
1180 '
      VECTOR LINKED LIST SORT.

1190 FOR P = 1 TO N-1
1200   J = P
1210 '
      COMPARE ITEMS VIA POINTER ARRAY (V).

1220   C = C + 1:
       IF A(V(J)) <= A(V(J+1)) THEN 1270
1230 '
      EXCHANGE ITEMS.

1240     T = V(J):
         V(J) = V(J+1):
         V(J+1) = T:
         E = E + 1:
         J = J - 1
1250 '
      LOOP BACK TO COMPARE AS LONG AS POINTER J IS VALID.

1260     IF J >< 0 THEN 1220
1270 '
      TAKE SNAPSHOTS.

1280     PRINT:
         PRINT"PASS #";P;"ITEMS   ";
1290     FOR KK = 1 TO N
1300       PRINT A(V(KK));
1310     NEXT KK
1320     GOSUB 1170
1330 NEXT P
1340 RETURN
1350 END
```

Listing 3. Vector Sort Program

```
1000 DIM A(10), V(10)
1010 N = 10
1020 FOR X = 1 TO 4
1030   CLS:
       PRINT"ORIGINAL ITEMS   ";
1040   FOR K = 1 TO N
1050    READ A(K):
         PRINT A(K);:
         V(K) = K
1060   NEXT K
1070  GOSUB 1170
1080   C = 0:
       E = 0:
       GOSUB 1180
1090   PRINT:
       PRINT"SORTED ITEMS     ";
1100  PRINT A(S2);:
       J = V(S2)
1110  FOR K = 2 TO N
1120    PRINT A(J);:
         J = V(J)
1130  NEXT K:
       PRINT:
       PRINT"COMPARISONS =";C;"EXCHANGES =";E:
       GOSUB 1170
1140 NEXT X
1150 END
1160 DATA 0, 1, 2, 3, 4, 5, 6, 7, 8, 9:
     DATA 9, 8, 7, 6, 5, 4, 3, 2, 1, 0:
     DATA 8, 0, 9, 4, 3, 5, 7, 2, 6, 1:
     DATA -7, 3, -9, -7, 0, 6, 6, 4, 5, -1
1170 FOR T = 1 TO 1000:
     NEXT T:
     RETURN
1180 '
      MERGE LINKED LIST SORT.

1190 S2 = 1:
     V(1) = 0
1200 FOR P = 2 TO N
1210   C = C + 1:
       IF A(P) <= A(S2) THEN V(P) = S2:
                             S2 = P:
                             PT = S2:
                             E = E + 1:
                             GOTO 1260
1220     ID = V(S2):
         TS = S2
1230     IF ID = 0 THEN 1250
1240       C = C + 1:
           IF A(P) > A (ID) THEN TS = ID:
                                 ID = V(ID):
                                 GOTO 1230
1250         V(TS) = P:
             V(P) = ID:
             E = E + 1
1260 '
      TAKE SNAPSHOTS.

1270 PRINT:
     PRINT"PASS #";P;"ITEMS   ";
1280 PRINT A(S2);
1290 J = V(S2)
1300 FOR KK = 2 TO N
1310   PRINT A(J);:
       J = V(J):
       IF J = 0 THEN 1330
1320 NEXT KK
1330 GOSUB 1170
1340 NEXT P
1350 RETURN
1360 END
```

Listing 4. Merge Sort Program

✓See List of Advertisers on page 354

A Basic program for rotating 3-D figures.

Rotation

Bruce Yellin
19 Marine Avenue
Brooklyn, NY 11209

I guess my obsession with three dimensional figures all started years ago when I was a computer science major in college. Being impressed with photographs of automobile engineers designing next year's cars on a graphic CRT, and dreaming of rotating the starship Enterprise on a video screen, I set out to develop a digital computer system capable of transforming three dimensional objects into two dimensional representations. As a frustrated mathematician in school, I knew that my dream could only be realized once I had a Ph.D. in mathematics and a $30,000 graphics computer at my disposal. But I decided to give it my best shot anyway!

My first attempt at displaying a 3-D object was on a UNIVAC 1106 computer using a hardcopy terminal and Fortran. Although my ideas seemed to work, a timeshared computer and continuous forms would hardly do the job.

When my own 16K Level II TRS-80 arrived, my obsession with 3-D returned. I had my Masters Degree in Computer Science and was using my LA 36 Decwriter II for hardcopy by utilizing an RS-232 serial output driver from Small Systems. I was "armed to the teeth" for the challenge. A month later, after many sleepless nights, I proudly proclaimed that 3-D lives in Brooklyn, N.Y.!

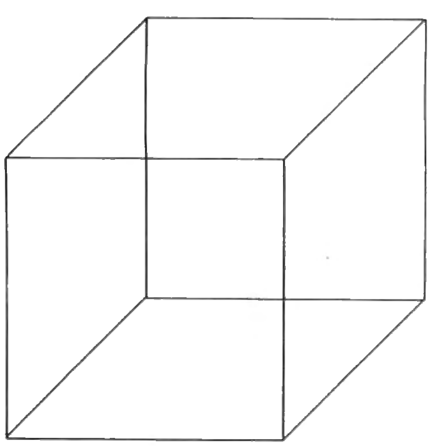


Fig. 1. The Cube as it is Traditionally Depicted

Why Cubes?

Now, let's get down to business. I've selected a cube because it is the simplest of regular figures to construct. It also has many interesting properties when rotated. All of us have probably drawn one when doodling. (see Fig. 1.) In its most recognizable form, you see six squares touching each other on all four sides. It is the easiest polyhedron to conceptualize and digitize compared to other members of the polyhedra family. (I will be explaining just how you digitize a cube a little later on.)

Try to imagine a cube coming to life. You can see three sides very plainly with the other three sides hidden from view (see Fig. 2). Turn the cube to the right and you should be visualizing Fig. 3. Keep turning and you have Fig. 4. And now if you look at the cube head-on, five sides are hidden and all you should be able to see is the remaining side (Fig. 5).

As you are aware, a cube is a three dimensional figure. Instead of squares and the X-Y rectangular coordinate system you are most familiar with, for example, in the dimensions of length and height, we must briefly discuss the added dimension of depth. These three dimensions represent a method of locating a given point in space (rather than in a plane using X-Y coordinates) in terms of its distances from three mutually related perpendicular lines or axes. In layman's language: (x,y,z) = (length, height, depth). The Z axis is responsible for giving a shape its realism (for example, cartoons generally lack realism because those figures are flat or X-Y images without the dimension of depth). A cube has depth (see Fig. 6).

Now, the X-Y-Z system may still sound very foreign to you. And you might be wondering what the word "digitize" really means. Let's try to tie together some loose ends.

In the digitization of a solid object, we try to assign values from the X-Y-Z system to the object. Fig. 7 displays the X-Y-Z axes in dashed lines and the cube is drawn in solid lines. The lower rear corner of this unit cube (the length of each side is one) is at the origin (0,0,0)—point 1. Point 2 shows distances of one from the origin on the X-axis and 0 from both the Y and Z axes. Point 3 shows distances of one from the X and Y axes and zero from the Z axis. Point 4 is the same as point 2 except for the Y axis. Now point 5 gets interesting—a zero distance from the X axis, but one from the Y and Z axis (the line connecting points 4 and 5 has a depth of one unit). Point 6 has a depth of one from the origin. Point 7 is like point 5. Point 8 shows a distance of one from all three axes. And since we can see that each point differs from the three points connecting it (for example, point 3 is connected by

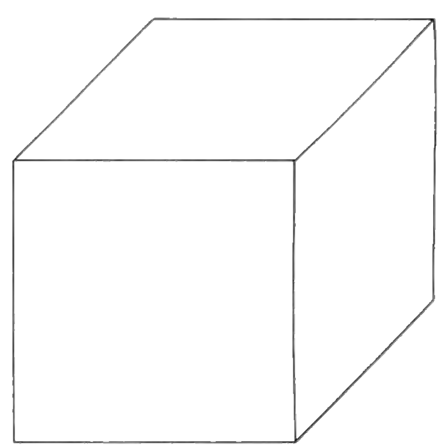


Fig. 2. The Solid Cube as Viewed from the Right Side

✓See List of Advertisers on page 354

"Now that I've given your mental CPU an 'Excedrin-80' headache, let me explain. . . how we rotate a cube."

points 2, 4 and 8) by one unit and the figure rests on three perpendicular axes, we indeed have a cube. Another way to demonstrate this is to note that each side of the figure is itself a square of equal size.

Rotation

Now that I've given your mental CPU an "Excedrin-80" headache, let me explain just how we rotate a cube. Rotation is accomplished by viewing the figure from different (X,Y,Z) coordinates. Instead of turning the cube, you are placed at different locations from which to view it. In essence, the cube isn't turning, but you are moving around the cube—a simulation of rotation.

The study of cubes and their rotation has its roots in solid analytic geometry. From the myriad of formulas and calculations in this field, I've identified two major steps needed before rotation of a cube (or any other 3-D object) can occur. First, a new set of viewing points must be established such that the figure will look as though it were on a gimbal and spinning freely. To accomplish this, we freeze the X and Y coordinates, say at 50 each, and just manipulate the Z coordinate. This will yield maximum control of the figure and a rotated cube that is quite appealing to the eye. The new Z value is assigned values that differ by an order of magnitude (exponentiation) because they produce very interesting plots. Secondly, the (X,Y,Z) coordinates used for viewing must be translated to (x,y) terms. To accomplish the feat, these equations are used:

$$x' = \arctan(Y/X)$$
$$y' = \arctan(Z/X)$$

The new points (x,y), created as follows, are used for plotting on your video screen:

$$x = Y*\sin(x') - X*\sin(x')$$
$$y = - X*\cos(x')*\sin(y') - Y*\sin(x')*\sin(y') + Z*\cos(y')$$

Using the coordinates from Fig. 7 and the algorithm above (having been derived from solid analytic geometry texts), we construct Table 1. If we plot the new (x,y) points, we would get the cube in Fig. 7 drawn on the X-Y axis. If you can visualize the point (30,30,30), the cube would look exactly as depicted in Fig. 8 when plotted on your CRT or on graph paper.

Not Done Yet

But even these figures are lacking something. Somehow all those lines, even to my imagination, look like a transparent etching rather than a solid cube.

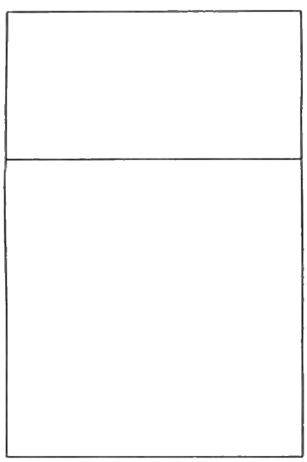


Fig. 3. The View of the Solid Cube from the Front and Top

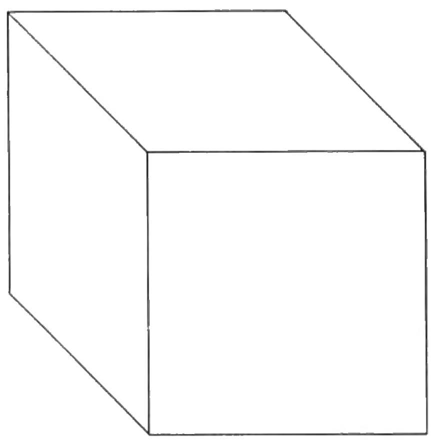


Fig. 4. The View of the Solid Cube from the Left Side

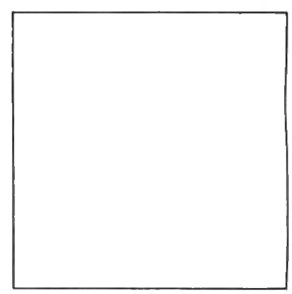
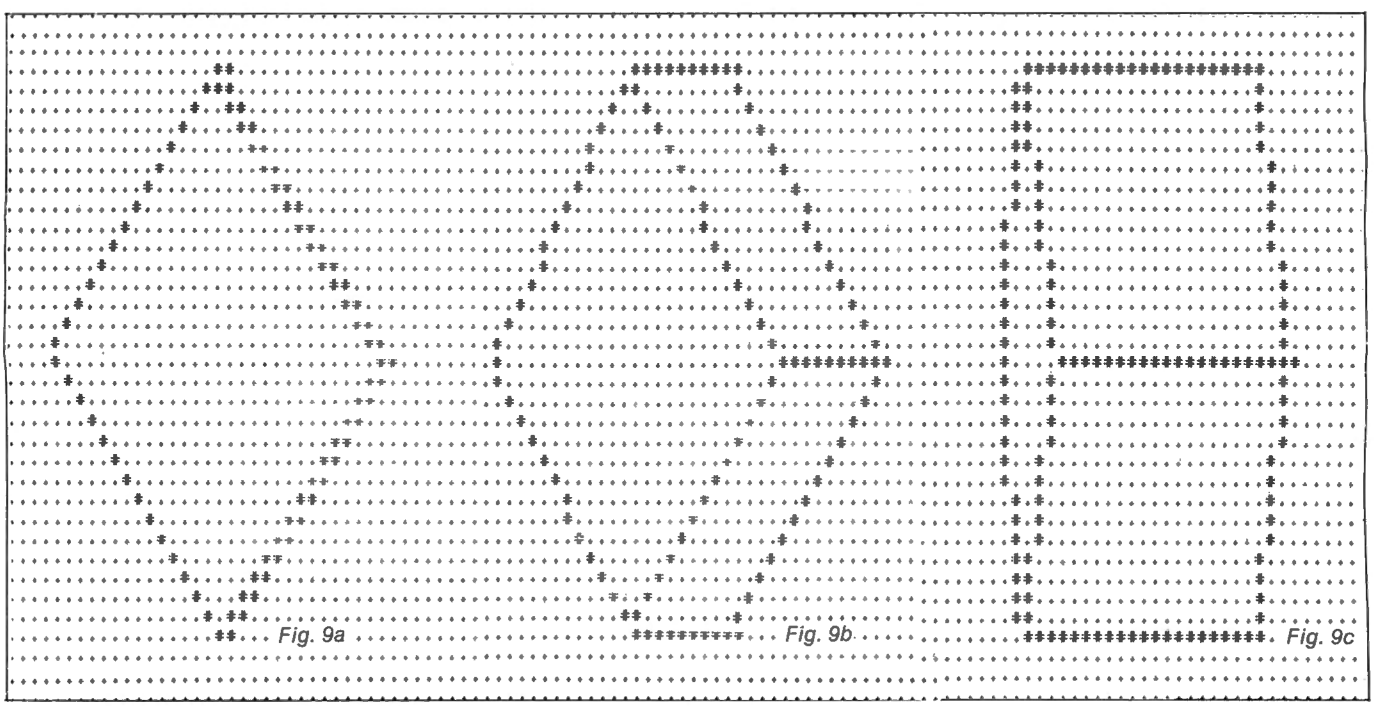


Fig. 5. The View of the Solid Cube from Head-On



"Program Listing 2 is the tightest Basic code I've ever written. It is fully optimized and undocumented."

How would the figure look with the hidden lines removed—just like a solid cube as illustrated in Figs. 2, 3 and 4! Next, how do we tell the Z-80 CPU to ignore the surfaces of the cube that our human eyes can't see?

The trick lies in the distance formula. As it turns out, the point farthest away from the viewing point (for example, (30,30,30)) is the vertex or corner that can't be seen. Attached to that vertex are lines, and without that vertex, the lines can't be drawn. Hence, they disappear! When Fig. 7 is viewed, for example, from (30,30,30) the distances to each of the verticies are shown in Table 2 when:

distance = square root[(p1 – X)**2 + (p2 – Y)**2 + (p3 – Z)**2]

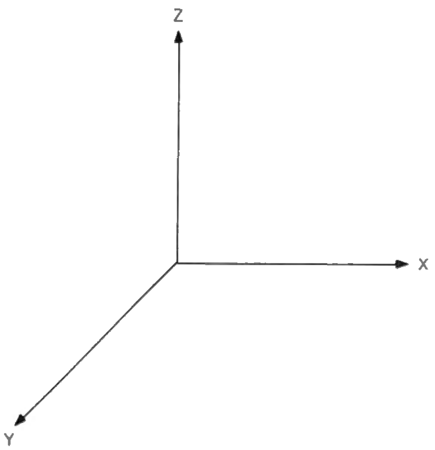


Fig. 6. The Representation of the X-Y-Z Coordinate System

where p1, p2 and p3 are the coordinates you wish to view the cube from. Point 1 is the farthest away from (30,30,30) with a distance of 51.96 units. Since it is not seen, line segments 1-2, 1-4, and 1-6 are not drawn. (The result, a solid cube with hidden lines removed (Fig. 2.).)

Ready At Last

Two versions of the same program are presented in this article. Program Listing 1 is fully documented and somewhat unoptimized. It takes about 10 seconds to display the cube without hidden lines and nine seconds to display the cube with hidden lines. Program Listing 2 is the tightest Basic code I've ever written. It is fully optimized and undocumented. I would be hard pressed to cut the execution time by one more second without direct machine language assistance. A 50 percent improvement in execution time is observed with Listing 2; the complete cube takes about five seconds and the hidden line cube takes slightly longer to draw. Needless to say, Listing 1 takes up much more MOS memory than Listing 2. Although Listing 2 looks totally different from 1, the overall logic for the two programs is identical. I've reduced the number of vertice connections needed to draw the cube, taken out all remarks, removed blank spaces, optimized the loops, and used many other techniques which I've imbedded into the code.

Enter the program of your choice, CSAVE it, then simply type Run. Three prompts are issued. Please Enter The Viewing Coordinates (X Y Z)? to which you enter the coordinates, separated by commas, of the point the cube is to be viewed from—for example, 30,30,30. Hardcopy Required (Y/N)? asks for a Y if a permanent copy of the cube is desired. The Auto Rotation (Y/N)? feature will spin the cube on its Z axis giving you seven different views while it is rotating.

Just sit back and be patient for a few seconds. Right before your eyes, the cube will be drawn. Actually, the plotting of the (x,y) coordinates using the Set statement is occurring. If you requested a hardcopy, dots are printed representing a pixel value of 0 (off) and a "#" where a pixel is lit (or 1). (See the hardcopy of a cube being rotated in figures 9a–f.) This paper image is designed for a 132 character by 66 line printout.

Looking Closer

After you've spent an hour or two display-

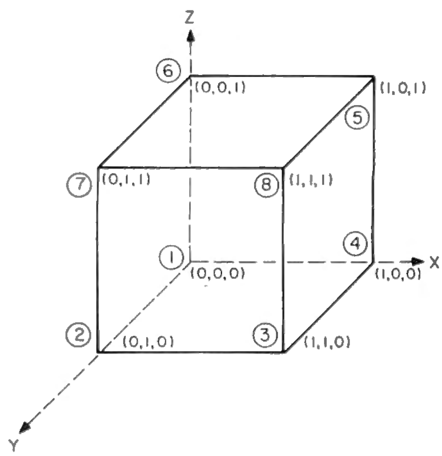


Fig. 7. A Fully Labeled Unit Cube Resting on the X-Y-Z Axis

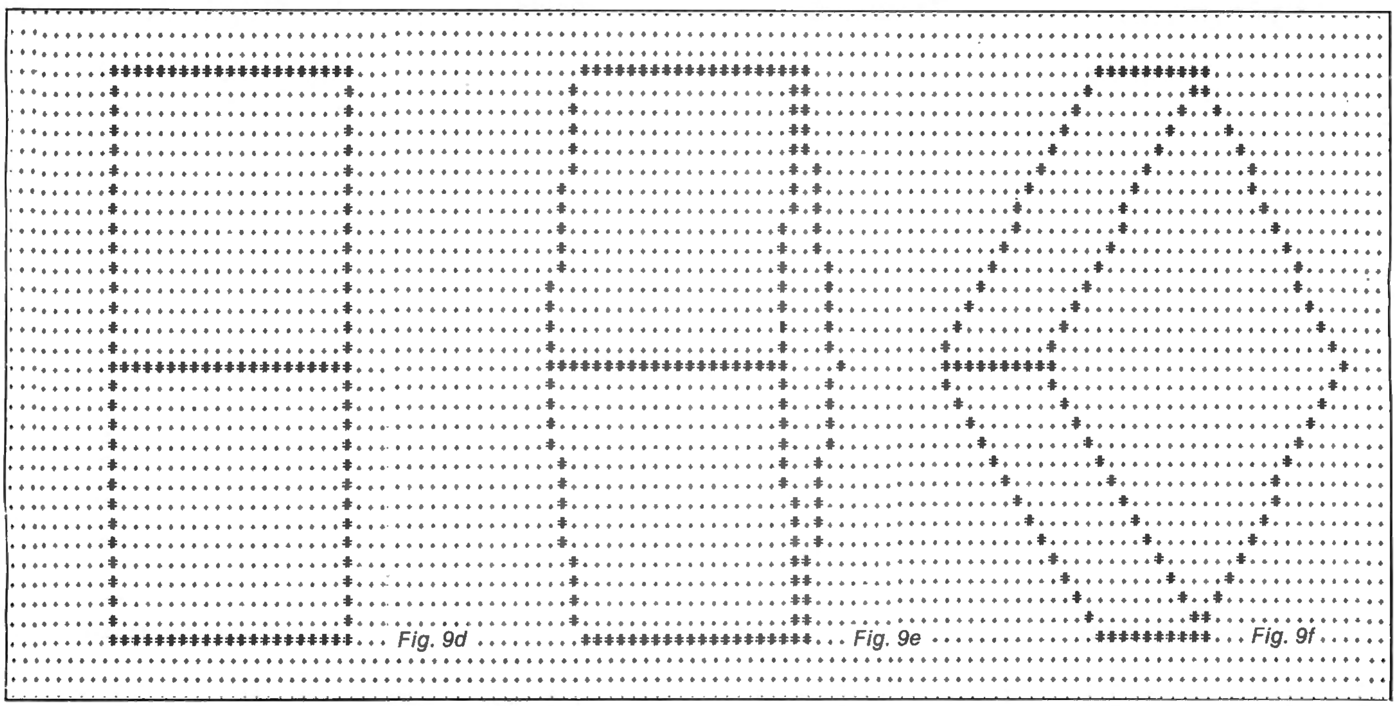


Fig. 9d *Fig. 9e* *Fig. 9f*

"A flowchart is a part of the documentation package for any software effort."

ing various cubes, you'll be ready to investigate the flowchart for a more in-depth treatment of the algorithms used.

A flowchart is a part of the documentation package for any software effort. It can either be very detailed or at a higher (or macro) level. Flowcharts use various symbols to represent different logical steps, decisions, and actions. Fig. 10 contains many of the common flowcharting symbols accepted by the data processing community.

The flowchart in Fig. 11 is at macro level for program 1. I have identified seven blocks worthy of a brief discussion. Block one defines the working array A to be eight rows of five columns. (You'll need to know this if

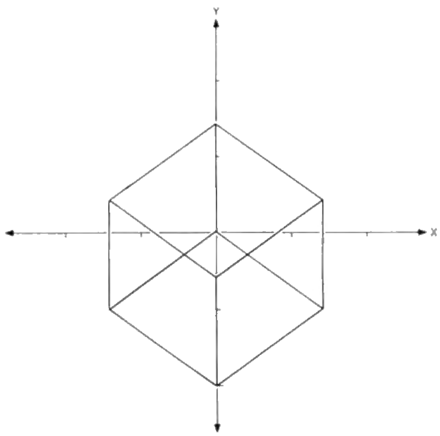


Fig. 8. The cube displayed on the X-Y axis.

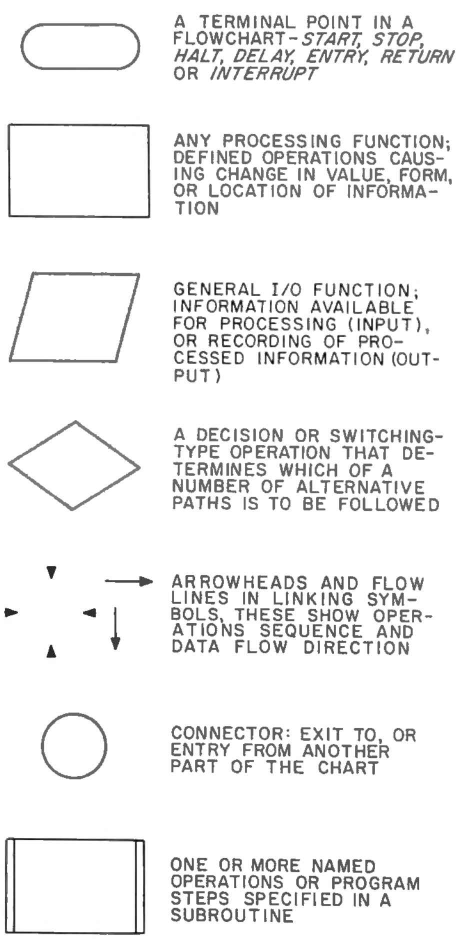


Fig. 10 Basic Flowcharting Symbols

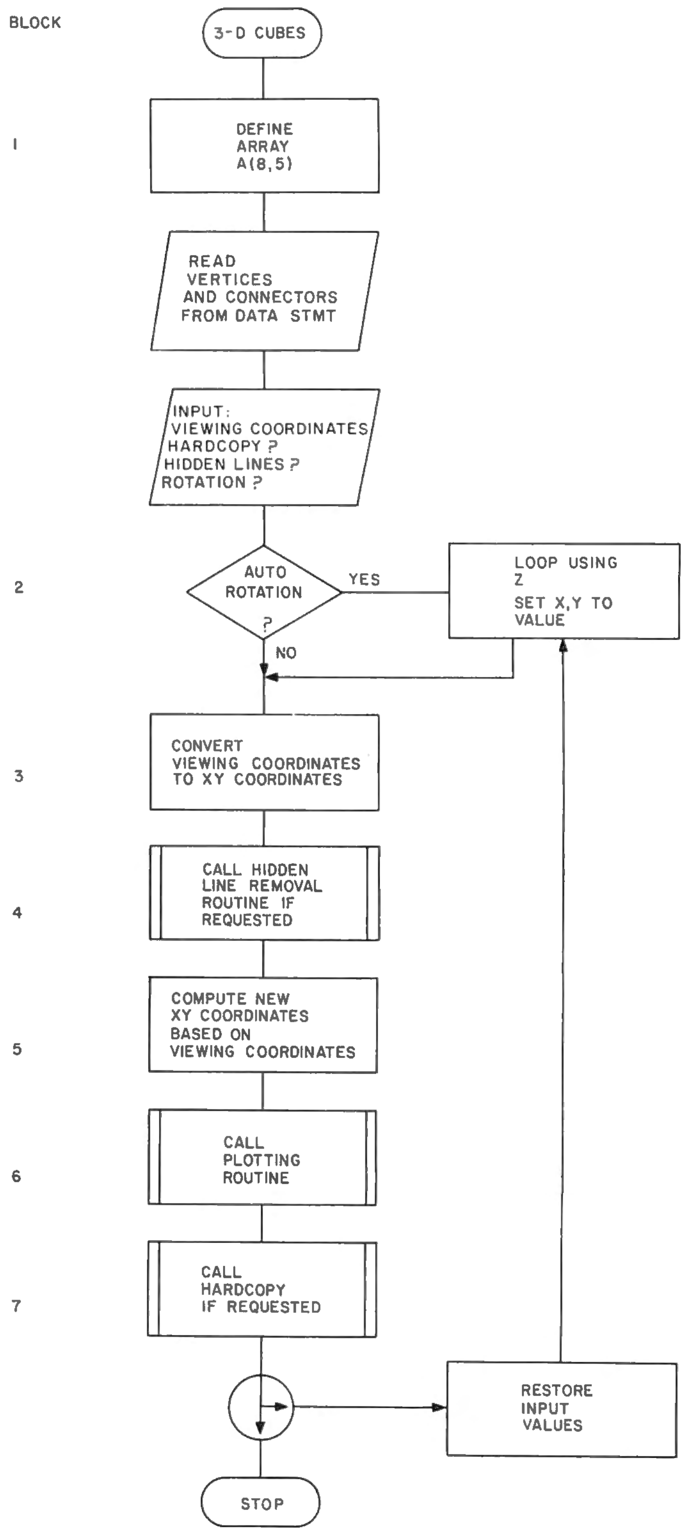


Fig. 11 Macro Flowchart of Program Listing 1

"The object of this method is not to calculate distance, but to find the hidden vertex."

you change from six-sided figures.) Eight rows are needed for eight verticies. Column one is for the X coordinate, column two for the Y coordinate with the Z coordinate in column three. Column four is used for the logical connection of the (X,Y,Z) vertice to the other adjacent vertice (i.e. the row number for connection purposes). Column five serves the same purpose as column four except that it contains the second row number/connection.

Block two sets up a Do loop if auto rotation is desired. The X and Y coordinates are set at 50 and values for Z of −1000, −100, −10, 0, 10, 100, 1000 are calculated by multiplying the sign of ZQ by 10, all raised to the ZQth power where ZQ ranges from minus three to three in increments of one. The B$ variable is required for the program to know about the existence of the ZQ loop.

The (x,y) coordinates to view the figure are obtained in block three as follows:

```
L = ATN(Y/X): M = ATN(Z/X)
```

where ATN is the arc tangent function. (This block involves some complicated mathematics and is beyond the scope of this article.)

Block four is activated if hidden line removal is desired. Removal of those lines hidden from sight is accomplished by using the distance formula. Its form for (X,Y,Z) coordinates in program 1 is:

```
G = SQR( (A(D,1) – X)↑2 + (A(D,2) – Y)↑2 + (A(D,3) – Z)↑2 )
```

where A(D,1), A(D,2) and A(D,3) are the coordinates of each vertice and X, Y and Z represent the coordinates of the viewing point.

This method, utilizing the square root function and exponentiation is very slow. An alternate way to find the vertice farthest away from the viewing coordinate is:

```
(1) G = ABS(A(D,1) – X) + ABS(A(D,2) – Y) +
        ABS(A(D,3) – Z).
```

This equation is faster to execute than the distance formula with its exponentiation and square root. The object of this method is not to calculate distance, but to find the hidden vertex. Table 2 shows that point 1 is the farthest from (30,30,30) with a distance of 51.96 units. Equation (1) would result in a distance of 90 units from (30,30,30)—(note: this equation is not to be construed as a distance formula) while point 8 (a distance of 50.22 units) measures 87 units. Both methods yield identical logical results—point 1 is a vertice that is hidden from sight in the cube. Note that more than one point may be hidden from view, but the program does not take this into account. The second or third point is redundant and merely causes the same line to be drawn again.

Block five computes the new (x,y) coordinates through:

```
A1 = A(I,1)
(2) A(I,1) = – A1*SIN(L) + A2*SIN(L)
(3) A(I,2) = – A1*COS(L)*SIN(M) – A2*SIN(L)*SIN(M)
    + A(I,3)*COS(M)
```

A1 is used because A(I,1) gets modified in equation (2) and the unmodified A(I,1) is required in equation (3). These new coordinates are now ready for plotting.

The routine to plot (x,y) point is the object of block six. The Y-intercept form of the straight line equation is used.

```
(4) M = (Y0 – Y1)/(X0 – X1)
(5) BZ = Y0 – M*X0
```

(X0,Y0) and (X1,Y1) are the end points for the

X	Y	Z	x'	y'
0	0	0	0	0
0	0	1	0	.707
0	1	0	.707	−.5
1	0	0	−.707	−.5
0	1	1	.707	.207
1	1	0	0	−1
1	0	1	−.707	.207
1	1	1	0	−.292

Table 1. The x and y values resulting from the cube being viewed from (30,30,30). x' and y' both equal .785.

X	Y	Z	Pt. #	Distance
0	0	0	1	51.96
0	0	1	6	51.39
0	1	0	4	51.39
1	0	0	2	51.39
0	1	1	5	50.81
1	1	0	3	50.81
1	0	1	7	50.81
1	1	1	8	50.22

Table 2. The distance shown is from each (X,Y,Z) point to the viewing coordinates of (30,30,30).

✓*See List of Advertisers on page 354*

"A paper image of the screen is produced by utilizing the Point command."

line to be plotted. Equation (4) is the formula for calculating the slope of a line and equation (5) calculates the Y1 value for plotting given (X0,Y0) and an incremented point between X0 and X1.

The Set command is used for plotting with the center of the TRS-80 CRT screen being (64,24). When the X0 value is not equal to X1 (i.e., a non-vertical line), the 'DO R' Do loop calculates the incremental value between X0 and X1, which is then multiplied by the slope. The Y intercept and the screen center values are then added in:

SET(M*R + BZ + 64, R + 24)

If X0 equals X1, a vertical line is needed. R gets an increment value between Y0 and Y1 and SET(R + 64, X0 + 24) is used.

Block seven contains the hardcopy routine. A paper image of the screen is produced by utilizing the Point command. If a pixel is not lit, the point of that pixel returns a zero. The "." is printed for a blank space (0) and a "#" if the pixel is on. I designed this little routine with a full width printer in mind (such as my LA 36 Decwriter II). If your printer can't produce plots of 128 characters across, you'll have to modify this routine to suit your needs.

Attached to block seven is a small circle representing the Next ZQ statement used by the auto rotation calculation. If this option was selected, the looping takes place; otherwise the program halts.

The eight vertices necessary for the cubes construction are contained in the data statements in lines 2010-2080 of Program 1. Each of the statements corresponds to the form of:

X, Y, Z, first connection, second connection.

Connection simply means which two data statements the vertice is connected to. For example, (0,0,0) is given by 2010 data 0,0,0,2,3 and is connected to the vertex in line 2020 (2) and to the vertex in line 2030 (3). Fig. 9a-f shows this very clearly. In turn, each vertex is linked to two other vertices for example, (0,1,0) connects to (1,1,0) and (0,1,1).

If you are observant, you'll notice that a vertex in a cube is connected by three other vertices, not just two. But only two are required by this program. This is true as long as the first and second connection numbers are used only twice by the entire set of eight data statements. (Because the hidden

Program Listing 1

```
10 REM ROTATION OF THREE DIMENSIONAL OBJECT BY BRUCE A. YELLIN
20 REM THIS LISTING IS FULLY DOCUMENTED AND IS NOT ORIENTED
30 REM TOWARDS OPTIMIZED CODE.
40 REM FOR THE FASTEST EXECUTIUON SPEED, USE LISTING #2
95 REM * * * * * * * * * * * * * * * * * * * * * * * * * * * *
96 REM *                                                     *
97 REM *               INITIALIZE VARIABLES                  *
98 REM *                                                     *
99 REM * * *$* * * * * * * * * * * * * * * * * * * * * * * * *
100 DIM A(8,5):                 REM ARRAY A HAS ROOM FOR 8
101                             REM VERTICIES OF 3 COORDINATES
102                             REM EACH AND 2 CONNECTION
103                             REM NUMBERS
110 FORI=1 TO 8:                REM ENTER OBJECTS' VERTICIES &
111                             REM AND CONNECTIONS
120    READ A(I,1), A(I,2), A(I,3), A(I,4), A(I,5)
130 NEXT I
140 CLS
195 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
196 REM *                                                   *
197 REM *       PROMPT FOR VIEW AND OPTIONS                 *
198 REM *                                                   *
199 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
200 INPUT "PLEASE ENTER THE VIEWING COORDINATES (X Y Z)";X,Y,Z
210 INPUT "HARDCOPY REQUIRED (Y/N)";A$
220 INPUT "ARE HIDDEN LINES DESIRED (Y/N)";B$
230 INPUT "AUTO ROTATION (Y/N)";C$
295 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
296 REM *                                                   *
297 REM * SET UP AUTO ROTATION LOOP, CONVERT VIEWING POINTS *
298 REM * AND HIDDEN LINE SUBROUTINE                        *
299 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
300 IFC$="N"THEN 340:           REM AUTO ROTATION REQUIRED, SET
301                             REM UP LOOP ZQ FOR VALUES SO
302                             REM   X       Y       Z
303                             REM   50      50      -1000
304                             REM   50      50      -100
305                             REM   50      50      -10
306                             REM   50      50      0
307                             REM   50      50      10
308                             REM   50      50      120
309                             REM   50      50      1000
310 FOR ZQ= -3 TO 3
320    Z=SGN(ZQ)*10[ABS(ZQ)
330    X=50:Y=50
340 IF X=0 THEN X=0.000001:     REM PREVENT DIVISION BY ZERO
350 L=ATN(Y/X): M=ATN(Z/X):                REM CONVERT VIEWING
COORDINATES
360 IF B$="N" GOSUB 1100:       REM HIDDEN LINE REMOVAL ROUTINE
361                             REM IF REQUESTED(Y=LINES DESIRED)
395 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
396 REM *                                                   *
397 REM *     COMPUTE NEW SET OF X,Y COORDINATES            *
398 REM *                                                   *
399 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
400 FOR I=1 TO 8
410    A1=A(I,1)
420    A(I,1)= -A1*SIN(L) + A(I,2)*SIN(L)
```

Program continues

> *"Let me also mention that any number can be used as a coordinate as long as the chosen value remains constant throughout the cube. . ."*

lines are not drawn when the option is selected, lines can be eliminated from the cube if they are improperly linked.) Take a moment and notice the data statements procedure for connecting the vertices.

Let me also mention that any number can be used as a coordinate as long as the chosen value remains constant throughout the cube (for example, (0,20,0), (20,20,0), etc.). In fact, drawing units of one will lead to a distorted plot on the TRS-80's low resolution CRT. I prefer units of 20 because it allows better resolution and also fits on the screen (whose height is only 48 pixels). However, be my guest and try your own set of vertices—experimentation is half the fun!

The cube is slightly distorted when displayed on the CRT. To understand why, let's examine the size of a TRS screen. The area used on the 12 inch monitor measures 7.25 inches across and six inches down. Divide these measurements by 128 pixels across and 48 down. A pixel measures approximately 0.056 inches by 0.125 inches. These rectangular shaped pixels result in a greatly distorted plot of a line in most cases. For example, the plot of a line from (0,0) to (47,47) results in an angle of line to horizon of over 70 degrees while it should be only 45 degrees. In terms of a cube, the sides then become flatter than they really are. (After all, the TRS-80 isn't a $30,000 graphics computer.) The hardcopy also has some distortion, although it is not as great as on the CRT.

Other drawbacks include the fact that Basic is extremely slow. However, it would

Program continued

```
430    A(I,2)= -A1*COS(L)*SIN(M) - A(I,2)*SIN(L)*SIN(M) + A(I,3)
*COS(M)
440 NEXT I
495 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
496 REM *                                                   *
497 REM *       PLOT X,Y COORDINATES ON SCREEN              *
498 REM *                                                   *
499 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
500 CLS
510 PRINT@0, X; Y; Z;
520 FOR I = 1 TO 8
530    IF I = G2 THEN 580
540    X0 = A(I,1)
550    Y0 = A(I,2)
560    IF A(I,4) <> G2 THEN X1 = A(A(I,4),1): Y1 = A(A(I,4),2):
GOSUB 900
570    IF A(I,5) <> G2 THEN X1 = A(A(I,5),1): Y1 = A(A(I,5),2):
GOSUB 900
580 NEXT I
595 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
596 REM *                                                   *
597 REM *              HARDCOPY REQUESTED ?                 *
598 REM *                                                   *
599 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
600 IF A$="Y" GOSUB 800
695 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
696 REM *                                                   *
697 REM *          WAS AUTO ROTATION IN EFFECT?             *
698 REM *                                                   *
699 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
700 IF C$="N" THEN STOP
710 RESTORE:                       REM REINITIALIZE VERTICIES &
711                                 REM CONNECTIONS, END ZQ LOOP
720 FOR I = 1 TO 8
730    READ A(I,1), A(I,2), A(I,3), A(I,4), A(I,5)
740 NEXT I
750 NEXT ZQ
760 STOP
800 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
801 REM *                                                   *
```

Program continues

Program continued

```
802 REM *  HARDCOPY ROUTINE - DESIGNED FOR 132 BY 66 FORMAT *
803 REM *                                                   *
804 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
805 LPRINT X; Y; Z: LPRINT
810 FOR I = Ø TO 47
820     FOR J = Ø TO 127
830          IF POINT(J,I)=Ø THEN LPRINT ".";: ELSE LPRINT "#";
840     NEXT J
850     LPRINT
860 NEXT I
870 FOR I = 1 TO 16
880    LPRINT
890 NEXT I
899 RETURN
900 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
901 REM *                                                   *
902 REM *                PLOTTING ROUTINE                   *
903 REM *                                                   *
904 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
910 IF XØ =X1 THEN 980:         REM IF DIVISION BY ZERO, USE
911                             REM Y FORMULA WITH X CONSTANT
920 M=(YØ-Y1)/(XØ-X1):          REM SLOPE FORMULA
930 BZ=YØ-M*XØ:               . REM EQUATION OF A LINE
940 FOR R = XØ TO X1 STEP SGN(X1-XØ)
950    SET(M*R+BZ+64,R+24)
960 NEXT R
970 RETURN
980 FOR R = YØ TO Y1 STEP SGN(Y1-YØ)
990    SET(R+64,XØ+24)
1000 NEXT R
1010 RETURN
1100 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
1101 REM *      HIDDEN LINE REMOVAL SUFROUTINE                 *
1102 REM *   CALCULATE THE FARTHEST VERTICE FROM vIEW POINT   *
1103 REM *                                                   *
1104 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
1110 G1=Ø
1120 FOR D = 1 TO 8
1130   G=SQR((A(D,1)-X)[2 + (A(D,2)-Y)[2 + (A(D,3)-Z)[2)
1140   IF G>G1 THEN G1=G: G2=D
1150 NEXT
1160 RETURN
1995 REM* * * * * * * * * * * * * * * * * * * * * * * * * * *
1996 REM*                                                   *
1997 REM*      THE FORM OF THE DATA STATEMENTS ARE :        *
1998 REM*      X, Y, Z, CONNECTION #1, CONNECTION #2        *
1999 REM* * * * * * * * * * * * * * * * * * * : * * * * * * *
2010 DATA  Ø,  Ø,  Ø,  2,  3
2020 DATA  Ø,  Ø, 2Ø,  1,  5
2030 DATA  Ø, 2Ø,  Ø,  6,  5
2040 DATA 2Ø,  Ø,  Ø,  1,  7
2050 DATA  Ø, 2Ø, 2Ø,  3,  8
2060 DATA 2Ø, 2Ø,  Ø,  4,  8
2070 DATA 2Ø,  Ø, 2Ø,  4,  2
2080 DATA 2Ø, 2Ø, 2Ø,  7,  6
3000 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
3010 REM * LIST OF VARIABLES USED AND THEIR MEANING            *
3020 REM *                                                     *
3030 REM * A( , )       USED FOR BUILDING/STORING THE CUBE     *
3040 REM * I            FOR LOOPING PURPOSES                   *
3050 REM * A$           HARDCOPY RESPONSE                      *
3060 REM * B$           HIDDEN LINE RESPONSE                   *
3070 REM * C$           AUTO ROTATION RESPONSE                 *
3080 REM * ZQ           FOR LOOPING PURPOSES IN AUTO ROTATION*
3090 REM * X            X COORDINATE                           *
3100 REM * Y            Y COORDINATE                           *
3110 REM * Z            Z COORDINATE                           *
3120 REM * L            ARC TANGENT(Y/X)                       *
3130 REM * M            ARC TANGENT(Z/X)                       *
3140 REM * A1           TEMPORARY HOLDING VARIABLE             *
3150 REM * XØ           X COORDINATE (FROM)                    *
3160 REM * X1           X COORDINATE (TO)                      *
3170 REM * YØ           Y COORDINATE (FROM)                    *
3180 REM!* Y1           Y COORDINATE (TO)                      *
3190 REM * J            FOR LOOPING PURPOSES                   *
3200 REM * M            SLOPE                                  *
3210 REM * R            FOR LOOPING VALUE                      *
3220 REM * D            FOR LOOPING PURPOSES (HIDDEN LINES)    *
3230 REM * G            DISTANCE FORMULA                       *
3240 REM * G1           LONGEST DISTANCE VALUE                 *
3250 REM * G2           POINTEr TO LONGEST DISTANCE VALUE      *
3260 REM *                                                     *
3270 REM * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

See List of Advertisers on page 354

"There are an infinite number of shapes to draw and an infinite number of ways to look at them."

prove to be a difficult program in Z-80 assembly language. Also, due to the distortion by the CRT, I've found that rotation with just the Z axis works better than any other axis or combination of axes.

Expansion of this program is necessary to speed it up and allow for other shapes to be drawn. Three areas can be identified for further investigation. First, smaller intervals should be used in the auto-rotation routine. This would make the cube rotate with more fluidity. Next, machine language should be part of the code, helping to speed up the plotting and allowing the cube to be turned faster. Finally, other polyhedra should be tried. Unfortunately, this will probably lead to the use of a matrix manipulation method for representation and an entirely different program logic.

I've just scratched the surface of a very technical, yet beautiful subject. There are an infinite number of shapes to draw and an infinite number of ways to look at them. For those with adventure in their hearts, having the program rotate rectangles should provide some viewer excitement. A step beyond rectangles are pyramids, but I dare not hazard a guess as to what they look like when rotated!

A text, *Principals of Interactive Computer Graphics* by William Newman and Robert Sproull (McGraw Hill, 1973) deals with this subject and others in great detail. If I had read this book years ago, these rotating cubes would undoubtedly have approached the quality of those produced on fine graphics computers. After all, the TRS-80 can't be too far behind.■

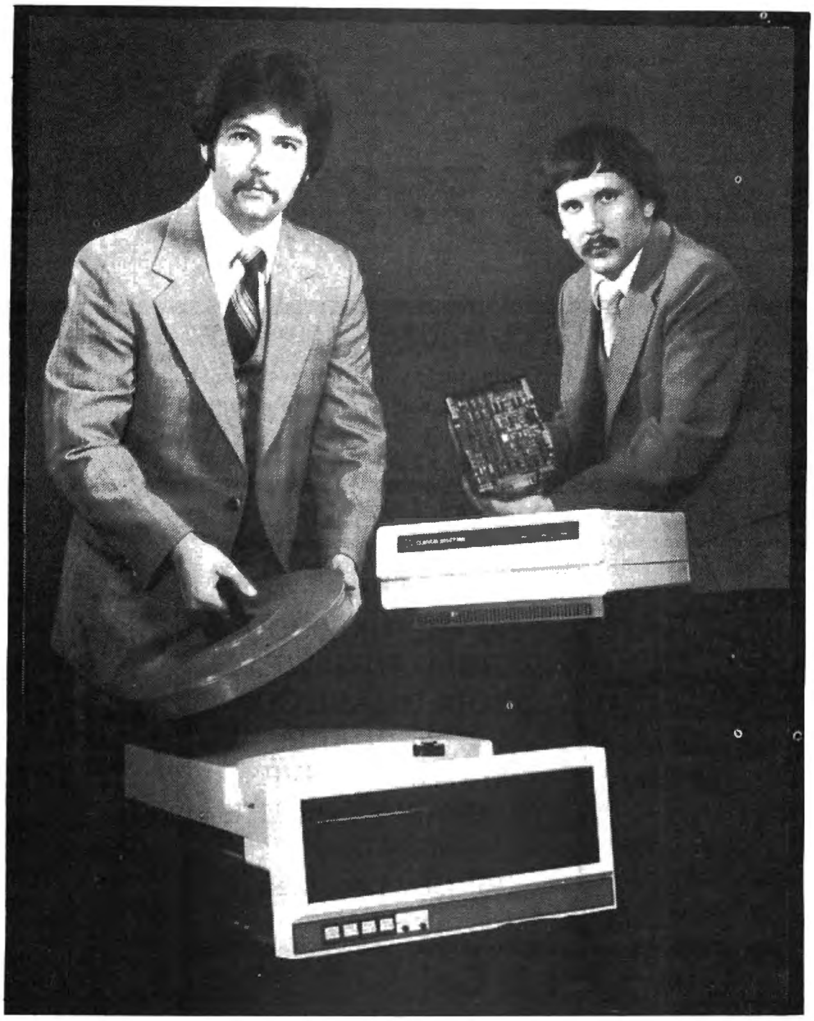
```
10 DEFINT I,X,Y,J,W,E,R,K,G,D:E=60:W=25:DIMA(8,5)
20 FORI=1TO8:READA(I,1),A(I,2),A(I,3),A(I,4),A(I,5):NEXT
50 CLS
60 INPUT "PLEASE ENTER THE VIEWING COORDINATES (X Y Z)";X,Y,Z
70 INPUT "HARDCOPY REQUIRED (Y/N)";A$
71 INPUT "ARE HIDDEN LINES DESIRED (Y/N)";B$
72 INPUT "AUTO ROTATION (Y/N)";C$
75 IFC$="Y"FORZQ=-3TO3:Z=SGN(ZQ)*10[ABS(ZQ):X=50:Y=50
80 IFX=0 L=SGN(Y)*1.5708:M=SGN(Z)*1.5708:GOTO90
85 L=ATN(Y/X):M=ATN(Z/X)
90 IFB$="N"GOSUB 900
100 SL=SIN(L):SM=SIN(M):CL=COS(L)*SM:Cm=COS(M):A4=SL*SM
130 FORI=1TO8
135 IFI-G2THEN A1=A(I,1):A(I,1)=A(I,2)*SL-A1*SL+W:A(I,2)=A(I,3)*
CM-A1*CL-A(I,2)*A4+E
140 NEXT
200 CLS:PRINT@0,X;Y;Z
210 FORI=1TO8:IFG2=ITHEN240
215 X0=A(I,1):Y0=A(I,2):J=A(I,4)
225 IFJ-G2 THEN X1=A(J,1):Y1=A(J,2):GOSUB700
230 J=A(I,5):IFJ=0ORJ=G2THEN240
235 X1=A(J,1):Y1=A(J,2):GOSUB700
240 NEXTI
250 GOSUB500
260 IFC$="Y"THENRESTORE:FORI=1TO8:READA(I,1),A(I,2),A(I,3),A(I,4
),A(I,5):NEXTI,ZQ
270 STOP
500 IF A$="N"RETURN
503 LPRINT X;Y;Z:LPRINT
505 FORI=0TO47
510 FORJ=0TO127
520 IFPOINT(J,I)THENLPRINT"#";:ELSE LPRINT".";
530 NEXT:LPRINT:NEXT
540 FORI=1TO16:LPRINT:NEXT:RETURN
700 IFX0-X1THEN720
710 FORR=Y0TOY1STEPSGN(Y1-Y0):SET(R,X0):NEXT:RETURN
720 M=(Y0-Y1)/(X0-X1):BZ=Y0-M*X0
730 FORR=X0TOX1STEPSGN(X1-X0):SET(M*R+BZ,R):NEXT:RETURN
900 G1=0
910 FORD=1TO8:G=ABS(A(D,1)-X)+ABS(A(D,2)-Y)+ABS(A(D,3)-Z):IFG>G1
THENG1=G:G2=D
920 NEXT:RETURN
2000 DATA 0,0,0,3,0
2010 DATA 0,0,20,5,1
2020 DATA 0,20,0,6,0
2030 DATA 20,0,0,1,7
2040 DATA 0,20,20,8,3
2050 DATA 20,20,0,4,8
2060 DATA 20,0,20,2,0
2070 DATA 20,20,20,7,0
```

Program Listing 2

NOBODY KNOWS MORE ABOUT ADDING MEGABYTES TO YOUR TRS-80 MODEL II* THAN AMERICAN SMALL BUSINESS COMPUTERS

THE CAMEO CONNECTION

The Cameo 10 Million Character Disk Drive Subsystem (pictured in foreground) Is A 5 Megabyte Fixed, 5 Megabyte Removable Cartridge-Type Disk Drive. Advantages To This Unit Are Ease of Backup For Data and Program Files and Low Cost. By Attaching A Multiplexor Unit Several Model II's Can Be Attached To The Cameo Disk Drive. For Further Pricing, Availability, and Operating Systems Options, Call or Write to American Small Business Computers and ask for the CAMEO CONNECTION.



THE CORVUS CONNECTION

The Corvus Disk Drive Subsystem (center of picture) Is Available in 5, 10, and 20 Million Character Capacities. The Corvus System Is The Ultimate In The Sealed "Winchester, All-Fixed" Type Drive. The Corvus Is Designed To Operate In The Harshest Environments Such As Factories, Oil Rigs, Etc. Backup Of Data Is Accomplished Via Optional MIRROR Backup Device Whereby Data Is Written Onto A Standard Video Tape Recorder. A Great Number of Model II's May Be Attached To The Corvus Via An Optional CONSTELLATION Multiplexor. For Further Pricing, Availability, and Operating Systems, Call Or Write to American Small Business Computers and Ask for the CORVUS CONNECTION.

THE AMERICAN SMALL BUSINESS COMPUTERS MINI-WINCHESTER SPECIAL

The MINI-5 (being held by Corporate President Robert Webster) Is American Small Business Computer's Very Own Disk Subsystem. The MINI-5 Offers Approximately 5 Million Characters of Information Storage and is Based On the Famous Seagale Technology ST-506 Micro Winchester Drive. Now You Can Own State-Of-The-Art Technology At An Affordable Price — Only $3500 — Which Includes Drive, Controller Electronics, Power Supply, Cabinet, Cables, Documentation and Operating System Software.

DEALER INQUIRIES INVITED

AMERICAN SMALL BUSINESS COMPUTERS ✓529

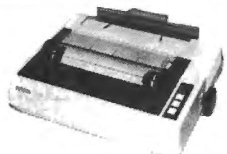
*TM TANDY CORPORATION

118 SOUTH MILL ST. PRYOR, OK. 74361 918-825-4844

SMALL BUSINESS COMPUTERS

118 SO. MILL ST. • PRYOR, OK 74361 • (918) 825-4844 *TM TANDY CORP.

SUPERPRINTER THE EPSON MX-80



This is the new Epson MX-80 dot matrix printer. It does just about everything you could ask a printer to do-quickly, quietly, reliably. In fact, for OEM installations, the MX-80 may be the single best, all-around printer you can buy. But that's not the best reason to buy it.

The MX-80 prints bidirectionally at 80 CPS in a user-defined choice of 40, 80, 66 or 132 columns. And if that's not fast enough, its logical seeking function minimizes print head travel time. The MX-80 prints 96 ASCII, 64 graphic and eight international characters with a tack-sharp 9 x 9 matrix. For a long time. Epson printers are known for reliability and the MX-80 is no exception. But that's not the best reason to buy it, either.

The print head has a life expectancy of up to 100 x 10^6 characters, and when it wears out, just throw it away. A new one costs less than $30 and the only tool you need to change it is attached to the end of your arm. The MX-80 is compact, weighs only 12 lbs., and the whole unit, including the two stepper motors controlling carriage and paper feeding functions, is precisely controlled by an internal microprocessor. But even that isn't why you should specify the MX-80.

The best reason is this: because Epson makes more printers than anyone else in the world, we can afford to sell each one for a little less.

IF YOU BUY AN MX-80 AND DON'T CHECK OUR PRICE — YOU PAID TOO MUCH! ✓526

OKIDATA MICROLINE 80 PRINTER — $399.00

The Microline is built on a rugged cast aluminum base to withstand the rigors of continuous business use. It is driven by two motors and will run all day with no duty cycle limitations. Microline printers use a dense 9 x 7 dot pattern to produce crisp, clean copies, first copy to last. The seven pins in the head are "fired" using energy stored in tension members. This technology permits the use of short, low mass pins made with an extremely hard alloy. The head produces less heat, thereby extending its life. ✓528

★ CHECK IT OUT! ★ EPSON MX-80FT

That's right — *MX-80FT*.

The FT on the end means Friction and Tractor. The Fantastic *MX-80* Printer is now available in a version which will accept letterhead-type paper AND tractor-type paper. Call or write for our (as always) unbelievable low price.

Also available MX-70 and MX-100 ✓560

AW...WHAT THE HECK

RAM Memory Chips for the TRS-80

It is the policy of American Business Computers to offer merchandise at the lowest price possible. Several months back, we began selling RAM Memory Chips for the TRS-80 for $45.00 per set. Someone else sold chips for $44.00. We sold them for $38.00. They sold them for $37.95. So we say "AW...WHAT THE HECK!" Let's see the other guys beat this price.

$19.95 PER 16K SET

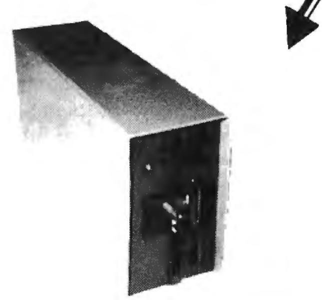
These chips are brand new "4116's". These 200 nanosecond chips are fully compatible with all TRS-80 products. Instructions for insertion are included; however, the dip shunts required for converting a 4K Model I to a 16K Model I are not included at this low price. ✓527

★ RETROFIT YOUR ★ MX-80 TO AN MX-80FT

ADD A PLATEN TO YOUR MX-80 TO ALLOW USE OF LETTER-HEADS OR ROLL PAPER. **$69.95** ✓561

DIRT CHEAP ...$275

*TRS-80 MODEL I VERSION
Complete with Power Supply & Cabinet



But Highly Reliable. Yes, our Mini Disk Drives have an estimated 8000 Power-On-Hours Mean Time Between Failure (*That's 8 Hours A Day, 5 Days A Week, for 3.85 YEARS*). Our drives also feature a lead screw head positioner - *not a cheap band-type positioner*. If you purchase a Mini Disk Drive with a band-type positioner, you are buying an INFERIOR PRODUCT! If you have any doubts about our drives, our *ONE YEAR* Warranty should convince you that *we* know this unit is the *best* drive at the *best* price on the *American* market. *This is one example where you can pay less and get more.* ✓562

✓See List of Advertisers on page 354

A look at some advanced utilities for the serious hacker.

Software Tools for the Craftsman

Remodel and Proload
Infinite Basic
Racet Computes
Orange, CA

Maurice M. Small, Ph.D.
University of Missouri
5400 Arsenal St.
St. Louis, MO 63139

Any programmer knows that TRS-80 Level II Basic has limitations, primarily its lack of renumbering, merge and append functions. It is also difficult to add new statements and commands to the language. Finally, Level II Basic is an interpreted rather than a compiled language. This discourages a structured programming approach, making it difficult for even experienced programmers to understand a long program several months after it was written. Three utilities available today all but eliminate these limitations.

Remodel & Proload

Remodel & Proload by Racet Computes of Orange, CA, comes in three versions (16K, 32K and 48K), loads into the top of memory with the System command, and is approximately 3K long. Also included are two well-written user's manuals which document the utility in a straightforward manner with text and examples. The program comes on a high-quality cassette (two copies on the same side), loads without hitch (in six months of frequent use I have yet to experience a loading problem), and can be easily transferred to disk or stringy floppy.

Remodel (REnumber MOve DELete) gives complete freedom in renumbering any portion of a Basic program (or the entire program) and in moving or deleting sections of Basic code (either a line or a complete subroutine) at will.

About the only restriction on this freedom is that Remodel and Proload will not execute a command to overwrite an existing line. For example, if you tried to move a section of code into an area where it would overlap other lines (e.g., move lines 10 and 20 between lines 30 and 31) Remodel and Proload will give you an error message (Range Bad Redo) and return you to the input state.

This procedure could be accomplished, though, if you renumber line 30.

The command format is simple and straightforward, and it returns to the command input state after executing each command. This is convenient for executing a series of commands. Exit and reentry are also uncomplicated: Simply depress the Break key (return to Basic) or type ? USR (0) (reenter Remodel and Proload). The other side of the cassette contains two copies of just Remodel. Since Remodel alone takes about 1K less room than the combination, load this side of the tape when memory space is at a premium.

With Remodel and Proload you can develop a library of data statements or string literals (1000 A$ = "in capital letters"). Since Proload allows partial or complete loads, loading programs from tape, and appending or merging them with a program in memory, you can effortlessly pass data statements and even USR routines (if the USR routines are stored in data/string literal statements) between programs. This method is considerably faster than using Input #—1 and Print #—1 statements.

Possibly of even greater importance, Remodel and Proload encourages structured programming. The program is divided into two major parts: a control section consisting of a loop which calls in sequence the relevant subroutines via GOSUB statements, and a list of subroutines which make up the body of the program. Proload creates a library of useful general and special-purpose routines which can then be appended or merged into your application program as needed. Not only does this simplify debugging and greatly increase your programming productivity, but with proper documentation it also makes your programs more readable and understandable.

I can now create long and complex programs in much less time than it took to write, type in and debug many shorter subroutines in the library. Remodel and Proload has increased my productivity at least threefold.

Infinite Basic and Extensibility

Infinite Basic comes in two versions (cassette and disk) on a high-quality cassette, and includes two well-written and complete manuals. Infinite Basic has a system module (basic interface, system routines and linking loader) as well as several application modules (Infinite String—50 new string functions, Infinite Matrix—30 new matrix functions, and Infinite Business—selling separately for $29.95 but requiring Infinite Basic and providing 20 additional functions).

Obviously with so many functions available not all of them are loaded into RAM at once. The user must first select the functions for a particular application program, and, with the systems module, create a machine language load module containing these desired functions. This load module can reside at almost any memory location. It is then saved separately on cassette, disk or wafer. When reloaded the load module provides the desired functions. The Infinite Basic functions run considerably faster (at machine language speed) and are more compact than similar Level II Basic subroutines. Moreover, the machine language load modules can be stored in Basic statements (data, string, string literals and dummy REM statements). Thus, the modified load module can be selectively loaded and saved by Proload (like Basic subroutines) as well as becoming a permanent part of the user's application program.

Infinite Basic Functions

The Matrix Input/Output group—the Matrix Read and Write Tape as well as the Matrix Data Read and Restore statements—are extremely efficient because they operate on entire blocks or arrays. This speeds up data (string and/or numeric) input and output at least twofold.

The statement format allows you to specify the number of elements to be operated on, a block number permitting only selected parts of the data tape to be read

"In a nutshell, Infinite Basic confers upon your Level II ROMs a considerable degree of extensibility."

by a subsequent Matrix Read Tape statement, and the cassette number (0 or 1). Similarly the Matrix (data) Element Read and Restore statements, which operate on data statements, are considerably faster than their Level II counterparts. They allow you to specify the number of data elements to be read in addition to the Basic statement (line) number, where any subsequent read will begin.

The Infinite Basic Matrix package also contains a generalized subroutine Call and Return function. These functions, as in Fortran, permit you to temporarily specify variable names corresponding to those used in a subroutine and are very useful, for example, if you have a subroutine to sum the data from several different arrays.

The Infinite Basic String package uses a number of functions to compress and decompress strings and data from eight bits per character to four, five, six, or seven bits depending on the type of character involved (numeric, alphabetic, alpha/numerics and alpha/numerics containing lowercase characters respectively). Also included are functions which pack and unpack character strings by special encoding of repeated characters.

Appropriate use of these functions often lets you store a lot more data in memory than you could otherwise. String translation functions (a character-by-character translation of one string into another), string manipulation functions (justification, truncation, packing, shifting, rotation, deletion and insertion), as well as others, are supported.

The Absolute String Manipulation functions enable you to manipulate strings directly without the considerable overhead inherent in the use of the Basic MID$, LEFT$, or RIGHT$ statements. For example, consider: 10 A$= "ABCD..GHIJ" 20 A$= LEFT$(A$,4)+"XX"+RIGHT$(A$,4). Line 20 creates three separate strings (LEFT$ part, LEFT$+ "XX", and the final result) in the string storage area.

By allowing you to directly manipulate strings, the Infinite Basic functions (such as &STIN-String Insert) all but eliminate string reorganization problems, which occur because garbage collection routines in the Basic monitor must take time out to eliminate some of the overhead produced by string manipulation in order to make room for new text.

Finally, the sorting functions of Infinite Basic are extremely powerful, flexible and fast (30 times faster than the fastest sorts in Basic).

Infinite Basic provides to the Level II or Disk Basic user a wide range of powerful functions (100 in all if you purchase Infinite Business). This is like having a library of tested, well-documented and highly optimized machine language user routines.

Since the Infinite Basic functions are called by Basic-like statements, you don't have to use the awkward and slow USR command every time you need a new function. Furthermore, you add just the Infinite Basic enhancements that you need, making for efficient use of memory.

In a nutshell, Infinite Basic confers upon your Level II ROMs a considerable degree of extensibility. For me this has often eliminated the need to resort to Assembly programming when my application program required specialized, extremely fast subroutines.■

✓See List of Advertisers on page 354

Change the image of Level II Basic by adding to its lexicon.

New Words for Basic

Alan Neibauer
11138 Hendrix Street
Philadelphia, Pa. 19116

If you've ever run into some of the extended Basic features and were unable to find them in your Level II manual, don't give up. This happened to me while working on the DEC computer at LaSalle College in Philadelphia. I noticed some strange statements peeking at me from my neighbor's terminal, but once I saw how they worked I knew there had to be a way to use them on my 80.

Change

Since, there were a few nice programs that I wanted to try on my own machine, I started working out appropriate subroutines. The first add statement I encountered was Change. This usually takes the form: Change X$ to X.

This establishes the array X(n..len n) assigning to each array element the ASCII value of each letter in the string. For example, if the string Micro is changed through the Change A$ to A statement, an array A with six elements is established. Elements A(1) to A(5) are assigned the ASCII codes for the letters M I C R O. In this case, the array is 77, 73, 67, 82, 79. What about the sixth element? The zero subscript A(0) is assigned a number equal to the length of the string. In this case A(0) = 5 (or in Level II, A(0) = Len(A$). Program Listing 1 demonstrates how this would appear in a short program converting a string to its ASCII equivalent and printing the resulting array.

This Change statement also works in reverse: Change A to A$. This converts the ASCII codes back to letters and restuctures the original string. However, the zero subscript is untouched and usually is converted into an individual program line. Program Listing 2 demonstrates this reversal.

By combining the ASC and MID$ functions, you can use a short subroutine whenever Change is called for. By converting each string element to its ASCII code and assigning this code to an array element, the Change function can be simulated on the TRS-80. The same holds true for the reversal. Through the CHR$ function, you can convert the ASCII back to its letter equivalent and print the entire string. (See Program Listing 3.)

The actual Change subroutines are lines 1000-1030 and 2000-2030. This sample program requests a string to be changed. The first subroutine converts A$ into the array A assigning ASCII values to each array element, returning to the program body and printing the results. Line 1000 allows us to loop only the number of times needed to change each element of the string. These elements are pulled apart by the MID$ function on line 1010 and assigned ASCII codes. Line 1030 takes care of the zero subscript.

When the program is sent to the second subroutine, the Change A to A$ function is simulated. Line 2000 establishes the number of loops. Line 2010 converts each array element (remember—an ASCII code) and assigns it to a string array X$. In line 2020, each temporary X$ array is added or linked to form a new string, B$. When the loops are completed, the final B$ contains

```
5 'NEIBAUER,ALAN  *ITS STILL BASIC....*
6 'FIGURE #1, THE CHANGE FUNCTION - DOES NOT RUN ON TRS-80!!!!
10 A$="MICRO"
20 CHANGE A$ TO A
30 FOR X=1 TO A(0)
40 PRINT A(X);
50 NEXT
```

Program Listing 1. The Change Function.

```
5 'NEIBAUER,ALAN *ITS STILL BASIC.....*
6 'FIGURE 2, SIMULATION OF CHANGE A TO A$-DOES NOT RUN ON TRS-80
!!!!
10 CHANGE A TO A$
20 FOR X=1 TO A(0)
30 PRINT A$(X);
40 NEXT
```

Program Listing 2. Change A to A$.

"From now on, the Change function shouldn't scare you away from useful programs."

the original input string A$. This is printed in line 120.

From now on, the Change function shouldn't scare you away from useful programs. Program Listing 4 demonstrates this useful idea in a simple children's game, Hangman.

Image

Another statement is Image. Image, however, is simply a method of formatting a Print Using statement in an individual program line. This could appear as:

```
100 PRINT USING 200,A
200 :AREA IS ##.## SQ. MILES.
```

In this case, line 200 specifies the Print Using format and is referred to in line 100. While Level II Basic does not recognize the Image statement, it does recognize strings. The lines above can easily be converted to TRS-80 language by assigning the format line 200 to a string and making reference to that string in the Print Using line. Of course, the format string would come first:

```
90 A$="AREA IS ##.## SQ. MILES."
100 PRINT USING A$,A
```

If you run across any Image statements, simply convert the Image lines (those starting with the colon) into strings and replace the Image line numbers referred to with the appropriate string.

Matrices

Finally, some extended Basics make great use of multi-dimensional arrays, called matrices. While the Level II manual suggests a number of useful array/matrix manipulation subroutines, these won't help if you come across a Mat Read or Mat Print. However, there is nothing in Mat statements the TRS-80 cannot handle.

The Mat Read statement reads from data a multi-dimensional array. For example:

```
10 DIM A(3,3)
20 MAT READ B
30 MAT PRINT B;
40 DATA 1,2,3,4,5,6,7,8,9
```

would read and print the following matrix:

```
1  2  3
4  5  6
7  8  9
```

We can do the same thing in our own Basic, however, it takes a few more lines. In

```
5 'NEIBAUER,ALAN *ITS STILL BASIC.....*
6 'FIGURE 3 LEVEL 2 SIMULATION OF CHANGE
10 'CHANGE SUBROUTINES - FIRST IS CHANGE A$ TO A
30 INPUT A$
40 GOSUB 1000
50 FOR X=1 TO A(0)
60 PRINT A(X);
70 NEXT
75 PRINT
80 'NEXT IS REVERSAL - CHANGE A TO A$
90 'OUTPUT B$ = INPUT A$
110 GOSUB 2000
120 PRINT B$
130 END
1000 FOR X=1 TO LEN(A$)
1010 A(X)=ASC(MID$(A$,X,1))
1020 NEXT X
1030 A(0)=LEN(A$)
1040 RETURN
1999 END
2000 FOR X=1 TO A(0)
2010 X$(X)=CHR$(A(X))
2020 B$=B$+X$(X)
2030 NEXT X
2040 RETURN
```

Program Listing 3. Level II Simulation of Change.

"You might want to try your own hand at taming foreign Basic statements."

our Basic, each row must be read by column and printed, then the next row, and so on. While this takes a little longer (see Program Listing 5), useful programs having Mat Read and Mat Print statements need not be tossed aside.

You might want to try your own hand at taming foreign Basic statements. The Level II manual tells how to simulate the Instring function found on some computers. But it doesn't go into user-defined functions or margin statements—two perfect places to start working up your own extended Basic subroutines. ■

```
5 'NEIBAUER,ALAN *ITS SILL BASIC....*
6 ' FIGURE 4 -HANGMAN PROGRAM
10 RANDOM
20 DIM D$(100),D(10)
25 CLEAR 200
30 FOR D=1 TO 10
40     READ D$(D)
50 NEXT D
60 I=RND(10)
70 GOSUB 1000          'SIMULATE CHANGE D$(I) TO D
80 FOR Q=1 TO D(0)
90     K(Q)=ASC("-")
100 NEXT Q
120 PRINT@832,"GUESS A LETTER";
130 INPUT A$
140 A=ASC(A$)
150 FOR J=1 TO D(0)
160     IF A<>D(J) THEN 180
170     K(J)=A
180 NEXT J
185 PRINT@ 478," ";
190 FOR Q=1 TO D(0)
210     PRINT CHR$ (K(Q));
220 NEXT Q
230 GOSUB 1500          'SIMULATE CHANGE D TO B$
240 IF B$<>D$(I) THEN B$="":GOTO 120
250 PRINT @832,"GOOD WORK! YOU GUESSED THE WORD"
500 DATA THIS,IS,THE,LOCATION,FOR,WORDS,TO,BE,GUESSED,END
599 END
1000 FOR X=1 TO LEN(D$(I))
1010 D(X)=ASC(MID$(D$(I),X,1))
1020 NEXT X
1030 D(0)=LEN(D$(I))
1040 RETURN
1050 END
1500 FOR X=1 TO D(0)
1510 Y$(X)=CHR$(K(X))
1520 B$=B$+Y$(X)
1530 NEXT X
1540 RETURN
```

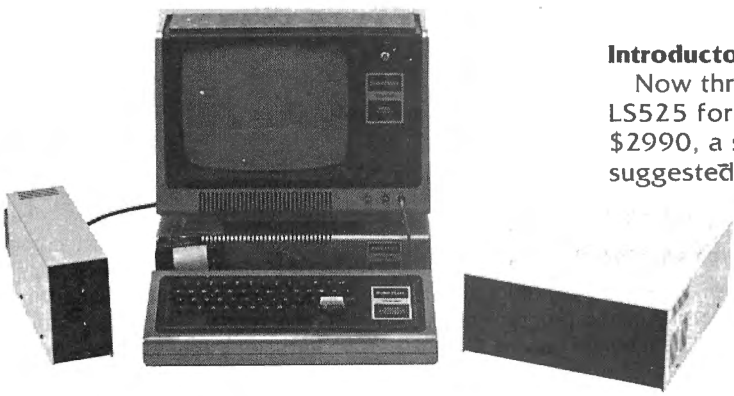
Program Listing 4. Hangman.

```
5 'NEIBAUER,ALAN *ITS STILL BASIC.....*
6 'FIGURE 5 - SIMULATION OF MAT/MAT READ
10 FOR ROW=1 TO 3
20      FOR COL=1 TO 3
30      READ B(ROW,COL)
40      PRINT B(ROW,COL);
50      NEXT COL
60      PRINT             'FORCES CARRIAGE RETURN
70      NEXT ROW
80 DATA 1,2,3,4,5,6,7,8,9
```

Program Listing 5.

THE NEW LAREDO LS525. WHEN YOU'RE READY TO GET DOWN TO BUSINESS.

Laredo Systems, Inc. introduces the LS525 five megabyte rigid disk memory system for TRS80 Models I and III. Laredo's LS525 rounds up a Seagate ST506 Drive, LDOS Operating System by Logical Systems, and Laredo's own LSI

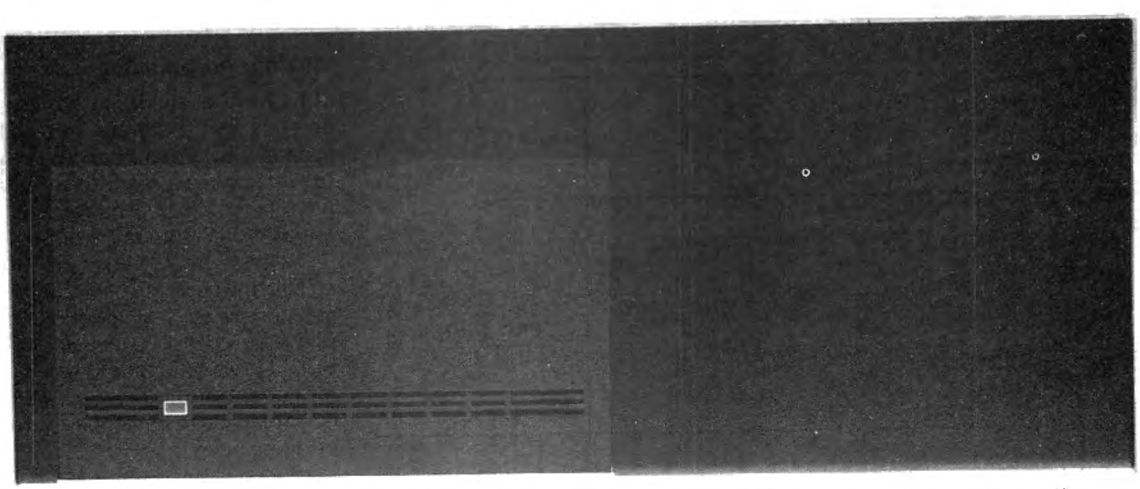


500 Series Controller into a single-board controller that emulates the famed IBM 3370 disk system, complete with:

- Improved read/write/seek access time through full block buffering and variably tuned interleave.
- On-board data separator, micro-code diagnostics and write-precompensation.
- Increased disk media reliability and data integrity with automatic defect block allocation and extended ID fields.
- Versatile CPU compatibility provided by separate host adaptor.

Introductory Offer To TRS80 Users.
Now through October 1, 1981 buy the LS525 for the OEM/dealer price of $2990, a savings of $760 off the usual suggested retail price of $3750.
LDOS Operating System with manual and diskette available for $149.

For more information about the LS525 Memory System and the LSI 500 Series Controller, contact:
Laredo Systems, Inc. ✓26
669 Giraudo Drive, San Jose, CA 95111
(408) 629-2283



laredo systems inc.

✓See List of Advertisers on page 354

An assembly language program to delete spaces and remarks.

The Memory Expander

Tim Kenealy
1235 Folkstone Ct.
Wheaton, IL 60187

Many people are lucky when it comes to programming the TRS-80. Many people are not and receive OM errors. If you type your programs with many spaces and remarks, you might get some too.

Described in this article is an assembly language program that deletes spaces and REMs in a Basic program. It will not delete spaces inside quotes or in data lines. This will reduce a program's size, saving memory, and the program will run faster.

Basic Program Storage

Before I can explain my program, there is some information on program storage in memory to be learned. Look at Fig. 1. The Basic program storage buffer starts at 42E9H, or 17129 decimal. Each line is stored one after the other. After the last line in the program, the variable chart starts.

Now look at Fig. 2. Each line has four bytes preceding the actual data of the line. The first two bytes are the LSB and MSB of the pointer to the next line. If X = the LSB of the pointer and Y = the MSB, then X+256(Y) will give you the start address of the next line.

The last two bytes of the beginning four is the line number (least significant byte first). The above formula will give you the line number.

After the four byte overhead, the actual data comes. All Basic reserved words have a one byte code. This code can be found in the Level II manual. All other characters are stored as their ASCII equivalent. After the line is complete, a zero follows. Then the next line starts. After the end zero of the last line, two more zeros mark the end of the program.

After the program, the variable chart starts. The first part is made up of simple variables, and the second part is made up of arrays. Addresses 16633-16638 decimal hold the pointers to these charts.

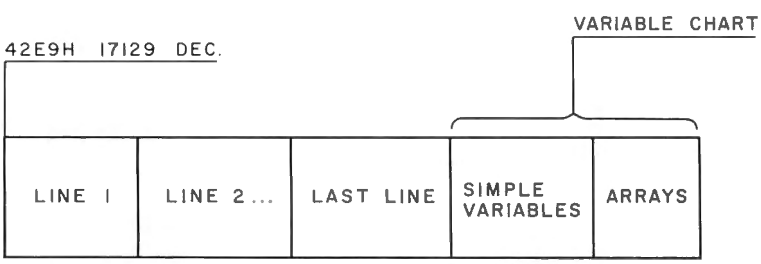


Fig. 1 Basic Program Storage.

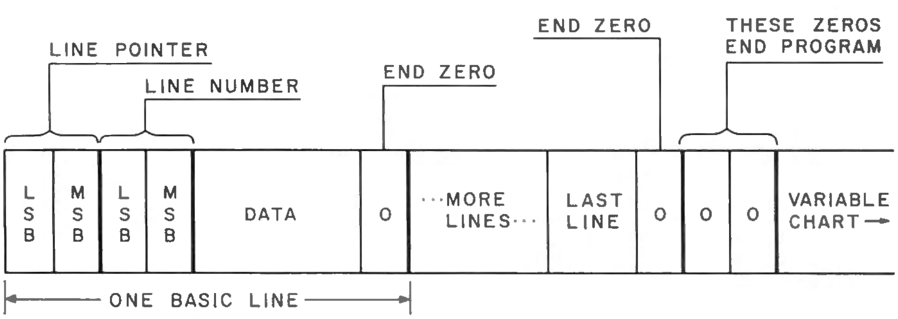


Fig. 2. Basic Line Storage.

What the Program Does

To have a successful space and REM deleter, the program must delete the spaces and remarks and push everything together. My program takes one line at a time, places it in a buffer, reduces it and brings it back into the Basic buffer.

Let's go through the program step by step. In the beginning, the screen is cleared, and a message is displayed on the screen. BUFF1 and BUFF2 are two-byte buffers. BUFF1 always points to where in memory the current line stored in the work buffer will be replaced. BUFF2 points to where the next line up in memory is.

BUFF1 is initialized at 42E9H, because that's where the basic buffer begins. BUFF2 is initialized in lines 170–180. At line 190, HL is set at 42E9H, the first byte of the first line.

Starting at line 210, the line to be reduced is put in a work buffer. The first four bytes are put in with an LDIR, because that is the

"When the Assembly program is finished... all spaces and remarks will be gone, but spaces in quotes and data lines will still be there."

overhead and may contain zeros. From lines 240 to 300, the rest of the line is put in the work buffer.

Now the line is reduced. BC is set at zero, since it is used later. HL is loaded with the first byte after the line overhead.

The first byte is checked for a 147, the code for REM, because if the line is deleted, the line number will be left, unless the program deletes it too.

Starting at line 360, a series of checks are made of the byte. When the final zero is found, the line is totally reduced. If none of the checks are true, HL is incremented, and the next byte is checked.

Let's go through each routine. At line 510 is the data routine. The C register is loaded with a one to tell the program that it's in a data line. That way, no spaces will be deleted while C equals one. HL is incremented, and the next byte is checked.

At line 540 is the colon routine, which is used only to get out of a data line. First, the routine checks if the line is in data form (if C register is one). If it is not, then the matter is dropped. HL is incremented, and the next byte is checked.

If the program is in a data line, the B register is checked for a one. This means that the colon is in quotes. If it does equal one, then the program goes back to the checking routine.

If C is a one, and B is a zero, then the data line is over. C is loaded with zero to tell the program that there is no more data.

Starting at line 630 is the quote routine. All it does is load B with zero if it's one or load B with one if it's zero. B equals one if the line is in quotes at the time. HL is incremented and the next byte is checked.

At line 710 is the space routine. First, the routine checks if the space is inside quotes, using the B register. If so, then HL is incremented and the next byte is checked.

If the space is not in quotes, then the routine checks to see if it is in a data line. If it is, HL is incremented and the next byte is checked.

If neither is true, then the space must be deleted. HL is saved on the stack and loaded onto IX. Lines 820 to 880 delete the space by pulling all following bytes back one slot, demolishing the space. HL is then restored, and the program goes back to the checking routine.

At line 910 is the REM routine. Since most programs have a ":REM" at the end of a line, the colon must be deleted also. HL is decremented to point to where the colon would be. The test is made and if it is a colon, the routine goes back to see if there is another.

When all colons are found, the program places a zero to mark the end of the line. When the end zero is encountered in the checking routine or at the end of the REM routine, the program goes to line 970.

When the program gets to line 970, the Basic line is totally reduced. BUFF1 points to where in memory this line will be replaced. BUFF2 points to where the next line in memory begins.

First, the line is copied back into memory, the overhead with an LDIR and the rest from lines 1010 to 1070. After the line is recopied, DE is incremented to point to where the next line will start when it's reduced.

Now, HL is loaded with the first byte of the current line. The bytes at (HL) and (HL+1) are the pointers to the next line. Since the line has been changed, the pointers must be set. That is done at lines 1100 to 1120.

HL is loaded with the address of where the next line starts in memory. The first two bytes of the line would be the pointer. If these two bytes are zeros, the program is over.

HL is incremented to point to the MSB of the pointer. I only check that byte for a zero, because it alone wouldn't be zero unless the program was over.

At line 1180, BUFF1 is set up for the next line. Then BUFF2 is set up, one byte at a time. HL points to the first byte of the next line, and everything is all set to reduce a new line. At line 1260, a jump is made to reduce the new line.

At line 1270 is the Allrem routine. This loads DE with the address stored in BUFF1. Then the program goes back to line 1130 to test for the end of the program, and so on.

The program goes to line 1290 when the whole Basic program is reduced. The two ending zeros are placed. Then lines 1350 1370 set up the variable chart. Finally, the program jumps back to Basic at line 1380.

How to Use the Program

Type in the program with the editor/assembler. The program, as is, is for 16K. It is fully relocatable, though. For 4K change the ORG and End addresses to 4DF2H. For 32K, change them to 0BDF2H. For 48K, 0FDF2H.

When you're ready to use the program, power up the computer. Answer the Memory Size question with 19953 for 4K, 32241 for 16K, 48625 for 32K, and 65009 for 48K.

Type "SYSTEM (enter)". Then type the file name you used. After loading is finished, hit the Break key. Now you can type in or load a Basic program.

When you are ready to delete spaces and REMs from the program, type "SYSTEM", then type a / followed by the starting address. For 4K, the address is 19954. For 16K, it's 32242. For 32K, the address is 48626, and for 48K, 65010.

When the assembly language program is finished, List your program. All spaces and REMs will be gone, but spaces in quotes and in data lines will still be there.

One final note: If a whole line is a remark but you used the apostrophe, a line number will remain with nothing after it. Just look through your list, and delete those lines. ■

Program Listing 1.

```
7DF2          00100          ORG     7DF2H
7DF2 CDC901   00110          CALL    1C9H             ;CLEAR SCREEN
7DF5 21E17E   00120          LD      HL,MSSG          ;MESSAGE
7DF8 CDA728   00130          CALL    28A7H            ;DISPLAY IT
              00140 ;SET UP BUFF1 AND BUFF2
7DFB 21E942   00150          LD      HL,42E9H         ;BASIC BUFFER
7DFE 22FD7E   00160          LD      (BUFF1),HL       ;SAVE IN BUFF1
7E01 2AE942   00170          LD      HL,(42E9H)       ;LINE POINTER
7E04 22FF7E   00180          LD      (BUFF2),HL       ;SAVE IN BUFF2
7E07 21E942   00190          LD      HL,42E9H         ;BASIC BUFFER
              00200 ;COPY BASIC LINE INTO INPUT/OUTPUT BUFFER
7E0A 11017F   00210 COPY     LD      DE,BUFFER        ;STORAGE BUFFER
7E0D 010400   00220          LD      BC,4             ;4 BYTES
7E10 EDB0     00230          LDIR                     ;TRANSFER THEM
7E12 7E       00240 MORE     LD      A,(HL)           ;BYTE TO TRANSFER
7E13 12       00250          LD      (DE),A           ;TRANSFER IT
7E14 B7       00260          OR      A                ;IS IT ZERO?
7E15 2804     00270          JR      Z,DONE           ;GO IF DONE
7E17 23       00280          INC     HL               ;BUMP --
7E18 13       00290          INC     DE               ;POINTERS
7E19 18F7     00300          JR      MORE             ;GET NEXT BYTE
7E1B 010000   00310 DONE     LD      BC,0             ;INITIALIZE BC REG.
7E1E 21057F   00320          LD      HL,BUFFER+4      ;PAST 4 BYTES
7E21 7E       00330          LD      A,(HL)           ;BYTE TO CHECK
7E22 FE93     00340          CP      147              ;LINE ALL REM?
7E24 CAC97E   00350          JP      Z,ALLREM         ;TO ROUTINE
7E27 7E       00360 CHECK    LD      A,(HL)           ;BYTE TO CHECK
7E28 FE88     00370          CP      136              ;IS IT DATA LINE?
7E2A 2816     00380          JR      Z,DATA           ;TO ROUTINE
7E2C FE3A     00390          CP      3AH              ;IS IT A COLON?
7E2E 2817     00400          JR      Z,COLON          ;TO ROUTINE
7E30 FE22     00410          CP      22H              ;IS IT A QUOTE?
7E32 2822     00420          JR      Z,QUOTE          ;TO ROUTINE
7E34 FE20     00430          CP      20H              ;IS IT A SPACE?
```

Program continues

Program continued

```
7E36 282C      ØØ44Ø         JR     Z,SPACE     ;TO ROUTINE
7E38 FE93      ØØ45Ø         CP     147         ;IS IT A REM?
7E3A 284C      ØØ46Ø         JR     Z,REM       ;TO ROUTINE
7E3C B7        ØØ47Ø         OR     A           ;IS IT ZERO?
7E3D 2852      ØØ48Ø         JR     Z,FINISH    ;TO ROUTINE
7E3F 23        ØØ49Ø         INC    HL          ;BUMP POINTER
7E4Ø 18E5      ØØ5ØØ         JR     CHECK       ;CHECK NEXT BYTE
7E42 ØEØ1      ØØ51Ø DATA    LD     C,1         ;SET POINTER
7E44 23        ØØ52Ø         INC    HL          ;BUMP POINTER
7E45 18EØ      ØØ53Ø         JR     CHECK       ;NEXT BYTE
7E47 79        ØØ54Ø COLON   LD     A,C         ;LOAD A WITH POINTER
7E48 FEØ1      ØØ55Ø         CP     1           ;IN DATA LINE?
7E4A 2ØØ7      ØØ56Ø         JR     NZ,NODATA   ;NOT IN DATA LINE
7E4C 78        ØØ57Ø         LD     A,B         ;LOAD A WITH POINTER
7E4D FEØ1      ØØ58Ø         CP     1           ;INSIDE QUOTES?
7E4F 28Ø2      ØØ59Ø         JR     Z,NODATA    ;INSIDE QUOTES
7E51 ØEØØ      ØØ6ØØ         LD     C,Ø         ;OUT OF DATA NOW.
7E53 23        ØØ61Ø NODATA  INC    HL          ;BUMP POINTER
7E54 18D1      ØØ62Ø         JR     CHECK       ;CHECK NEXT BYTE
7E56 78        ØØ63Ø QUOTE   LD     A,B         ;LOAD B ONTO A
7E57 FEØ1      ØØ64Ø         CP     1           ;SECOND QUOTE?
7E59 28Ø4      ØØ65Ø         JR     Z,SECOND    ;TO ROUTINE
7E5B Ø6Ø1      ØØ66Ø         LD     B,1         ;FIRST QUOTE
7E5D 18Ø2      ØØ67Ø         JR     CONT        ;CONTINUE
7E5F Ø6ØØ      ØØ68Ø SECOND  LD     B,Ø         ;NO QUOTES NOW
7E61 23        ØØ69Ø CONT    INC    HL          ;BUMP POINTER
7E62 18C3      ØØ7ØØ         JR     CHECK       ;CHECK NEXT BYTE
7E64 78        ØØ71Ø SPACE   LD     A,B         ;LOAD B ONTO A
7E65 FEØ1      ØØ72Ø         CP     1           ;INSIDE QUOTES?
7E67 2ØØ3      ØØ73Ø         JR     NZ,OUT      ;NOT IN QUOTES
7E69 23        ØØ74Ø INDATA  INC    HL          ;BUMP POINTER
7E6A 18BB      ØØ75Ø         JR     CHECK       ;CHECK NEXT BYTE
7E6C 79        ØØ76Ø OUT     LD     A,C         ;LOAD A WITH POINTER
7E6D FEØ1      ØØ77Ø         CP     1           ;IN DATA LINE?
7E6F 28F8      ØØ78Ø         JR     Z,INDATA    ;BACK SOME
7E71 E5        ØØ79Ø         PUSH   HL          ;SAVE IT FOR NOW
7E72 E5        ØØ8ØØ         PUSH   HL          ;LOAD HL --
7E73 DDE1      ØØ81Ø         POP    IX          ;ONTO IX
7E75 DD7EØØ    ØØ82Ø LOOP    LD     A,(IX)      ;CHECKING --
7E78 B7        ØØ83Ø         OR     A           ;FOR ZERO
7E79 28ØA      ØØ84Ø         JR     Z,DONE2     ;GO IF ZERO
7E7B DD7EØ1    ØØ85Ø         LD     A,(IX+1)    ;THIS BYTE IS --
7E7E DD77ØØ    ØØ86Ø         LD     (IX),A      ;BROUGHT BACK ONE
7E81 DD23      ØØ87Ø         INC    IX          ;BUMP POINTER
7E83 18FØ      ØØ88Ø         JR     LOOP        ;NEXT BYTE
7E85 E1        ØØ89Ø DONE2   POP    HL          ;RESTORE
7E86 189F      ØØ9ØØ         JR     CHECK       ;CHECK NEXT BYTE
7E88 2B        ØØ91Ø REM     DEC    HL          ;CHECK FOR --
7E89 7E        ØØ92Ø         LD     A,(HL)      ;A COLON
7E8A FE3A      ØØ93Ø         CP     3AH         ;IS IT?
7E8C 28FA      ØØ94Ø         JR     Z,REM       ;CHECK FOR ANOTHER
7E8E 23        ØØ95Ø         INC    HL          ;NO COLON
7E8F 36ØØ      ØØ96Ø         LD     (HL),Ø      ;ENDING ZERO
7E91 ED5BFD7E  ØØ97Ø FINISH  LD     DE,(BUFF1)  ;SET UP TO --
7E95 21Ø17F    ØØ98Ø         LD     HL,BUFFER   ;RECOPY LINE --
7E98 Ø1Ø4ØØ    ØØ99Ø         LD     BC,4        ;INTO BASIC BUFFER
7E9B EDBØ      Ø1ØØØ         LDIR               ;COPY 4 BYTES
7E9D 7E        Ø1Ø1Ø MORE2   LD     A,(HL)      ;BYTE TO COPY
7E9E 12        Ø1Ø2Ø         LD     (DE),A      ;COPY IT
7E9F B7        Ø1Ø3Ø         OR     A           ;LAST BYTE?
7EAØ 28Ø4      Ø1Ø4Ø         JR     Z,FINIS     ;GO IF DONE
7EA2 23        Ø1Ø5Ø         INC    HL
7EA3 13        Ø1Ø6Ø         INC    DE          ;BUMP POINTERS
7EA4 18F7      Ø1Ø7Ø         JR     MORE2       ;COPY ANOTHER BYTE
7EA6 13        Ø1Ø8Ø FINIS   INC    DE          ;NEXT LINE WILL BE HERE
7EA7 2AFD7E    Ø1Ø9Ø         LD     HL,(BUFF1)  ;BEGINNING OF LINE
7EAA 73        Ø11ØØ         LD     (HL),E      ;LSB OF LINE POINTER
7EAB 23        Ø111Ø         INC    HL          ;BUMP HL
7EAC 72        Ø112Ø         LD     (HL),D      ;MSB OF LINE POINTER
7EAD 2AFF7E    Ø113Ø ETC     LD     HL,(BUFF2)  ;NEXT LINE IN MEM
7EBØ 23        Ø114Ø         INC    HL          ;POINT TO MSB
7EB1 7E        Ø115Ø         LD     A,(HL)      ;LOAD BYTE TO TEST
7EB2 B7        Ø116Ø         OR     A           ;PROGRAM DONE?
7EB3 281A      Ø117Ø         JR     Z,PRGEND    ;GO IF DONE
7EB5 ED53FD7E  Ø118Ø         LD     (BUFF1),DE  ;SET UP BUFF1
7EB9 2AFF7E    Ø119Ø         LD     HL,(BUFF2)  ;NEXT LINE IN MEM
7EBC 7E        Ø12ØØ         LD     A,(HL)      ;LSB OF POINTER
7EBD 32FF7E    Ø121Ø         LD     (BUFF2),A   ;SAVE IN BUFF2
7ECØ 23        Ø122Ø         INC    HL          ;BUMP HL
7EC1 7E        Ø123Ø         LD     A,(HL)      ;MSB OF POINTER
7EC2 32ØØ7F    Ø124Ø         LD     (BUFF2+1),A ;SAVE IN BUFF2+1
7EC5 2B        Ø125Ø         DEC    HL          ;DOWN AGAIN
7EC6 C3ØA7E    Ø126Ø         JP     COPY        ;NEW LINE
7EC9 ED5BFD7E  Ø127Ø ALLREM  LD     DE,(BUFF1)  ;SET UP DE
7ECD 18DE      Ø128Ø         JR     ETC         ;GO TO ROUTINE
7ECF AF        Ø129Ø PRGEND  XOR    A           ;LOAD A WITH ZERO
7EDØ 12        Ø13ØØ         LD     (DE),A      ;FIRST END ZERO
7ED1 13        Ø131Ø         INC    DE          ;BUMP POINTER
7ED2 12        Ø132Ø         LD     (DE),A      ;SECOND END ZERO
7ED3 13        Ø133Ø         INC    DE          ;BUMP POINTER
7ED4 EB        Ø134Ø         EX     DE,HL       ;LOAD DE ONTO HL
7ED5 22F94Ø    Ø135Ø         LD     (16633),HL  ;LOAD --
7ED8 22FB4Ø    Ø136Ø         LD     (16635),HL  ;VARIABLE POINTERS --
7EDB 22FD4Ø    Ø137Ø         LD     (16637),HL  ;WITH HL
7EDE C3CCØ6    Ø138Ø         JP     6CCH        ;BACK TO BASIC
7EE1 44        Ø139Ø MSSG    DEFM   'DELETING SPACES AND REMS --'
7EFC ØØ        Ø14ØØ         NOP                ;END OF MESSAGE
7EFD ØØØØ      Ø141Ø BUFF1   DEFW   Ø           ;2 BYTE BUFFER
7EFF ØØØØ      Ø142Ø BUFF2   DEFW   Ø           ;2 BYTE BUFFER
ØØFF           Ø143Ø BUFFER  DEFS   255         ;STORAGE BUFFER
7DF2           Ø144Ø         END    7DF2H
ØØØØØ TOTAL ERRORS
```

Orange Micro

"THE COMPUTER PRINTER SPECIALISTS"

UP TO 25% DISCOUNTS! — SAME DAY SHIPMENT!

NEW CENTRONICS 739 (RADIO SHACK LINE PRINTER IV)

With Graphics and Word Processing Print Quality

• 18 x 9 dot matrix; suitable for word processing • Underlining • proportional spacing • right margin justification • serif typeface • 80/100 CPS • 9½" Pin Feed/Friction feed • Reverse Platen • 80/132 columns • Top of form

CENTRONICS 739-1 (Parallel)................(List $995) $ Call
CENTRONICS 739-3 (Serial)..................(List $1045) $ Call
GRAPPLER™ Apple graphics interface $165

ANADEX

Dot Graphics, Wide Carriage

• 11 x 9 dot matrix; lower case descenders • Dot resolution graphics • Bi-directional, logic seeking • Up to 200 CPS • RS 232 Serial & Parallel • Forms control • X-ON/X-OFF • Up to 6 part copy.

ANADEX 9501 ....(List $1650) $1350
GRAPPLER™ Apple graphics Interface $165

VISTA — C. ITOH STARWRITER

Daisy Wheel Letter Quality

• 25 CPS (Optional 45 CPS) • Typewriter quality • Centronics parallel • RS 232 Serial (Optional) • Proportional spacing • Bidirectional • Programmable VFU • Self test • Diablo compatible • Friction feed (Optional tractors) • 136 printable columns • Manufactured by C. ITOH.

VISTA V300 (C. ITOH) STARWRITER ....(List $1895) $ Call

IDS PAPER TIGERS

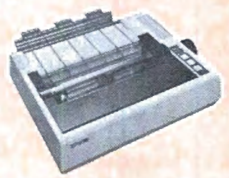
Dot Resolution Graphics, quality print, speed

• 7 wire printhead (445); 9 wire printhead (460) with lower case descenders • Over 150 CPS • bi-directional, logic seeking (460) • 8 character sizes; 80-132 columns • Adjustable tractors • High-resolution dot graphics • Proportional spacing & text justification (460).

IDS 460G 9 wire printhead, graphics ....(List $1394) $1150
IDS 560G 9 wire, wide carriage, graphics ......(List $1794) $1400
GRAPPLER™ Apple graphics interface $165

EPSON MX80/MX70

Low-Priced Professional Print Quality



• 9 x 9 dot matrix • Lower case descenders • 80 CPS • Bidirectional, Logic seeking • 40, 66, 80, 132 columns per line • 64 special graphic characters: TRS-80 Compatible • Forms handling • Multi-pass printing • Adjustable tractors

We also carry a full line of Epson Accessories.

EPSON MX80 ....(List $645) $Call
EPSON MX 70 Dot graphics, 5 x 7 matrix.........(List $450) $Call
EPSON MX100 wide carriage.....................(List $995) $Call
GRAPPLER™ Apple Graphics Interface.................... $165

MX80/70 FRICTION FEED KIT

User installable kit for single sheets.
Easy 30 minute installation .... $ 75

ANACOM

Low Cost, High Speed, Wide Carriage

• 9 x 9 dot matrix • Lower case descenders • Wide carriage • Adjustable tractors to 16" • 150 CPS, Bidirectional, Logic Seeking

ANACOM 150 ....(List $1350) $1075

NEC SPINWRITER

High Speed Letter Quality

• 55 CPS • Typewriter quality • Bidirectional • Plotting • proportional spacing.

5510-5 RO, Serial, w/tractors ....(List $2995) $2675
5530-5 RO, Parallel, w/tractors.................(List $2970) $2650

TELEVIDEO CRT'S

AT DISCOUNT PRICES!

TVI 910
TVI 912C
TVI 920C
TVI 950
} Please Call Toll Free. Prices are too low to advertise

QUANTITY PRICING AVAILABLE

CRT'S

ADDS VIEWPOINT ....(List $698) $600

PRINTERS

MALIBU 165 wide carriage, graphics, letter quality ...(List $2495) $ 1975
QUME 5/45 typewriter quality ....(List $2905) $ 2559

INTERFACE EQUIPMENT

EPSON ACCESSORIES.................................... $ Call
ORANGE INTERFACE for Apple II
parallel interface board & cable .... $ 110
MICROTRONICS Atari parallel interface..................... $ 69
TRS-80 CABLES to keyboard or Exp. interface .... $ Call
NOVATION D-CAT direct connect modem.................... $ Call

CALL FOR INFORMATION & CATALOG

(800) 854-8275

CA, AK, HI (714) 630-3322

RETAIL PRINTER STORE

Store #1	3150 E. La Palma, #I, Anaheim, CA	(714) 630-3622
Store #2	13604 Ventura Bl., Sherman Oaks, CA	(213) 501-3486
Store #3	COMING SOON ??	

Store Hours: M-F 10-6 Sat. 10-4

Orange Micro inc. ✓296

3150 E. La Palma, Suite G, Anaheim, CA 92806

Phone order WELCOME; same day shipment. Free use of VISA & MASTER-CARD. COD's accepted. Manufacturer's warranty included on all equipment. Prices subject to revision.

At Orange Micro, we try to fit the right printer to your application. Call our printer specialists for free consultation.

Yes, You Can Increase Your Programming Productivity

SNAPP II EXTENDED BASIC

SNAPP II EXTENDED BASIC A family of enhancements to the Model II BASIC interpreter. Part of the package originated with the best of APPARAT, INC.'s thoughts in implementing NEWDOS BASIC. The system is written entirely in machine language for SUPER FAST execution. The extensions are fully integrated into Model II BASIC, and require NO user Memory, and NO user disk space. The package is made up of the following six modules.

XBASIC—Six single key stroke commands to list the first, last, previous, next or current program line, or to edit the current line. Includes quick way to recover BASIC program following a NEW or system or accidental re-boot. Ten single character abbreviations for frequently used commands: AUTO, CLS, DELETE, EDIT, KILL, LIST MERGE, NEW, LLIST, and SYSTEM.

XREF—A powerful cross-reference facility with output to display and/or printer. Trace a variable through the code. Determine easily if a variable is in use.

XDUMP—Permits the programmer to display and/or print the value of any or all program variables. Identifies the variable type for all variables. Each element of any array is listed separately.

XRENUM—An enhanced program line renumbering facility which allows specification of an upper limit of the block of lines to be renumbered, supports relocation of renumbered blocks of code, and supports duplication of blocks of code.

XFIND—A cross reference facility for key words and character strings, also includes global replacement of keywords.

XCOMPRESS—Compress your BASIC programs to an absolute minimum Removes extraneous information; merge lines; even deletes statements which could not be executed. Typically saves 30-40% space even for programs without our REM statements! Also results in 7-10% improvement in execution speed.

ENTIRE PACKAGE FOR MODEL II $200
ENTIRE PACKAGE FOR MODEL III $125

SNAPP SCREEN

Another dependable product from Snapp, Inc. Using rigid plastic, Snapp, Inc. designed the ultimate green screen at the lowest price. The Snapp Screen installs in a Snapp with the use of pressure sensitive tape. Reduce user fatigue with the Snapp Screen.

DOSFIX $15

A collection of patches to TRSDOS and BASIC to enhance their usability and function. Includes our well known BREAK7E patches and facilities to disable and verify detect which will increase average disk speed by 30%. Free with any Model II Software Package. Purchased separately $15

SUPERSNAPP X

You can sort fast using your present facilities, or you can do it faster with Racets' superb facilities, or you can use the fastest: SUPERSNAPP X. The heart of SUPERSNAPP X is a SUPER FAST in-memory sort routine that has been benchmarked against everything on the market and beats them all . . . hands down.

SUPERSNAPP X is the most important component of SNAPP X EXTENDED BUILTIN FUNCTIONS which is a much needed set of additions to the Model II BASIC interpreter that will put time saving power at your fingertips. Let's compare (using random data) SUPERSNAPP X and Racet's GSF SORT for speed:

SORT	SUPERSNAPP X	RACET GSF
10,000 integers	39 seconds	59 seconds
5,000 Singles	22 seconds	34 seconds
2,000 Strings	10 seconds	15 seconds

SUPERSNAPP X is guaranteed to be the FASTEST in memory SORT on the market or your money back. With it you also get these EXTENDED BUILTIN FUNCTIONS: PEEK, PEEKW, POKE, POKEW, XDAT$, XTIM$, ETIM$, FILES, AND THE SPECIAL SCMD (SNAPP___COMMAND). PLUS: open "E" Set SCROLL PROTECTION. ERASE all ARRAYS in one command. Specify size and Blink rate of CURSOR. LONG ERROR MESSAGES. Read from Video Screen Read.

PEEK complete strings from memory. POKE complete strings to memory, convert upper case to lower case and vice-versa, turn complete screen off and on at will, extract largest or smallest values from user supplied list of numbers

We consider this next function as probably the most powerful addition ever made to Microsoft BASIC. **PRINT USING INTO A STRING.** The ability to arrange data into a string variable should perk your imagination.

Model II $100.00
Model III $75.00

SPOOLER Model I, Model II and Model III

Our workhorse! Unlike the one supplied with TRSDOS 2.0, ours requires no special knowledge or training on the part of the operator. Additionally, ours performs much better. On the Tandy SPOOLER, every time a disk is accessed, the printer stops dead! This package is available for Model I, in the TRSDOS/NEWDOS 80 versions, or for the Model II or Model III. Greatly enhances system performance when running typical business applications. Many applications have been benchmarked to run nearly TWICE AS FAST with the SPOOLER installed. Installs in minutes and no changes are required to your programs. The Model II version requires NO user memory. $100

SORTDIR

Locate directory entries with ease. SORTDIR will rearrange your directory entries in alphameric sequence on either a TRSDOS operating system or data diskette The routine performs its function in seconds. No more wasted time hunting for your entries. Another time saver from SNAPPWARE 30.00

XSCREEN

Supports the copying of the full video screen to the printer. Can be invoked by the operator with a keystroke, or from your program with a USR call. Requires NO user memory $30

AUTOFILE

Another big time saver from Snapp, Inc. Autofile offers all programmers much needed time saving tools to use when working with direct files. Eliminate tedious commands such as LSET's, MKI$'s, MKD$'s, MKS$'s, CVD's, CVI's and CVS's. Even the FIELD statement has been eliminated. You won't have to guess as to where the FIELDed variable is. The ASC and CHR$ function references will be performed automatically.

MODEL II $75
MODEL III $60

AUTOMAP

Save time creating a formated screen with our Extended Basic Mapping Support system. The programmers task of communicating and displaying information with the user operator is completely automated with simple SEND and RECEIVE statement commands. Many different types of information can be sent and received from the operators with the use of a single command. Many hours of programming time are saved and cost efficiency is truly realized using Automap.

MODEL II $100
MODEL III $75

THE GARBAGE COLLECTOR

By using a "variable length string" in the BASIC interpreter, the strings are constantly being relocated in a "string pool" giving them a new location each time they are assigned a new value. Periodically the "string pool" must be reorganized and condensed into a single contiguous area. Performing this string space reclamation function is crude and time consuming because of space limitations on hardware. The system seems to "lock up" and will not respond to the operator at all until the process is complete. The time required to perform string space reclamation is roughly proportional to the square of the number of active strings in the resident program. This time consuming task cries out for a better solution. Snappware has developed a solution which takes advantage of the auxiliary memory available when needed. Our enhanced version requires only four bytes per active string as a work area, and when the amount of space is available as "free storage", will temporarily borrow that space and return it to the free storage pool when completed. If storage is not available, our system will temporarily transfer out to disk enough of the BASIC program to make room for our work area and return the "paged out" information to its correct location when completed.

In almost every conceivable situation, our method produces significantly enhanced performance when compared to the Microsoft approach. Benchmarked times show in some situations our enhanced routine performs one hundred times as fast as the original approach. $100.00

PL/B

The most powerful logical components have been selected from the worlds most powerful language to give you the most unbeatable program development tool to improve your programming productivity! Snappware has merged the most useful components of BASIC with the power of PL/I. An unbeatable structured programming facility resulted that can increase programmers productivity up to 50%. Unlike competitive products, which are slow rigid preprocessors' representing a primitive attempt to bring Structured Programming concepts to the Micro-Based user, PL/B is today's fourth generation answer to a fourth generation problem! PL/B can give you triple the power of the earlier preprocessors at triple the speed. PL/B is fully integrated into the BASIC interpreter. Two modes of operation are available. Transparent (hidden) or translate and save mode. Increase your productivity and save time with PL/B. $200.00

XPRINT

Print neatly formated hard copy listings of BASIC programs from disk. Programs may be ASCII or compressed. Quick and easy group selection allows you to print many listing with one command. $35.

BUGZAP

A powerful utility oriented toward the machine language programmer. Display/Modify/Print/Memory/Disk sectors Use this to help you learn more about the internals of the Model II. $50.

3M SCOTCH DISKETTES

Double density certified 8" Floppies for the Model II. Better quality is not available at any price. Ten diskettes to a box.

Quantity (boxes)	Price Per/Box
1	$34.50
5	$33.50
10	$32.50

HEAD CLEANERS. The wet process cleaners causes less wear on your drive heads. Package of two including fluid. 25.00

Authorized Distributor Information Processing Products 3M

TERMS OF SALE

Shipments normally made within one day of receipt of your order. Customer normally pays shipping cost, except pre-paid orders including software, in which case shipping cost is absorbed by Snapp, Inc. C.O.D. orders will have standard C.O.D. fee added to shipping charge. Net 10 days will be granted to governmental agencies, educational institutions and well rated business firms. Please include purchase order number. Ohio residents add 5 ½% sales tax or exemption certificate.

OUR GUARANTEE:

If your diskette arrives damaged, we will replace it without charge. If you ever accidentally damage it, we will replace it for a $15 handling charge. For a period of one year, we will provide you with any enhancements or updates for a $15 handling charge. For a period of one year, if errors are discovered in the programs, they will be corrected without charge. In the event we cannot correct an error, you may return the program material for a refund.

TRS-80 and TRSDOS are trademarks of the Radio Shack division of Tandy Corporation. NEWDOS and NEWDOS/80 are trademarks of Apparat, Inc. RACET GSF is a trademark of Racet Computes.

SNAPPWARE
SNAPPWARE
SNAPPWARE
SNAPPWARE
SNAPPWARE
SNAPPWARE

Call our toll free number: 1-800-543-4628
Ohio residents call collect: (513) 891-4496

SNAPPWARE
3719 Mantell
Cincinnati, Ohio 45236

Read this and you'll be up on down-loading.

Smart Terminal Software

Smart80d
The Microperipheral Corp.
Mercer Island, WA
$75.
ST80-III
Small Business Systems Group
Westford, MA
$150

Bill Everett
14645 NE 34th C-24
Bellevue, WA 98007

Smart80d and ST80-III are smart terminal programs that, in conjunction with a modem, allow a TRS-80 (with a disk drive) to access another computer by telephone or radio. They are the best two programs I have seen on the market. Both are easy to use, well-documented, and readily adaptable to the various communication requirements of the system to be accessed.

A terminal program gives the TRS-80 the ability to communicate with another computer on terminal. A smart terminal program also allows the transfer of data or programs both to and from the other terminal or computer. This is called up-loading and down-loading respectively.

The versions of the programs that I am reviewing are for use with the Micro Connection direct connect modem. Both programs are also available for use with an RS232 modem. The RS232 versions are almost identical in their use and commands.

Features

I have divided the features of Smart80d and ST80-III into the following categories: communication parameters, receiving data, transmitting data, log-on, and miscellaneous functions.

The communication parameters of both programs can be set to conform to the requirements of the host computer. Full or half-duplex transmission is software selectable with both programs. The number of bits per word, the number of stop bits, and even, odd, or no parity can be selected with Smart80d but only with the RS232 version of ST80-III. In addition ST80-III allows the user to both send and receive a line feed after a carriage return and to send a predetermined number of nulls after a carriage return.

"A terminal program gives the TRS-80 the ability to communicate with another computer on terminal."

Both Smart80d and ST80-III have the ability to store what is seen on the screen to a buffer in memory and then, on command, dump the buffer to a file on disk. The buffer can be opened and closed automatically by both programs. This feature is the heart of any smart terminal program. It allows a user to down-load programs from bulletin boards or any computer system that can be accessed by telephone.

Smart80d and ST80-III can send text or programs to another terminal or computer. The file to be sent is loaded from disk to the buffer area in memory and then sent on command. Smart80d also can automatically open and close the receiving terminal's buffer if it is equipped for auto receive. Both programs have three other functions in common. One is the ability to route everything that goes to the screen to a printer. Another is the ability to temporarily exit the program to execute a DOS command. The last function both programs have in common is the ability to transmit a true break code.

Messages On Command

Smart80d has two buffers in which a log-on or any other message can be stored. The two buffers' contents then can individually be sent on command to log a user onto a bulletin board. ST80-III loads its log-on buffer from a table that is stored on disk. The message can also be transmitted on command.

ST80-III can display all of its commands on the screen. It also can be programmed so any key can send any code. All incoming codes can be changed to another single code for proper control of the terminal or computer. This is useful when matching the protocol of the TRS-80 with that of another system. ST80-III has two features that are unique to terminal programs that I have seen. First it allows use of the system clock to keep track of the time a user has been logged on a bulletin board. It also will allow an amateur radio operator to monitor a frequency for his call sign or any character string for which ST80-III is programmed.

When the character string is decoded, ST80-III comes to life and prints everything until it is deactivated again. This allows unattended monitoring of a radio frequency so the user receives all personal messages.

Utilities

Both Smart80d and ST80-III have the following two similar utility programs on their disk. The first creates a message for uploading to a bulletin board. The other takes a Basic program that has been downloaded and allows the user to delete extraneous carriage returns and lines that are not needed in the Basic program. Both programs do the same job, but the ones that come with Smart80d have better prompts on the screen. Both programs also have a utility list of all possible 255 codes that can be sent or received in decimal and their definitions. The Smart80d version also gives the codes' hex equivalent and its use in the TRS-80.

ST80-III also has the following utilities on disk: A program that creates a checksum for any file, which is used to see that a file has been received correctly; a program to generate the table that is loaded with ST80-

"Smart 80d and ST80-III do an excellent job of making the TRS-80 a smart terminal."

III contains the log-on message generator and can redefine control codes to eliminate possible conflicts between systems; and a program to convert a binary file to ASCII so it can be sent, and another to reverse the process when a binary file is received. These last two programs are also capable of scrambling and unscrambling a file for a security transmission. There are also two programs to send and receive machine code programs in Intel Paper Tape Format. mat.

Documentation

The documentation provided with both programs is excellent. The Smart80d manual goes into more detail on the actual use of the software and the ST80-III manual gives better examples of how it accesses various bulletin boards, The Source, and MicroNet. The documentation for Smart80d also comes on the disk in a file that can be printed on a printer or listed from DOS.

Smart80d and ST80-III do an excellent job of making the TRS-80 a smart terminal. ST80-III has a few more features than Smart80d. Yet, Smart80d is much more cost-effective than ST80-III. ■

✓See List of Advertisers on page 354

Modify some popular machine language utilities for duty in the Model III.

Level II Utilities for the Model III

Kenneth J. Bigelow
4117 55th Ave.
Bladensburg, MD 20710

The Model III TRS-80, like the Model I, is supplied with two versions of Basic. Level I Basic is essentially the same in either machine. However, the Level II Basic used in the Model I has been replaced by Model III Basic in the Model III. This is an enhanced version of Level II Basic, with such features as a real-time clock/calendar. However, there are some important differences which affect both Basic and machine language programs.

The Model III ROM occupies 14K, rather than the 12K used by Level II. The new ROMs are addressed from 000 hex through 37FF hex. The keyboard matrix still appears from 3800 through 3BFF, and the video display memory is still addressed from 3C00 through 3FFF. As with the Model I, this leaves 48K (from 4000 through FFFF) for RAM.

Because of the extra 2K of ROM, the disk drives and the parallel printer interfaces can no longer be addressed within the memory map. Instead, they are addressed as I/O ports. The appropriate ROM routines have been modified to account for this change, but any user program that handles these devices directly will no longer work.

Only port FF was used in the Model I, and it handled data input and output through the cassette recorder, as well as the cassette remote control. The Model III uses at least eight separate I/O ports and ignores the two least significant address lines when decoding them. The Model II therefore sees ports E0, E1, E2, and E3 as being identical. This is critical for users who add their own peripheral equipment.

The I/O ports used by the Model III are E0, E4, E8, EC, F0, F4, F8 and FF. The addressing scheme causes each of these I/O assignments to occupy four port addresses, so that no external port is allowed to occupy a port address above DF.

The third major difference is the amount of housekeeping RAM used. The Basic program is placed into RAM immediately following the housekeeping assignments which start at 4000. In Level II, the Basic program starts at address 42E9. The Model III requires more housekeeping space; Basic programs for it begin at 43E9. This means that the Model III has 256 fewer bytes (memory locations) available for program and variable storage than the Level II machine.

The reduced program space calls for revision of several Basic programs before they can be run on the 16K Model III. Specific changes are provided on a sheet enclosed with the machine. Once these changes are made, the Basic programs run with no further trouble.

Revising Machine Language Programs

Some machine language programs have problems beyond over-lengthy lines or storage requirements. Radio Shack lists three different categories of machine language (System) programs: those that will work correctly; those that should be replaced with an alternate program supplied by Radio Shack; and those that will not work and for which no replacement program is in the works. This last category includes Microchess, the Editor/Assembler, and T-Bug. It also includes some System programs written by others for the TRS-80, such as the TRS232 and Formatter programs.

I wanted to be able to use these programs on the Model III, so I set about finding out exactly what was wrong with each, in terms of the Model III machine.

My first requirement was to get printout dumps of T-Bug, the Level II ROMs, and the Model III ROMs. This would allow me to trace through specific sections of each for comparison so I could determine what changes to make. To do this, I needed to get either TRS232 or Formatter working, to drive the printer (a Model 43 teletype) from the Model III computer. This, in turn, meant that I would have to get T-Bug working so I could find the changes I needed to make to the printer driver.

I tried loading T-Bug into the Model III, just to see what would happen. To my very great surprise, only two functions did not work, and they were the cassette operations P (Punch) and L (Load). All other functions worked correctly, so I was able to use the M (Memory) function of T-Bug to look around in ROM and the housekeeping part of RAM. I was also able to use T-Bug to modify the printer driver routines.

The TRS232 program is designed to allow the TRS-80 to drive an RS232 converter (supplied with the program tape), using the cassette output cable. This allows the user to connect any RS232 printer to the computer without including the expansion interface. The same hardware can also drive a 20 milliamp current loop for printers accepting this type of interface.

The Formatter program is a more advanced driver program that uses the same hardware, but incorporates some extra bells and whistles. Both programs use the printer device control block, and modify the driver routine address in that block. This means that either program allows the computer to respond to LPRINT and LLIST commands, even though the Radio Shack printer is not connected to the system.

Using T-Bug, I found that the device control blocks were in the same place, but that the driver routine addresses had changed. Both TRS232 and Formatter use these addresses, so it was no wonder that the programs would not work. Unfortunately, the TRS232 program also has some calls to a Level II ROM routine called Port. This routine is not in the same location in the Model III. No such calls exist in Formatter, however, so I was able to correct all ROM interactions by adjusting the driver addresses called for the keyboard and video display.

One more change was required to get Formatter working properly: The Model III

"With Formatter working, I was able to get a complete dump of the Model III ROMs."

system clock is 14 percent faster than the Model I clock. Bit timing for the interface is accomplished through software, so I had to change the timing delay constants for the baud rates I wanted. The result was a perfectly operating Formatter program.

Because the Formatter program is supplied as a Basic program which POKEs the machine language program into high memory, it is very easy to modify. A few changes to some data statements will solve the problem. Then, when the Basic program is run, the resulting machine language program will be corrected.

The first change involves two calls to the video driver routine in ROM. In the Level II machine, this routine appeared at 0458 hex. In the Model III, it has been moved to 0473 hex. To make the change, locate lines 3480 and 3540 in the Basic program. In each line, the number 88 must be changed to 115.

The second change is to the keyboard scanning routine. Formatter originally contained this call to correct a keyboard bounce problem in the early Level II ROMs. The problem has long since been corrected, but the routine may still be used if desired; it resets the printed line count each time the Clear key is pressed. The same result may be obtained by typing in the command: LPRINT CHR$(3).

If you choose to use this function, change the calling address in ROM. In Level II, the address was 03FB hex; in the Model III it is 340D hex. Change the sequential numbers 251,3 to 13,52 in line 4780.

The final change is to line 5300. Each delay constant must be multiplied by a factor of 1.143. The resulting line is: 5300 DATA 703,574,514,254,167,123,58,26,9,1. I could not test all the values, of course, but 254 works perfectly for 300 baud.

With Formatter working, I was able to get a complete dump of the Model III ROMs. Then I could tackle the cassette functions of T-Bug.

Since only the cassette functions gave me trouble, I located and disassembled the cassette control routines in Level II, Model III, and T-Bug for comparison. I found that the T-Bug routines made no attempt to call any ROM routines at all. Instead, all T-Bug operations are performed independently, to avoid interactions between Basic and T-Bug. Only one program can use either the keyboard or cassette interface at a time, so why not use ROM routines?

I found that the entry points for all but one Level II routine were preserved in the Model III. The last entry point is the routine to turn on the cassette motor. In Level II, this is a separate routine, but in the Model III, it is incorporated into the routines that write or search for the header on tape. Except for this difference, T-Bug could be run on either machine, once modified.

ADDRESS	NEW BYTE
4644	12
4645	02
4547	96
4648	02
4649	00
464A	00
464B	00
464C	00
4698	C3
4699	F8
469A	01
46DE	12
46DF	02
46EB	CD
46EC	87
46ED	02
46EE	C3
46EF	FB
46FO	46
473F	C3
4740	F8
4741	01
4782	C3
4783	35
4784	02
478C	C3
478D	64
478E	02

Table. 1. Required Changes to T-Bug.

A fast check of the Model III dump showed that the address of the Level II turn-on routine was harmless. It contains an XOR A instruction (which clears the A register), followed by a RET (return from subroutine) instruction. In the Model III, this amounts to a dummy subroutine call and affects nothing of any importance. Therefore, this call can be left in the Model III version of T-Bug, and the Level II version will be identical. All I had to do was locate the points in T-Bug where the ROM calls could be inserted to replace the original T-Bug routines. This was not difficult, and only 28 bytes need be changed.

The locations of the changes and the new data to be used are shown in Table 1. For all these changes, T-Bug can modify itself because all the changes are in the L and P routines within T-Bug. The M function still works properly in either machine. No interaction occurs while the changes are being made.

The modifications to T-Bug may be made on either machine, and the modified program is then transportable to either machine. It may be used to save itself on tape (use the command P 4380 4824 43A0 name). Furthermore, it will work at either 500 baud or 1500 baud on the Model III.

Some Precautions

Even though T-Bug will now work properly on the Model III, there are a few precautions to take. These are required because T-Bug overlaps the input buffer and the start of the Basic program buffer.

The main requirement in the Model III is that T-Bug not be entered while a Basic program is present, or vice-versa. The moment one is loaded, the other will be wiped out. By the same token, a long input line, even in the absence of any Basic program, will make inroads into T-Bug. And, the moment you define a variable, T-Bug is shot.

The remaining precaution involves the Reset button. In the Model III, this is a true cold-start reset; no pointers or Basic program will be preserved. However, except for setting a few nulls to indicate no program in the program space, memory is not altered as it is sized. This means that most machine language programs will be unchanged. Unfortunately, this is not true of T-Bug; those nulls write over the command structure of the program, and the only command that will still work is R (register display). Therefore, T-Bug must be reloaded following reset in the Model III.

The changes in T-Bug are required because the Model III does not handle cassette control in the same way as the Model I. Cassette data, both input and output, is still performed through port FF. However, the motor control function has been displaced to port EC, along with the clock control and the video mode select. Bit 0 turns the clock display on or off, bit 1 turns the cassette motor on or off, and bit 2 when set selects 32 characters per line. Bits 3 and 5 are preset during initialization, but I have not been able to determine what function they perform. Memory location 4210 hex (16912 decimal) contains a copy of the last byte sent to port EC. By changing a bit in location 4210 as a direct command, either the display format or the clock display may be controlled. The Ready routine will always turn off the cassette motor.

I was unable to load Microchess into the Model III. This program seems to occupy most, if not all, of the housekeeping section of RAM, and disturbs some pointers before it can be run. As a result, any attempt to load Microchess results in a loss of all control, followed by the cassette query on the screen.

What Else Can Be Done?

From what I have learned about the Model III, it seems that any machine language program that resides above address 43E9 and calls ROM routines (from their normal entry points) to control peripheral devices will work as well on the Model III as the Level II.

Programs that attempt to directly access peripheral devices must be rewritten to accommodate the changed addressing method. If possible, they should use the built-in ROM routines, so they can be made transportable.

Programs that occupy low memory (4000 through 42E9) cannot be used at all in the Model III. Programs that occupy memory from 42E9 through 43E9 might be used, with care.

Programs that use critical software timing loops must be modified to account for the faster clock of the Model III. Any delay constants must be multiplied by a factor of 1.143 to accomplish this.

Basic programs with more than 256 bytes of free memory may be transported directly to the Model III. A Basic program that does not meet this criterion must be shortened before it can be successfully run on the Model III. When shortening a program, remember that memory is allocated dynamically during run time to handle GOSUBs, For. . . Next loops, parentheses, etc. Thus, memory size will be larger when the program halts than while it is running.■

DISK III

100% Compatible

Model III Disks

Complete Business System includes: 48K TRS-80™ Model III, Disk III™ 2 Drive System, TRSDOS and Manual.

$1882

DISK III Single drive assy[1]	**$599.00**
DISK III Two drive assy	**864.00**
DISK III Assy w/out drives	**435.00**
TRSDOS™ & Manual	**21.90**
External drives (3 & 4)	**299.00**

IMMEDIATE DELIVERY - COMPARE AND SAVE

VR Data's DISK III features:

- Completely compatible with TRSDOS
- State-of-the-art circuitry
- Fully tested, calibrated & burned-in
- Up to 4 drive configuration
- Heavy duty linear power supply
- Warranty 120 days - 100% parts and labor
- Installation with simple hand tools
- Optional 80 tk and 2 sided drives

[1]DISK III single drive assembly includes: one 40 track 5¼" double density drive, power supply, controller, mounting hardware, and applicable cables.

Peripherals

Epson MX80	$485.00
Epson MX80FT	600.00
Microline 80	425.00
Centronics 739	750.00
Starwriter 25 (P)	1395.00
RS-232 Model III	75.00
Lexicon Modem	125.00

MODEL I/III Drive w/PS & Enc **$299**

Fully Compatible
120 day warranty
Easy installation

80 tk or 2 sided	$419.00
80 tk & 2 sided	549.00

IF YOU DON'T SEE IT ADVERTISED CALL US AND ASK FOR IT.
PUBLISHED PRICES REFLECT CASH DISCOUNT.
ALL PRICES ARE SUBJECT TO CHANGE WITHOUT NOTICE.

TRS-80 and TRSDOS are trademarks of Tandy Corp.
DISK III is a trademark of VR Data Corp.
Dealership available. **To order call toll free-800 345-8102 or (215) 461-5300 in PA.**

OTHER PRODUCTS

TRS-80 Color Computer 16K	$525.00
Disk & Other Mysteries	22.50
Basic & Other Mysteries	29.95
The Custom TRS-80	29.95
Superbrain 64K	2849.00
9½ x 11 paper-case	29.95
Ribbons	QUOTE

VR Data
VR Data Corporation
777 Henderson Boulevard • Folcroft, PA 19032

master charge VISA

See List of Advertisers on page 354

An Assembly code program that keeps the computer, not the user, number crunching.

Base Conversions

James Yowell Yelvington
811 East Will Rogers Drive
Stillwater, OK 74074

No doubt there are good reasons why the TRS-80 was designed to utilize several different number formats. Whatever the reasons, this multiplicity would challenge even such a polyglot as the loquacious 'droid See Threepio. And as for mere humans, how can we expect to cope?

Not long after I bought my computer, it began to dawn on me that I was doing nearly as much computing for it as it was doing for me. This unexpected situation struck me as both paradoxical and unfortunate, but there seemed little I could do about it. The computer was inflexible about its numbers: When it wanted decimal, it had to have decimal; when it wanted split decimal, nothing else would do. The case was further complicated by various utility programs I had bought—they refused to speak anything but hexadecimal.

For awhile I endured the situation. I learned to translate numbers fairly quickly and accurately by using either direct Basic commands or a small calculator which I kept near the computer. Then I decided to write a Basic program to do the conversions.

What I really needed was a machine language program. Such a program could be stashed in high memory where it would be out of the way of Basic and other programs, and would be easily accessible. It would not interfere with Basic variables, would not disappear on system re-initialization, and would be faster than a Basic program.

I looked through software ads in various computing magazines to see if I could locate a suitable program of this kind, but I had no success. My choice was to write my own.

Number Formats

As a first step in planning the program I reviewed the various number formats used when computing with the TRS-80. The first, simplest, and most obvious is plain decimal. Numbers such as 21808, −149, and 17.54 are quite familiar and present no problems.

Another format is exponential notation, which is often used for very small or very large numbers. While scientists and engineers feel comfortable with them, most laymen find numbers like −5.12E+22, 4.659E-08, −6.253D−12, and 1.109D+26 rather peculiar at first sight. They are not really very difficult, however.

The letter indicates the amount of storage the computer allots for the number, which in turn determines the maximum number of significant digits. Single-precision numbers (E) have up to six significant digits; double-precision (D) have up to 17. The signed number following the letter tells how many places to move the decimal point to the right (+) or to the left (−) in order to arrive at the actual value of the number.

The operand of Basic's PEEK function (and the first operand of the POKE statement) is another form. For systems with 16K or less of memory, ordinary decimal integers are used. For larger systems, however, addresses higher than 32767 are represented by negative integers which are found by subtracting 65536 from the actual, or absolute, address. Thus, the address following 32767 is given as −32768 in the PEEK function, and the highest allowable address (65535) is given as −1.

Split decimals are used primarily when POKEing the starting address of a machine-language subroutine into reserved memory. They are used to set up a call from Basic through the USR(n) function. This format consists of an ordered pair of decimal numbers derived from the absolute form of the address. The address is divided by 256. The first number of the ordered pair is the remainder; the second number is the integer part of the quotient. The address 31279, for example, is given as the pair 47 and 122 (least and most significant bytes, respectively).

Computing Hex Numbers

The hexadecimal, often abbreviated as hex, format occurs most commonly as a four-digit representation of a two-byte memory address. It also may be found as a two-digit representation of the contents of a single byte in memory or in a register. The hex format is commonly used in machine language programs.

To computer hex, the decimal number is divided by the largest power of 16 that will go into it. The first hex digit is the integer part of the quotient. The second digit is derived from the remainder, which is divided by the next lower power of 16. The second hex digit is the integer part of this quotient.

The process continues through the final division by 16 to the zero power (which equals one and never leaves a remainder). Hex digits whose values are greater than nine are represented by the letters A–F, which represent 10–15, respectively. Hex numbers are often followed by the letter H to distinguish them from decimal numbers. The decimal numbers 2857 and 51208, for example, are expressed in hexadecimal format as B29H and C808H.

Binary Numbering

Binary is the most basic and elemental language which the computer knows. The other numeric formats are simply alternative ways of expressing binary information for the convenience of human operators. Binary numbers appear as strings (often rather lengthy) of ones and zeros.

No matter how hard it is to understand binary numbers, there are times when you

↙See List of Advertisers on page 354

"You can't tell two's complement. . . by appearance alone. Context reveals the format."

can't avoid using them. For example, if you want to use the cassette port for sound effects, you must control the status of individual binary digits. These digits affect bits within the byte that outputs through the port.

Binary numbers are computed from decimal by the same method that hex numbers are. Of course, division is by powers of two rather than 16.

A special type of binary is "two's complement," which is used internally by the computer to represent signed integers. Such integers are used as operands for various Basic and machine language instructions. In Basic they are also used for integer variables. Machine-language programmers often have to deal with this format, though Basic programmers generally can avoid it. (Knowledge of it does help to explain certain Basic operations. To learn more about two's-complement numbers, see William Barden's *TRS-80 Assembly-Language Programming* published by Radio Shack.) It's worth noting that you can't tell two's complement from regular binary or even hex by appearance alone. Context reveals the format. The binary number 1110 0101 could mean 229 (absolute) or −27 (two's complement).

A further complication of binary is the floating-point format used internally by the computer to represent single and double-precision Basic variables. Conversion to floating point is not simple. If you're interested in learning more about it, see Wes Thielke's article "ROM Routines," *80 Microcomputing*, Feb. 1980.

Writing the Program

This conversion program, called CONVRT, is designed to be short, fast, and convenient. It provides the most frequently needed conversions, and many that are less common.

CONVRT displays results clearly, using only a small part of the screen so that prior conversions can remain on display.

"No matter what outlandish dialect the computer speaks, I can force it to translate."

Conversions can be performed one-to-one on demand, rather than presenting multiple conversions from a single input. As far as practical, conversions are available in both directions. Direct conversions are not provided if two or three conversions can accomplish the same result.

Specific Conversions

Decimal to hex and the reverse are included, as are hex to binary and vice versa for single bytes. Since my system has over 16K of memory, CONVRT has absolute-to-PEEK and the reverse conversions. Because of the difficulty of converting decimal integers and floating point numbers to their internal storage formats, CONVRT includes these conversions, but only with hex output.

Where possible I use ROM routines to cut down on programming and space requirements. I also employ the RST 48 vector to provide a one-byte call to the frequently used video display routine at 28A7H in ROM.

For convenience, I use a Disk Basic command vector to set up access to the program from Basic through the Get command. There is single keystroke access to the major conversion routines. There are similar exits to Basic and to the system monitor. The program occupies 1160 bytes, including a 30-byte internal stack.

No matter what outlandish dialect the computer speaks, I can now force it to translate.

Setting Up the Program

To take advantage of CONVRT, you will have to assemble it from Program Listing 1 and record the object file on tape. Before assembly you may want to make certain changes. As listed, CONVRT loads into the top of a 16K Level II system, but a change in the ORG statement will let you load it wherever you like.

If you have Disk or Level III Basic, you may want to delete lines 27–30, which set up access to the program by Get. Change the operand of the End statement (line 499) from Start to ENT2. Enter the program each time by the System/entry address method.

(You may leave lines 27–30 in, however. If you always enter the program at ENT2 by System/entry address method, the Get command vector is undisturbed.)

Using the Program

To use CONVRT, first set the memory size, then load. If the ORG statement is left at 7B78H, the memory size should be 31607 and the entry address, 31608.

After using CONVRT, reenter anytime by System/entry address. If the Get access method is set up for the first entry, reentry from Basic can be achieved simply by typing "GET."

After CONVRT is entered, the main prompt identifies the program and lists the commands available in the main menu. All commands are single keystrokes: B, D, H, N, P, the Enter key, and the Clear key.

Commands and Functions

B—Byte conversion between hex and binary. A secondary prompt requests B (binary) or H (hex) to identify type of input and the direction of conversion. To convert a byte of binary to its hex equivalent, press B, then enter a string of eight binary digits, zeros and ones with no spaces before or between them. The hex value will be displayed. To convert a byte of hex to binary, press H, then enter two hex digits. The binary form, or bit pattern, will be displayed (from most significant to least significant bit, with intervening spaces for greater legibility). After any conversion, you are

2-25	Define ROM routine entry points and reserved RAM locations. Labels M1 through M9 identify arithmetic transfer, conversion, and computation routines in ROM. Labels RM1 through RM8 identify reserved RAM locations.
27-30	Set up access to CONVRT through Get.
31-44	General setup routine.
45-62	Main menu. Display main prompt, scan keyboard, jump to main routines.
63-71	Return routine. Exit to Basic or system monitor.
72-79	Secondary menu for B command. Display prompt, scan keyboard, jump to subordinate routines.
80-121	Perform byte conversion from binary to hex.
122-163	Perform byte conversion from hex to binary.
164-215	Perform decimal to hex (two-byte) conversion.
216-257	Perform hex to decimal (two-byte) conversion.
258-267	Secondary menu for N command. Display prompt, scan keyboard, jump to subordinate routines.
268-293	Display internal format of integer (in hex).
294-318	Display internal format of single-precision number (in hex).
319-342	Display internal format of double-precision number (in hex).
343-350	Secondary menu for P command. Display prompt, scan keyboard, jump to subordinate routines.
351-365	Perform absolute to PEEK address conversion.
366-399	Perform PEEK to absolute address conversion.
400-453	Various utility subroutines called by main or secondary routines.
454	Reserve stack space.
455-495	Define messages and parts of messages for display.
496-498	Reserve memory for storage.

Table 1

"The two most important conversions... are from decimal to hex and vice versa... Two-byte integers are easiest."

returned to the main prompt and menu.

(Note: If you make an error while entering a number anywhere in the program, you can backspace to correct it.)

D—Decimal to hex. Enter a decimal integer from 0 to 65535; the hex equivalent is displayed (most significant byte first) as four hex digits.

H—Hex to decimal. Enter exactly four hex digits (use leading zeros if necessary), and the absolute (positive) decimal equivalent is displayed.

N—Number (decimal) to internal format. A secondary prompt requests I (integer), S (single-precision), or D (double-precision) to indicate type of input number. To display the internal format of an integer, press I, then enter any decimal integer between −32768 and 32767. Two bytes (least significant byte first) are displayed as four hex digits.

To display the internal, floating point format of a single-precision decimal number, press S, then enter the number. (It can be exponential notation.) Four bytes, from least to most significant, will be displayed as eight hex digits (grouped into fours for legibility). To display the internal, floating point format of a double-precision number, press D, then enter the number. (It may be exponential notation.) Eight bytes are displayed as 16 hex digits. They are grouped in fours from least to most significant.

P—PEEK and absolute address. A secondary prompt requests A (absolute) or P (PEEK) address format to be input. To convert an absolute address to PEEK format, press A, then enter any address from 0 to 65535. The equivalent PEEK format of the address appears on the screen. To convert a PEEK address to its absolute equivalent, press P, then enter an address value from −32768 to 32767.

ENTER—Return to Basic. To return to Basic from the main menu, press Enter. This returns you to Basic at 06CCH. This command makes it easy to jump back and forth between CONVRT for conversions and Basic for calculations.

CLEAR—Return to system monitor. To return to the system monitor from the main menu, press Clear. You return to the system monitor at 02B2H, and can load a machine language program or jump with the /entry address command to any location.

These commands and functions can be used to make two's-complement and split-decimal conversions.

Two's Complement Conversions

The most important conversions involving the two's-complement format are from decimal to hex and vice versa, with two-byte and one-byte integers. Two-byte integers are easiest.

Signed decimal integers from −32768 to 32767 can be converted to two's-complement-hex simply by using the D command. Press D, then enter the signed decimal integer. The correct two's-complement-hex format appears with the most significant byte first.

Two-byte hex integers in two's-complement format are converted to their signed decimal equivalents by a two-step process. First press H, then enter the four-digit hex integer. The second step is to press P, then A, and then enter the result from the first step. The signed-decimal integer is shown with the PEEK label.

Signed-decimal integers from −128 to 127 may be converted to one-byte two's-complement hex. Press D, then enter the integer. Ignore the first two hex digits shown. The last two are the result.

One-byte hex integers in two's-complement format are converted to signed decimal by two different methods. If the first hex digit is eight or greater, press H and enter two Fs followed by the two-digit hex integer. If the first hex digit is seven or less, press H and enter two zeros followed by the two-digit hex integer. The signed decimal equivalent results.

Split Decimal Conversions

Split decimal conversions can be done with CONVRT by using multiple step operations. Some may be easier to do by using direct arithmetic commands in Basic, however. The principal conversions are described below.

To convert hex to split decimal press H, and enter two zeros followed by the last two hex digits. This yields the first number of the split decimal pair, the least significant byte. Once again, press H, and enter two zeros. This time follow with the first two hex digits. This yields the second number, the most significant byte.

To convert split decimal to hex format press D and enter the second number of the split-decimal pair (most significant byte). The last two hex digits displayed are the first two of the hex result. Press D again. Enter the first number of the split-decimal

Program Listing 1

```
                   ;CONVRT 3.2, J. YELVINGTON, JUNE 1980
0049                INCH     EQU     49H
02B2                SYS      EQU     2B2H
032A                PRCH     EQU     32AH
06CC                BAS      EQU     6CCH
0716                M1       EQU     716H
0994                M2       EQU     994H
09A4                M3       EQU     9A4H
0A7F                M4       EQU     0A7FH
0A9A                M5       EQU     0A9AH
0AB1                M6       EQU     0AB1H
0E65                M7       EQU     0E65H
0E6C                M8       EQU     0E6CH
0FBD                M9       EQU     0FBDH
1BB3                INST     EQU     1BB3H
28A7                PRST     EQU     28A7H
400F                RM1      EQU     400FH
4010                RM2      EQU     4010H
40A7                RM3      EQU     40A7H
411D                RM4      EQU     411DH
4121                RM5      EQU     4121H
4123                RM6      EQU     4123H
417F                RM7      EQU     417FH
4180                RM8      EQU     4180H
41E8                INBUF    EQU     41E8H
7B78                         ORG     7B78H
7B78 3EC3           START    LD      A,0C3H
7B7A 327F41                  LD      (RM7),A
7B7D 21837B                  LD      HL,ENT2
7B80 228041                  LD      (RM8),HL
7B83 ED73FD7F       ENT2     LD      (OSP),SP
7B87 FD2A1040                LD      IY,(RM2)
7B8B 3A0F40                  LD      A,(RM1)
7B8E 32FF7F                  LD      (RV),A
7B91 3EC3                    LD      A,0C3H
7B93 320F40                  LD      (RM1),A
7B96 21A728                  LD      HL,PRST
7B99 221040                  LD      (RM2),HL
7B9C 213C7F                  LD      HL,TITLE
7B9F F9                      LD      SP,HL
7BA0 21E841                  LD      HL,INBUF
7BA3 22A740                  LD      (RM3),HL
7BA6 3E1B                    LD      A,27
7BA8 CD2A03                  CALL    PRCH
7BAB 213C7F         MENU     LD      HL,TITLE
7BAE F7                      RST     48
7BAF CD4900         SCN1     CALL    INCH
7BB2 FE0D                    CP      13
7BB4 CAD67B                  JP      Z,RTRN
7BB7 FE1F                    CP      31
7BB9 CAD67B                  JP      Z,RTRN
7BBC FE42                    CP      'B'
```

Program continues

Program continued

```
7BBE 282E     00053       JR    Z,BITR
7BC0 FE48     00054       CP    'H'
7BC2 CA217D   00055       JP    Z,HEXR
7BC5 FE44     00056       CP    'D'
7BC7 CAAD7C   00057       JP    Z,DECR
7BCA FE4E     00058       CP    'N'
7BCC CA7B7D   00059       JP    Z,NUMR
7BCF FE50     00060       CP    'P'
7BD1 CA367E   00061       JP    Z,PKR
7BD4 18D9     00062       JR    SCN1
7BD6 47       00063 RTRN  LD    B,A
7BD7 3AFF7F   00064       LD    A,(RV)
7BDA 320F40   00065       LD    (RM1),A
7BDD FD221040 00066       LD    (RM2),IY
7BE1 ED7BFD7F 00067       LD    SP,(OSP)
7BE5 78       00068       LD    A,B
7BE6 FE1F     00069       CP    31
7BE8 CAB202   00070       JP    Z,SYS
7BEB C3CC06   00071       JP    BAS
7BEE 21907F   00072 BITR  LD    HL,BTQ
7BF1 F7       00073       RST   48
7BF2 CD4900   00074 SCN2  CALL  INCH
7BF5 FE42     00075       CP    'B'
7BF7 2806     00076       JR    Z,BBR
7BF9 FE48     00077       CP    'H'
7BFB 2855     00078       JR    Z,HBR
7BFD 18F3     00079       JR    SCN2
7BFF 21897F   00080 BBR   LD    HL,CT2
7C02 F7       00081       RST   48
7C03 21697F   00082       LD    HL,BN
7C06 F7       00083       RST   48
7C07 21637F   00084       LD    HL,BYT
7C0A F7       00085       RST   48
7C0B 21E87F   00086       LD    HL,VAL
7C0E F7       00087       RST   48
7C0F CDB31B   00088       CALL  INST
7C12 E5       00089       PUSH  HL
7C13 21887F   00090       LD    HL,CTL
7C16 F7       00091       RST   48
7C17 21697F   00092       LD    HL,BN
7C1A F7       00093       RST   48
7C1B 21637F   00094       LD    HL,BYT
7C1E F7       00095       RST   48
7C1F 21E841   00096       LD    HL,INBUF
7C22 F7       00097       RST   48
7C23 218C7F   00098       LD    HL,EQ
7C26 F7       00099       RST   48
7C27 E1       00100       POP   HL
7C28 AF       00101       XOR   A
7C29 0608     00102       LD    B,8
7C2B 23       00103 BL0   INC   HL
7C2C CB27     00104       SLA   A
7C2E CB46     00105       BIT   0,(HL)
7C30 2802     00106       JR    Z,BL1
7C32 CBC7     00107       SET   0,A
7C34 10F5     00108 BL1   DJNZ  BL0
7C36 67       00109       LD    H,A
7C37 CB3F     00110       SRL   A
7C39 CB3F     00111       SRL   A
7C3B CB3F     00112       SRL   A
7C3D CB3F     00113       SRL   A
7C3F CD017F   00114       CALL  CVA
7C42 7C       00115       LD    A,H
7C43 E60F     00116       AND   0FH
7C45 CD017F   00117       CALL  CVA
7C48 216F7F   00118       LD    HL,HX
7C4B F7       00119       RST   48
7C4C CD187F   00120       CALL  CR
7C4F C3AB7B   00121       JP    MENU
7C52 21897F   00122 HBR   LD    HL,CT2
7C55 F7       00123       RST   48
7C56 216F7F   00124       LD    HL,HX
7C59 F7       00125       RST   48
7C5A 21637F   00126       LD    HL,BYT
7C5D F7       00127       RST   48
7C5E 21E87F   00128       LD    HL,VAL
7C61 F7       00129       RST   48
7C62 CDB31B   00130       CALL  INST
7C65 E5       00131       PUSH  HL
7C66 21887F   00132       LD    HL,CTL
7C69 F7       00133       RST   48
7C6A 216F7F   00134       LD    HL,HX
7C6D F7       00135       RST   48
7C6E 21637F   00136       LD    HL,BYT
7C71 F7       00137       RST   48
7C72 21E841   00138       LD    HL,INBUF
7C75 F7       00139       RST   48
7C76 218C7F   00140       LD    HL,EQ
7C79 F7       00141       RST   48
7C7A E1       00142       POP   HL
7C7B 0602     00143       LD    B,2
7C7D CDEA7E   00144       CALL  BFHX
7C80 DD21E841 00145       LD    IX,INBUF
7C84 CDF27E   00146       CALL  HXBI
7C87 4F       00147       LD    C,A
7C88 0608     00148       LD    B,8
7C8A CB79     00149 HB1   BIT   7,C
7C8C 2807     00150       JR    Z,HB2
7C8E 3E31     00151       LD    A,49
7C90 CD2A03   00152       CALL  PRCH
7C93 1805     00153       JR    HB3
7C95 3E30     00154 HB2   LD    A,48
7C97 CD2A03   00155       CALL  PRCH
7C9A CB21     00156 HB3   SLA   C
7C9C 3E20     00157       LD    A,32
7C9E CD2A03   00158       CALL  PRCH
7CA1 10E7     00159       DJNZ  HB1
7CA3 21697F   00160       LD    HL,BN
7CA6 F7       00161       RST   48
7CA7 CD187F   00162       CALL  CR
7CAA C3AB7B   00163       JP    MENU
```

Program continues

Program continued

```
7CAD 21897F   DECR  LD    HL,CT2
7CB0 F7             RST   48
7CB1 21757F         LD    HL,DC
7CB4 F7             RST   48
7CB5 21E87F   DPR   LD    HL,VAL
7CB8 F7             RST   48
7CB9 CDB31B         CALL  INST
7CBC E5             PUSH  HL
7CBD 21887F         LD    HL,CTL
7CC0 F7             RST   48
7CC1 21757F         LD    HL,DC
7CC4 3AFC7F         LD    A,(FLG)
7CC7 FE50           CP    'P'
7CC9 2004           JR    NZ,DP2
7CCB 110600         LD    DE,6
7CCE 19             ADD   HL,DE
7CCF F7       DP2   RST   48
7CD0 21E841         LD    HL,INBUF
7CD3 F7             RST   48
7CD4 218C7F         LD    HL,EQ
7CD7 F7             RST   48
7CD8 E1             POP   HL
7CD9 D7             RST   16
7CDA CD6C0E         CALL  M8
7CDD CDB10A         CALL  M6
7CE0 CDA409         CALL  M3
7CE3 068F           LD    B,8FH
7CE5 0EFF           LD    C,0FFH
7CE7 16FE           LD    D,0FEH
7CE9 1E00           LD    E,0
7CEB CD1607         CALL  M1
7CEE CD9409         CALL  M2
7CF1 C1             POP   BC
7CF2 D1             POP   DE
7CF3 ED532141       LD    (RM5),DE
7CF7 ED432341       LD    (RM6),BC
7CFB FE01           CP    1
7CFD 200A           JR    NZ,DP3
7CFF 0691           LD    B,91H
7D01 0E80           LD    C,80H
7D03 1600           LD    D,0
7D05 5A             LD    E,D
7D06 CD1607         CALL  M1
7D09 CD7F0A   DP3   CALL  M4
7D0C 3AFC7F         LD    A,(FLG)
7D0F FE50           CP    'P'
7D11 CA577E         JP    Z,PAR2
7D14 CDC57E         CALL  HLHX
7D17 216F7F         LD    HL,HX
7D1A F7             RST   48
7D1B CD187F         CALL  CR
7D1E C3AB7B         JP    MENU
7D21 21897F   HEXR  LD    HL,CT2
7D24 F7             RST   48
7D25 216F7F         LD    HL,HX
7D28 F7             RST   48
7D29 21E87F         LD    HL,VAL
7D2C F7             RST   48
7D2D CDB31B         CALL  INST
7D30 E5             PUSH  HL
7D31 21887F         LD    HL,CTL
7D34 F7             RST   48
7D35 216F7F         LD    HL,HX
7D38 F7             RST   48
7D39 21E841         LD    HL,INBUF
7D3C F7             RST   48
7D3D 218C7F         LD    HL,EQ
7D40 F7             RST   48
7D41 0604           LD    B,4
7D43 E1             POP   HL
7D44 CDEA7E         CALL  BFHX
7D47 DD21E841       LD    IX,INBUF
7D4B CDF27E         CALL  HXBI
7D4E 67             LD    H,A
7D4F DD23           INC   IX
7D51 DD23           INC   IX
7D53 CDF27E         CALL  HXBI
7D56 6F             LD    L,A
7D57 CD9A0A         CALL  M5
7D5A CD9409         CALL  M2
7D5D FEFF           CP    0FFH
7D5F 200C           JR    NZ,HR1
7D61 CDB10A         CALL  M6
7D64 0691           LD    B,91H
7D66 0E00           LD    C,0
7D68 51             LD    D,C
7D69 59             LD    E,C
7D6A CD1607         CALL  M1
7D6D CDBD0F   HR1   CALL  M9
7D70 F7             RST   48
7D71 21757F         LD    HL,DC
7D74 F7             RST   48
7D75 CD187F         CALL  CR
7D78 C3AB7B         JP    MENU
7D7B 21A57F   NUMR  LD    HL,NTQ
7D7E F7             RST   48
7D7F CD4900   SCN3  CALL  INCH
7D82 FE49           CP    'I'
7D84 280A           JR    Z,IR
7D86 FE53           CP    'S'
7D88 2841           JR    Z,SR
7D8A FE44           CP    'D'
7D8C 2874           JR    Z,DR
7D8E 18EF           JR    SCN3
7D90 21897F   IR    LD    HL,CT2
7D93 F7             RST   48
7D94 21D67F         LD    HL,INT
7D97 F7             RST   48
7D98 21E87F         LD    HL,VAL
7D9B F7             RST   48
7D9C CDB31B         CALL  INST
7D9F E5             PUSH  HL
7DA0 21887F         LD    HL,CTL
```

Program continues

✓ *See List of Advertisers on page 354*

Program continued

```
7DA3 F7                 RST     48
7DA4 21D67F             LD      HL,INT
7DA7 F7                 RST     48
7DA8 21E841             LD      HL,INBUF
7DAB F7                 RST     48
7DAC 21EE7F             LD      HL,EQN
7DAF F7                 RST     48
7DB0 E1                 POP     HL
7DB1 D7                 RST     16
7DB2 CD6C0E             CALL    M8
7DB5 CD7F0A             CALL    M4
7DB8 DD212141           LD      IX,RM5
7DBC DD6600             LD      H,(IX+0)
7DBF DD6E01             LD      L,(IX+1)
7DC2 CDC57E             CALL    HLHX
7DC5 CD187F             CALL    CR
7DC8 C3AB7B             JP      MENU
7DCB 21897F       SR    LD      HL,CT2
7DCE F7                 RST     48
7DCF 21DC7F             LD      HL,SNG
7DD2 F7                 RST     48
7DD3 21E87F             LD      HL,VAL
7DD6 F7                 RST     48
7DD7 CDB31B             CALL    INST
7DDA E5                 PUSH    HL
7DDB 21887F             LD      HL,CTL
7DDE F7                 RST     48
7DDF 21DC7F             LD      HL,SNG
7DE2 F7                 RST     48
7DE3 21E841             LD      HL,INBUF
7DE6 F7                 RST     48
7DE7 21EE7F             LD      HL,EQN
7DEA F7                 RST     48
7DEB E1                 POP     HL
7DEC D7                 RST     16
7DED CD6C0E             CALL    M8
7DF0 CDB10A             CALL    M6
7DF3 DD212141           LD      IX,RM5
7DF7 0602               LD      B,2
7DF9 CDB07E             CALL    BFNP
7DFC CD187F             CALL    CR
7DFF C3AB7B             JP      MENU
7E02 21897F       DR    LD      HL,CT2
7E05 F7                 RST     48
7E06 21E27F             LD      HL,DBL
7E09 F7                 RST     48
7E0A 21E87F             LD      HL,VAL
7E0D F7                 RST     48
7E0E CDB31B             CALL    INST
7E11 E5                 PUSH    HL
7E12 21887F             LD      HL,CTL
7E15 F7                 RST     48
7E16 21E27F             LD      HL,DBL
7E19 F7                 RST     48
7E1A 21E841             LD      HL,INBUF
7E1D F7                 RST     48
7E1E 21EE7F             LD      HL,EQN
7E21 F7                 RST     48
7E22 E1                 POP     HL
7E23 D7                 RST     16
7E24 CD650E             CALL    M7
7E27 DD211D41           LD      IX,RM4
7E2B 0604               LD      B,4
7E2D CDB07E             CALL    BFNP
7E30 CD187F             CALL    CR
7E33 C3AB7B             JP      MENU
7E36 21BE7F       PKR   LD      HL,ATQ
7E39 F7                 RST     48
7E3A CD4900       SCN4  CALL    INCH
7E3D FE41               CP      'A'
7E3F 2806               JR      Z,PAR
7E41 FE50               CP      'P'
7E43 2825               JR      Z,PPR
7E45 18F3               JR      SCN4
7E47 21897F       PAR   LD      HL,CT2
7E4A F7                 RST     48
7E4B 217B7F             LD      HL,AB
7E4E F7                 RST     48
7E4F 3E50               LD      A,'P'
7E51 32FC7F             LD      (FLG),A
7E54 C3B57C             JP      DPR
7E57 3E00         PAR2  LD      A,0
7E59 32FC7F             LD      (FLG),A
7E5C CDBD0F             CALL    M9
7E5F F7                 RST     48
7E60 21817F             LD      HL,PK
7E63 F7                 RST     48
7E64 CD187F             CALL    CR
7E67 C3AB7B             JP      MENU
7E6A 21897F       PPR   LD      HL,CT2
7E6D F7                 RST     48
7E6E 21817F             LD      HL,PK
7E71 F7                 RST     48
7E72 21E87F             LD      HL,VAL
7E75 F7                 RST     48
7E76 CDB31B             CALL    INST
7E79 E5                 PUSH    HL
7E7A 21887F             LD      HL,CTL
7E7D F7                 RST     48
7E7E 21817F             LD      HL,PK
7E81 F7                 RST     48
7E82 21E841             LD      HL,INBUF
7E85 F7                 RST     48
7E86 218C7F             LD      HL,EQ
7E89 F7                 RST     48
7E8A E1                 POP     HL
7E8B D7                 RST     16
7E8C CD6C0E             CALL    M8
7E8F CDB10A             CALL    M6
7E92 CD9409             CALL    M2
7E95 FEFF               CP      0FFH
```

Program continues

PROGRAMMING TOOLS FOR YOUR TRS-80™ MODEL I AND MODEL III

INSIDE LEVEL II

The Programmers Guide to the TRS-80 ROMS

INSIDE LEVEL II is a comprehensive reference guide to the Level II ROMs which allows the machine language or Basic programmer to easily utilize the sophisticated routines they contain. Concisely explains set-ups, calling sequences, and variable passage for number conversion, arithmetic operations, and mathematical functions, as well as keyboard, tape, and video routines. Part II presents an entirely new composite program structure which loads under the SYSTEM command and executes in both Basic and machine code with the speed and efficiency of a compiler. In addition, the 18 chapters include a large body of other information useful to the programmer including tape formats, RAM useage, relocation of Basic programs, USR call expansion, creating SYSTEM tapes of your own programs, interfacing of Basic variables directly with machine code, a method of greatly increasing the speed at which data elements are stored on tape, and special precautions for disk systems. INSIDE LEVEL II is a clearly organized reference manual. It is fully typeset and packed with nothing but useful information. It does not contain questions and answers, ROM dumps or cartoons.
Includes updates for Model III. INSIDE LEVEL II.....$15.95

SINGLE STEP THROUGH RAM OR ROM

STEP80 allows you to step through any Basic or machine language program one instruction at a time, and see the address, hexadecimal value, Zilog mnemonic, register contents, and step count for each instruction. The top 14 lines of the video screen are left unaltered so that the target program may perform its display functions unobstructed. STEP80 will follow program flow right into the ROMs, and is an invaluable aid in learning how the ROM routines function. Commands include step (trace), disassemble, run in step mode at variable step rate, display or alter memory or CPU registers, jump to memory location, execute a CALL, set breakpoints in RAM or ROM, write SYSTEM tapes, and relocate to any page in RAM. The display may also be routed to your line printer through the device control block so custom print drivers are automatically supported.
Specify Model I or Model III. STEP80.....$16.95

TELECOMMUNICATIONS PROGRAM

This machine language program allows reliable high speed file transfers between two disk-based computers over modems or direct wire. It is menu driven and extremely simple to use. Functions include real-time terminal mode, save RAM buffer on disk, transmit disk file, receive binary files, examine and modify UART parameters, program 8 custom log-on messages, automatic 16-bit checksum verification of accurate transmission and reception, and many more user conveniences. Supports line printers and lowercase characters. With this program you will no longer need to convert machine language programs to ASCII for transmission, and you will know immediately if the transmission was accurate.
Specify Model I or Model III. TELCOM.....$39.95

PROGRAM INDEX FOR DISK BASIC

Assemble an alphabetized index of your entire program library from disk directories. Program names and free space are read automatically (need not be typed in) and may be alphabetized with a fast Shell Metzner sort by disk or program. The list may also be searched for any disk, program, or extension, disks or programs added or deleted and the whole list or any part sent to the printer. Finally, the list itself may be stored on disk for future access and update. The best thing since sliced bread (January issue of '80 Microcomputing). Works with TRSDOS, NEWDOS, and NEWDOS 80. One drive and 32K required.
Model I only. Model III version soon. INDEX.....$19.95

4 SPEED OPTIONS FOR YOUR TRS-80

The SK-2 clock modification allows CPU speeds to be switched between normal, an increase of 50%, or a 50% reduction, selectable at any time without interrupting execution or crashing the program. Instructions are also given for a 100% increase to 3.54 MHz. The SK-2 may be configured by the user to change speed with a toggle switch or on software command. It will automatically return to normal speed any time a disk is active, requires no change to the operating system, and has provisions for adding an LED to indicate when the computer is not at normal speed. It mounts inside the keyboard unit with only 4 necessary connections for the switch option (switch not included), and is easily removed if the computer ever needs service. The SK-2 comes fully assembled with socketed IC's and illustrated instructions.
Model I only. SK-2.....$24.95

INSTANT ASSEMBLER

The INSTANT ASSEMBLER is a new powerful tape-based assembler and debugger for the TRS-80. Now you can assemble directly to memory and immediately debug your program with the built in single stepping debugger. Quickly switch from assembler to debugger and back again without losing the source code. This feature makes INSTANT ASSEMBLER an excellent learning tool for assembly language programming.
INSTANT ASSEMBLER is absolutely unique among tape based assemblers in that it produces relocatable code modules that can be linked with the separate LINKING LOADER, which is supplied in two versions for loading programs into either high or low RAM. This lets you build long programs with small modules. INSTANT ASSEMBLER also features immediate detection of errors as the source code is entered, a compactly coded source format that uses 1/3 as much memory as standard source, and many operational features including single stroke entry of DEFB and DEFW, pinpoint control of listings, alphabetic listing of symbol table, separate commands for listing error lines or the symbol table, block move function, and verification of source tapes.
INSTANT ASSEMBLER's debugger provides single stepping with full register displays, decimal or hex entry of addresses, forward or backward memory displays, disassembly of object code in memory, memory display in ASCII format, and hex to-decimal or decimal-to-hex conversion. The single-stepper will step one instruction at a time or at a fast rate to any defined address.
INSTANT ASSEMBLER occupies less than 8400 bytes of memory. In a 16K machine this will leave you enough memory to write assembly language programs of around 2000 bytes. This and its module-linking feature make INSTANT ASSEMBLER ideal for users with only 16K machines. The instruction manual may be purchased separately for $5 which will apply towards the purchase of the INSTANT ASSEMBLER.
Specify Model I or Model III. INTASM.....$29.95

RAM SPOOLER AND PRINT FORMATTER

This program is a full feature print formatting package featuring user definable line and page length (with line feeds inserted between words or after punctuation), screen dump, printer pause control, and baud rate selection. In addition, printing is done from a 4K expandable buffer area so that the LPRINT or LLIST command returns control to the user while printing is being done. Ideal for Selectric or other slow printers. Allows printing and processing to run concurrently. Output may be directed to either the parallel port, serial port, or the video screen.
Specify Model I or Model III. SPOOLER.....$16.95

MACHINE CODE FAST FOURIER TRANSFORM

This complete package includes 3 versions of the machine language FFTASM routine assembled for 16, 32 and 48K machines, a short sample Basic program to access them, a 10K Basic program which includes sophisticated interactive graphing and data manipulation, and a manual of instructions and examples. The machine language subroutines use variables defined by a supporting Basic program to make data entry and retrieval extremely fast and easy for custom implementation. They perform 20 to 40 times faster than their Basic equivalent (256 points in 12.5 seconds), and require less than 1550 bytes of memory. The FFT is useful in analyzing stock market and comodity trends as well as for scientific information.
Specify Model I or Model III. FFTASM.....$49.95

DUPLICATE SYSTEM TAPES WITH CLONE

Make duplicate copies of any tape written for Level II. They may be SYSTEM tapes or data lists. The file name, load address, entry point, and every byte (in ASCII format) are displayed on the video screen. Model III version allows changing tape speed.
Specify Model I or Model III. CLONE.....$16.95

RAMTEST FOR LEVEL II

This machine language program is a very thorough test for several types of RAM errors. A complete test of each individual bit in a 48K machine takes just 14 seconds. Includes a separate test for power line glitches.
Model I only. RAMTEST.....$9.95

EDIT BASIC PROGRAMS WITH ELECTRIC PENCIL

Load Basic programs or any other ASCII data file into the disk version of Electric Pencil for editing. One command from DOS quickly modifies existing files to Pencil format. One disk and 32K required.
Model I only. PENPATCH.....$9.95

MUMFORD MICRO SYSTEMS ✓144

ORDERING: Complete satisfaction is guaranteed or a full refund will be made. All programs are shipped on cassette unless $5 is included for a 35 track formatted (no system) disk. Include $1.50 for postage and handling. California residents add 6% sales tax. Visa, Mastercharge and COD orders accepted. SPECIFY MODEL I OR MODEL III. Dealer inquiries invited.

Box 400-E Summerland, California 93067 (805) 969-4557

✓See List of Advertisers on page 354

pair (least significant byte). The last two hex digits displayed are the last two of the hex result.

Using CONVRT to convert an absolute-decimal number to split decimal, press D, then enter the number. Next, convert the hex result from step one to split-decimal format by using the method described above.

To do the conversion in Basic, divide the absolute decimal number by 256, taking the integer part of the result as the second number of the split decimal pair (MSB). Then multiply the decimal part of the same result by 256 to get the first number of the pair (LSB). (If this number has a decimal part, then round it off to the nearest integer.)

It takes three steps to convert split decimal to absolute-decimal format using CONVRT. Convert the split decimal to hex format by the method described above. Next, press H, and enter the hex result.

To do the conversion in Basic, multiply the second number of the split decimal pair (MSB) by 256, then add the first number (LSB) to the result.

General Remarks

Keep the following points in mind when using CONVRT. Error detection has been *minimized* in order to keep the program short. For example, if you enter fewer than four hex digits after pressing H, the conversion is incorrect, but no error message appears. Entry of hex and binary is more prone to error than entry of decimal values. Leading or embedded spaces in hex or binary entries lead to errors without warning. On the other hand, entry of decimal numbers outside allowable ranges will cause overflow errors and immediate return to Basic.

All hex and binary results, except those produced by the N routine (number to internal format), are displayed in the normal order of most significant to least significant. Entries should be in this same order.

The returns to Basic and to the system monitor are not available within conversion routines. This means you cannot abandon a conversion once you select it. Entry of a dummy value lets the conversion proceed, then you can use the returns.

Program Structure

CONVRT is structured around a main menu, which lists the main commands, scans the keyboard for them, and, upon detecting one, jumps to the appropriate program segment. Each of the five general types of conversion is written as a major program segment, or main routine, which terminates by a return to the main menu. Three of the main routines have secondary menus which direct execution to subordinate routines within them. In addition, there is an access setup routine, a general setup routine, a return routine, and several utility subroutines called by one or more of the main routines. A running commentary explains lines of the program in Table 1.■

Program continued

```
7E97 2009              JR      NZ,PPR2
7E99 0691              LD      B,91H
7E9B 0E00              LD      C,0
7E9D 51                LD      D,C
7E9E 59                LD      E,C
7E9F CD1607            CALL    M1
7EA2 CDBD0F     PPR2   CALL    M9
7EA5 F7                RST     48
7EA6 217B7F            LD      HL,AB
7EA9 F7                RST     48
7EAA CD187F            CALL    CR
7EAD C3AB7B            JP      MENU
7EB0 DD6600     BFNP   LD      H,(IX+0)
7EB3 DD6E01            LD      L,(IX+1)
7EB6 CDC57E            CALL    HLHX
7EB9 3E20              LD      A,32
7EBB CD2A03            CALL    PRCH
7EBE DD23              INC     IX
7EC0 DD23              INC     IX
7EC2 10EC              DJNZ    BFNP
7EC4 C9                RET
7EC5 7C         HLHX   LD      A,H
7EC6 CB3F              SRL     A
7EC8 CB3F              SRL     A
7ECA CB3F              SRL     A
7ECC CB3F              SRL     A
7ECE CD017F            CALL    CVA
7ED1 7C                LD      A,H
7ED2 E60F              AND     0FH
7ED4 CD017F            CALL    CVA
7ED7 7D                LD      A,L
7ED8 CB3F              SRL     A
7EDA CB3F              SRL     A
7EDC CB3F              SRL     A
7EDE CB3F              SRL     A
7EE0 CD017F            CALL    CVA
7EE3 7D                LD      A,L
7EE4 E60F              AND     0FH
7EE6 CD017F            CALL    CVA
7EE9 C9                RET
7EEA D7         BFHX   RST     16
7EEB CD0E7F            CALL    CVH
7EEE 77                LD      (HL),A
7EEF 10F9              DJNZ    BFHX
7EF1 C9                RET
7EF2 DD7E00     HXBI   LD      A,(IX+0)
7EF5 CB27              SLA     A
7EF7 CB27              SLA     A
7EF9 CB27              SLA     A
7EFB CB27              SLA     A
7EFD DD8601            ADD     A,(IX+1)
7F00 C9                RET
7F01 C630       CVA    ADD     A,48
7F03 FE3A              CP      58
7F05 FA0A7F            JP      M,CA1
7F08 C607              ADD     A,7
7F0A CD2A03     CA1    CALL    PRCH
7F0D C9                RET
7F0E D630       CVH    SUB     48
7F10 FE0A              CP      10
7F12 FA177F            JP      M,CH1
7F15 D607              SUB     7
7F17 C9         CH1    RET
7F18 3E0D       CR     LD      A,13
7F1A CD2A03            CALL    PRCH
7F1D C9                RET
001E            ENDSTK DEFS    30
7F3C 1D1E       TITLE  DEFW    1E1DH
7F3E 43                DEFM    'CONVRT:  PRESS B/D/H/N/P/ENTER/CLEAR'
7F62 00                DEFB    0
7F63 42         BYT    DEFM    'BYTE '
7F68 00                DEFB    0
7F69 20         BN     DEFM    ' BIN '
7F6E 00                DEFB    0
7F6F 20         HX     DEFM    ' HEX '
7F74 00                DEFB    0
7F75 20         DC     DEFM    ' DEC '
7F7A 00                DEFB    0
7F7B 20         AB     DEFM    ' ABS '
7F80 00                DEFB    0
7F81 20         PK     DEFM    ' PEEK '
7F87 00                DEFB    0
7F88 1B         CTL    DEFB    27
7F89 1D1E       CT2    DEFW    1E1DH
7F8B 00                DEFB    0
7F8C 20         EQ     DEFM    ' = '
7F8F 00                DEFB    0
7F90 1D1E       BTQ    DEFW    1E1DH
7F92 20                DEFM    ' BYTE TYPE?  (B/H)'
7FA4 00                DEFB    0
7FA5 1D1E       NTQ    DEFW    1E1DH
7FA7 20                DEFM    ' NUMBER TYPE?  (I/S/D)'
7FBD 00                DEFB    0
7FBE 1D1E       ATQ    DEFW    1E1DH
7FC0 20                DEFM    ' ADDRESS TYPE?  (A/P)'
7FD5 00                DEFB    0
7FD6 20         INT    DEFM    ' INT '
7FDB 00                DEFB    0
7FDC 20         SNG    DEFM    ' SNG '
7FE1 00                DEFB    0
7FE2 20         DBL    DEFM    ' DBL '
7FE7 00                DEFB    0
7FE8 56         VAL    DEFM    'VALUE'
7FED 00                DEFB    0
7FEE 20         EQN    DEFM    ' = (LSB'
7FF5 5E                DEFB    94
7FF6 4D                DEFM    'MSB) '
7FFB 00                DEFB    0
7FFC 00         FLG    DEFB    0
7FFD 0000       OSP    DEFW    0
7FFF 00         RV     DEFB    0
7B78                   END     START
```

H & E COMPUTRONICS INC.

•••EVERYTHING FOR YOUR TRS-80™•••

* TRS-80™ is a trademark of the Radio Shack Division of Tandy Corporation

Currently Available

MOD-II PROGRAMS

- ★ *All orders processed within 24-Hours*
- ★ *30-Day money back guarantee on all TRSDOS Software*
- ★ *Add $3.00 for shipping in UPS Areas*
- ★ *Add $4.00 for C.O.D. or NON-UPS Areas*
- ★ *Add $5.00 outside U.S.A., Canada & Mexico*
- ★ *We will match any bonafide advertised price in any of the Major Computer Magazines*

ALL SOFTWARE LISTED HERE WORKS WITH TRSDOS*

(1) ELECTRIC PENCIL (Michael Shrayer Software). Complete word processor with extensive editing and printer formatting features....$325 (STANDARD TRSDOS VERSION)...$350 (DIABLO, NEC OR QUME TRSDOS VERSION).

(2) GENERAL LEDGER, ACCOUNTS RECEIVABLE, ACCOUNTS PAYABLE, INVENTORY CONTROL, INVOICING AND PAYROLL (Small Business Systems Group)...an extensive business system for the serious user...can be used one module at a time or as a co-ordinated system...$225...per module .$1299 for the complete system.

(3) GENERAL LEDGER, ACCOUNTS RECEIVABLE, ACCOUNTS PAYABLE, INVENTORY CONTROL AND PAYROLL (Compumax)...a complete user oriented business system...can be used one module at a time or as a coordinated system...$140 per module $995 for the complete system

(4) MOD-II UTILITY PACKAGE (Racet Computes) adds important utilities to TRSDOS...copy files selectively...faster and more accurate file copying. repair bad directories...displays sorted directory of all files on 1 to 4 disk drives. .SUPERZAP .change disk ID.. and more $150.

(5) ADVENTURE #1-#9 (Scott Adams - Adventure International)...a series of games formally only available on the large computers .your goal is to work your way through a maze of obstacles in order to recover a secret treasure or complete a mission .the package includes all 9 Adventures written by Scott Adams. .$99.95.

(6) GSF (Racet Computers) Generalized Subroutine Facility a series of super fast machine language utilities that can be called from a BASIC program (no machine language knowledge required) sorts 1000 items in under 5 seconds...allows PEEK and POKE statements. move data blocks. compress and uncompress data. works under TRSDOS $50

(7) DSM (Racet Computes) Disk Sort Merge . sorts and merges large multiple diskette files on a 1 to 4 drive system...NOT AN IN MEMORY SORT can actually alphabetize (or any other type of sort) 4 disk drives worth of data...sorts one complete disk of information in 10 minutes. information is provided to use DSM with the RS MAILING PROGRAM works under TRSDOS...$150.

(8) RSM (Small Systems Software) .a machine language monitor and disassembler .can be used to see and modify memory or disk sectors .contains all the commands found on the Model-I version plus some additional commands for the MOD-II works under TRSDOS. .$39.95.

(9) BLINK BASIC LINK FACILITY (Racet Computes) Link from one BASIC program to another saving all variables...chain programs without losing variables ...$50.

(10) BASIC CROSS REFERENCE UTILITY (Racet Computes). .lists all variables and strings used in a program (with the line numbers in which they appear) ...lists all GOTO's and GOSUB's (with the line numbers in which they appear) searches for any specific variables or strings (with the line number in which they appear) $50.

(11) DEVELOPMENT PACKAGE (Racet Computes). SUPERZAP (to see, print or change any byte on a diskette) Disassembler and MOD-II interface to the **MICROSOFT EDITOR ASSEMBLER PLUS** including uploading services and patches for Disk I/O assemble directly into memory . save all or portions of source to disk ..dynamic debug facility (ZBUG) entended editor commands...$125

(12) HARD/SOFT DISK SYSTEM (Racet Computes) The software essential to interface any of the popular large hard disk drives completely compatible with your existing software and files allows up to 20 megabytes of storage (and larger) directory expandable to handle thousands of files...$400

(13) CAMEO HARD DISK DRIVE CONTROLLER coming soon (November 1?)

(14) HARD DISK DRIVES coming soon (Nov. 1?)

(15) H & E COMPUTRONICS, INC. SHARE-A-PROGRAM DISKETTE #1 works under TRSDOS ..a collection of programs written by MOD-II owners programs include data base management a word processor ..mail system mortgage calculations checkbook register. .and many others $8 (add $3 postage outside of the United States, Canada and Mexico) ..FREE if you send us a diskette containing a program that can be added to the SHARE-A-PROGRAM DISKETTE

(16) WABASH CERTIFIED DISKETTES $39 95 (per box of 10)

(17) FLIP SORT DISKETTE STORAGE TRAY Stores 50 diskettes comes complete with index-dividers. tilt plates and adjustable spacing $44 95

(18) MASTER PAC 100 100 essential programs. BUSINESS. PERSONAL FINANCE STATISTICS MATH GAMBLING GAMES includes 125 page manual and 2 diskettes $99 95

(19) BUSINESS PAC 100 100 essential business programs. INVENTORY CONTROL PAYROLL BOOKKEEPING SYSTEM STOCK CALCULATIONS CHECKBOOK MAINTENANCE ACCOUNTS RECEIVABLE ACCOUNTS PAYABLE includes 125 page manual and two diskettes $149 95

(20) EDITOR ASSEMBLER (Galactic Software Ltd.) the first user oriented Editor Assembler for the MODEL II and was designed to utilize all the features of the MODEL II It includes innovative features for ease of coding and debugging and complete documentation (over 120 pages) works under TRSDOS $229 00

(21) BASIC˙ COMPILER (Microsoft) changes your source programs into machine language increases program execution by 3-10 times $395

(22) MAIL/FILE SYSTEM from Galactic Software Ltd stores 2,500 names per disk No sorting time is required since the file is automatically sorted by first and last name plus Zip Code on input Retrieve by any combination of 19 user codes Supports an 11 digit alphanumerica Zip Supports a message line Comes complete with user-oriented documentation (100-page manual) Allows for company name and individual of a company and complete phone number (and extension) works under TRSDOS $199 00

(23) INCOME TAX PAC Professional income tax package most forms and schedules output to video or line printer automatic memory storage of all information data can be loaded from diskette. changed and edited built in error checking $199 95

(24) COMPUTER GAMES (SBSG) Mean Checker Machine, Star-Trek III, Concentration, Treasure Hunt, Banco, Dog Star Adventure $74 95

ALL PROGRAMS LISTED HERE REQUIRE CP/M*

(1) CP/M (Lifeboat Associates).....an alternative operating system for the **MOD-II** that allows **MOD-II** owners to use any of the hundreds of programs available under **CP/M** ..$170.

(2) CP/M HANDBOOK...(Sybex)...a step-by-step guide to **CP/M**. .takes the reader through each of the **CP/M** commands. numberous sample programs... practical hints reference tables $13.95.

(3) GENERAL LEDGER, ACCOUNTS RECEIVABLE, ACCOUNTS PAYABLE, INVENTORY CONTROL, AND PAYROLL (Peachtree Software) requires CP/M and MICROSOFT BASIC professional business systems. turn key operation .can be used as single modules or as a coordinated system $500 per module.. $2500 for the complete system

(4) WORD-STAR. .The ultimate word processor ..a menu driven word processing system that can be used with any printer All standard word processing commands are included . plus many unique commands only found on WORD STAR. requires CP/M $495

(5) MAIL LIST MERGE . An add on package that allows the user to send form letters (created on WORD-STAR) to any compiled mailing list (using any CP/M based MAIL program such as the PEACHTREE MAIL PROGRAM) .requires CP/M, WORD STAR and andy CP/M based mail program. .$150

(6) SELECTOR III (Micro-Ap) complete data management system user defined fields and codes.. manages any list defined by the user...includes additional modules for simplified inventory control, accounts receivable and accounts payable...requires CBASIC-2 $295.

(7) SELECTOR IV (Micro-Ap) . the ultimate data management system ..all features use the **SELECTOR III** plus. data file format conversions. full page report formatter computations.. global search and replace hard disk compatible oata/text merging . $550

(8) GLECTOR (Micro-Ap) add on package to the **SELECTOR** general ledger that allows the user to define a customized chart of accounts $350

(9) CBASIC-2. .a non-interactive BASIC used for many programs that run under **CP/M** ..allows user to make more efficient use of disk files. .eliminates the use of most line number references require on such programs as the **SELECTOR** $120

(10) MICROSOFT BASIC an enhanced version of the **MICROSOFT BASIC** found on **TRSDOS** adds commands such as chaining (allows the user to LOAD and RUN a new program without losing the variables currently in memory) long variable length file records, WHILE/WEND and others. can be used with the BASIC COMPILER to speed up programs (3-10 times faster execution). $325

(11) MASTER TAX (CPAids). professional tax preparation program prepares schedules, A, B, C, D, E, F, G, R/RP, SE, TC, ES and forms 2106, 2119, 2210, 3468, 3903, 2441, 4625, 4726, 4797, 4972, 5695 and 6521 Printing can be on readily available pre-printed continuous forms. on overlays. or on computer generated IRS approved forms Maintains clint history files interactive with CP/Aids General Ledger $995

(12) GENERAL LEDGER II (CPAids) designed for CPA's .stores complete 12 month detailed history of transactions generates financial statements, depreciation, loan amortizations, journals, trial balances, statements of changes in financial position, and compilation letters includes payroll system with automating posting to general ledgers prints payroll register, W2's and payroll checks $450

(13) ELECTRIC PENCIL (Michael Shrayer Software) Complete word processor with extensive editing and printer formatting features.. $275 (Standard printer version) $300 (DIABLO, NEC or QUME version)

(14) BASIC COMPILER (Microsoft). .changes your source programs into machine language...increases program execution by 3-10 times . $395

• (CP/M IS A REGISTERED TRADEMARK OF DIGITAL RESEARCH)

H & E COMPUTRONICS INC.
MATHEMATICAL APPLICATIONS SERVICE ✓9

50 N. PASCACK ROAD
SPRING VALLEY, NEW YORK 10977

NEW TOLL-FREE ORDER LINE (OUTSIDE OF N.Y. STATE)
(800) 431-2818

24 HOUR ORDER LINE
(914) 425-1535

master charge · AMERICAN EXPRESS · VISA

NEW!!! MOD-II NEWSLETTER $18/year (or 12 issues)

H & E COMPUTRONICS INC.

•••EVERYTHING FOR YOUR TRS-80™•••

TRS-80 is a trademark of the Radio Shack Division of Tandy Corporation

NEW

BUSINESS PAC 100

★ All orders processed within 24-Hours
★ 30-Day money back guarantee on all Software

100 Ready-To-Run Business Programs

(ON CASSETTE OR DISKETTE).....Includes 110 Page Users Manual.....5 Cassettes (Or Diskettes)
Inventory Control.....Payroll.....Bookkeeping System.....Stock Calculations.....
Checkbook Maintenance.....Accounts Receivable.....Accounts Payable.....

BUSINESS 100 PROGRAM LIST

1	RULE78	Interest Apportionment by Rule of the 78's
2	ANNU1	Annuity computation program
3	DATE	Time between dates
4	DAYYEAR	Day of year a particular date falls on
5	LEASEINT	Interest rate on lease
6	BREAKEVN	Breakeven analysis
7	DEPRSL	Straightline depreciation
8	DEPRSY	Sum of the digits depreciation
9	DEPRDB	Declining balance depreciation
10	DEPRDDB	Double declining balance depreciation
11	TAXDEP	Cash flow vs. depreciation tables
12	CHECK2	Prints NEBS checks along with daily register
13	CHECKBK1	Checkbook maintenance program
14	MORTGAGE/A	Mortgage amortization table
15	MULTMON	Computes time needed for money to double, triple, etc.
16	SALVAGE	Determines salvage value of an investment
17	RRVARIN	Rate of return on investment with variable inflows
18	RRCONST	Rate of return on investment with constant inflows
19	EFFECT	Effective interest rate of a loan
20	FVAL	Future value of an investment (compound interest)
21	PVAL	Present value of a future amount
22	LOANPAY	Amount of payment on a loan
23	REGWITH	Equal withdrawals from investment to leave 0 over
24	SIMPDISK	Simple discount analysis
25	DATEVAL	Equivalent & nonequivalent dated values for oblig.
26	ANNUDEF	Present value of deferred annuities
27	MARKUP	% Markup analysis for items
28	SINKFUND	Sinking fund amortization program
29	BONDVAL	Value of a bond
30	DEPLETE	Depletion analysis
31	BLACKSH	Black Scholes options analysis
32	STOCVAL1	Expected return on stock via discounts dividends
33	WARVAL	Value of a warrant
34	BONDVAL2	Value of a bond
35	EPSEST	Estimate of future earnings per share for company
36	BETAALPH	Computes alpha and beta variables for stock
37	SHARPE1	Portfolio selection model-i.e. what stocks to hold
38	OPTWRITE	Option writing computations
39	RTVAL	Value of a right
40	EXPVAL	Expected value analysis
41	BAYES	Bayesian decisions
42	VALPRINF	Value of perfect information
43	VALADINF	Value of additional information
44	UTILITY	Derives utility function
45	SIMPLEX	Linear programming solution by simplex method
46	TRANS	Transportation method for linear programming
47	EOQ	Economic order quantity inventory model
48	QUEUE1	Single server queueing (waiting line) model
49	CVP	Cost-volume-profit analysis
50	CONDPROF	Conditional profit tables
51	OPTLOSS	Opportunity loss tables
52	FQUOQ	Fixed quantity economic order quantity model
	NAME	**DESCRIPTION**
53	FQEOWSH	As above but with shortages permitted
54	FQEOQPB	As above but with quantity price breaks
55	QUEUECB	Cost-benefit waiting line analysis
56	NCFANAL	Net cash-flow analysis for simple investment
57	PROFIND	Profitability index of a project
58	CAP1	Cap. Asset Pr. Model analysis of project
59	WACC	Weighted average cost of capital
60	COMPBAL	True rate on loan with compensating bal. required
61	DISCBAL	True rate on discounted loan
62	MERGANAL	Merger analysis computations
63	FINRAT	Financial ratios for a firm
64	NPV	Net present value of project
65	PRINDLAS	Laspeyres price index
66	PRINDPA	Paasche price index
67	SEASIND	Constructs seasonal quantity indices for company
68	TIMETR	Time series analysis linear trend
69	TIMEMOV	Time series analysis moving average trend
70	FUPRINF	Future price estimation with inflation
71	MAILPAC	Mailing list system
72	LETWRT	Letter writing system-links with MAILPAC
73	SORT3	Sorts list of names
74	LABEL1	Shipping label maker
75	LABEL2	Name label maker
76	BUSBUD	DOME business bookkeeping system
77	TIMECLCK	Computes weeks total hours from timeclock info.
78	ACCTPAY	In memory accounts payable system-storage permitted
79	INVOICE	Generate invoice on screen and print on printer
80	INVENT2	In memory inventory control system
81	TELDIR	Computerized telephone directory
82	TIMUSAN	Time use analysis
83	ASSIGN	Use of assignment algorithm for optimal job assign.
84	ACCTREC	In memory accounts receivable system-storage ok
85	TERMSPAY	Compares 3 methods of repayment of loans
86	PAYNET	Computes gross pay required for given net
87	SELLPR	Computes selling price for given after tax amount
88	ARBCOMP	Arbitrage computations
89	DEPRSF	Sinking fund depreciation
90	UPSZONE	Finds UPS zones from zip code
91	ENVELOPE	Types envelope including return address
92	AUTOEXP	Automobile expense analysis
93	INSFILE	Insurance policy file
94	PAYROLL2	In memory payroll system
95	DILANAL	Dilution analysis
96	LOANAFFD	Loan amount a borrower can afford
97	RENTPRCH	Purchase price for rental property
98	SALELEAS	Sale-leaseback analysis
99	RRCONVBD	Investor's rate of return on convertable bond
100	PORTVAL9	Stock market portfolio storage-valuation program

☐ **CASSETTE VERSION**	**$99.95**
☐ **DISKETTE VERSION**	**$99.95**
☐ **TRS-80* MODEL II VERSION**	**$149.95**

ADD $3.00 FOR SHIPPING IN UPS AREAS
ADD $4.00 FOR C.O.D. OR NON-UPS AREAS
ADD $5.00 OUTSIDE U.S.A, CANADA & MEXICO

H & E COMPUTRONICS INC.
MATHEMATICAL APPLICATIONS SERVICE™

50 N. PASCACK ROAD
SPRING VALLEY, NEW YORK 10977

NEW TOLL-FREE ORDER LINE (OUTSIDE OF N.Y. STATE)
(800) 431-2818

AMERICAN EXPRESS · VISA · master charge

24 HOUR ORDER LINE
(914) 425-1535

H & E COMPUTRONICS INC.

•••EVERYTHING FOR YOUR TRS-80™•••

TRS-80 is a trademark of the Radio Shack Division of Tandy Corporation

PAYROLL

Payroll invoices many complex calculations and the production of reports and documents, many of which are required by government agencies. It is an ideal candidate for the computer. With this Payroll system in-house, you can promptly and accurately pay your employees and generate accruate documents/reports to management, employees, and appropriate government agencies concerning earnings, taxes, and other deductions. The package has been converted to the TRS-80™ and is now a well documented, op-line, interactive, micro-computer system with the capabilities of (or exceeding) many larger systems.

CAPABILITIES:

★ performs all necessary payroll tasks including:
- file maintenance, pay data entry and verification
- computation of pay and deduction amounts
- printing of reports and checks

★ can handle salaried and hourly employees

★ employees can receive:
- hourly or salary wage
- vacation pay
- holiday pay
- piecework pay
- overtime pay

★ employees can be paid using any combination of pay types (except, hourly cannot receive salary and salary cannot receive hourly)

★ special non-taxable or taxable lump sums can be paid regularly or one time (bonus, reimbursements, etc)

★ health and welfare deductions can be automatically calculated for each employee

★ earnings-to-date are accumulated and added to permanent records; taxes are computed and deducted: US income tax, Social Security tax, state income tax, other deductions (regular or one time)

★ paychecks are printed; computer checks with your company letter-head can be purchased from SBSG

★ calculations are accumulated for; employee pay history, 941A report, W-2 report, insurance report, absentee report

★ fully linked to General Ledger. Each employee's payroll information can be distributed to as many as (12) twelve different GL accounts; system automatically posts to cash account

GENERAL LEDGER

The General Ledger accounting system consolidates financial data from other accounting subsystems (A/R, A/P, Payroll, direct posting) in an accurate and timely manner. Major reports include the Income Statement and Balance Sheet and a "special" report designed by management. The beauty of this General Ledger system is that it is completely user formatted. You "customize" the account numbers, descriptions, and report formats to suit particular business requirements. These programs were developed 5 years ago for the Wang micro-computer and have been tested in many environments since then. The package has been converted to the TRS-80™ and is now a well documented, on-line, interactive micro-computer system with the capabilities of (or exceeding) many larger systems.

CAPABILITIES:

★ more than 200 chart of accounts can be handled

★ account number structure is user defined and controlled

★ more than 1,750 transactions may be entered via:
- direct posting; done by hand; validated against the account file before acceptance
- external posting; generated by A/R, A/P, Payroll or any other user source

★ data is maintained and reported by:
- month
- quarter
- year
- previous three quarters

★ reports (samples on back) include:
- trial balances
- income statement
- balance sheet
- special accounts reports and more.....

★ user formats reports with the following designated as you wish:
- titles
- headings
- account numbers
- descriptions
- subtotals
- totals
- skip lines
- skip pages

★ up to eight levels of totals - fully user designated

★ menu driven; easy to use; full screen prompting and cursor control

INVENTORY CONTROL/INVOICING

★ **ISAM** (Indexed Sequential Access Method) eliminates the necessity for time consuming sort.

★ Pre-Allocated Files for IMMEDIATE update and inquiry capabilities.

★ Fast Disk storage and retrieval.

★ Inventory Master Record includes...class...SKU...Division...Retail...Cost...Beginning Balance...Period Sale Units...Period Receipts...On Order...On Hand...Minimum Reorder Point...Recommended Re-order Amount...Vendor Number...Period Sale Dollars...YTD Sale Units...YTD Sale Dollars.

★ Calculated and Displayed Formulas include...Gross Margin ($)...Gross Margin (%)...Gross Margin ROI (%)...Average Inventory Retail ($)...Average Inventory Cost ($)...Turn-Over (%).

★ Reports Generated include...Master File Listing...Class Description Listing...Transaction Audit Trail...Minimum Reorder Point by Vendor...Retail Price List...Retail & Cost Price List...Period Sales Report ...Year to Date Sales Report...Stock Status (Screen or printer output) ...Commission Report (for salesmen and buyers).

★ Transaction Types include...Sales, Vendor Receipts...Vendor Orders...Customer Returns...Vendor Returns...Transfer Stock.

H & E COMPUTRONICS INC.

50 N. PASCACK ROAD
SPRING VALLEY, NEW YORK 10977

NEW TOLL-FREE
ORDER LINE
(OUTSIDE OF N.Y. STATE)
(800) 431-2818

AMERICAN EXPRESS · master charge · VISA

24 HOUR ORDER LINE
(914) 425-1535

PRICING	MOD-1 VERSION	MOD-II VERSION	MOD-III VERSION
ACCOUNTS RECEIVABLE	$125	$225	$199.95
ACCOUNTS PAYABLE	$125	$225	$199.95
GENERAL LEDGER	$125	$225	$199.95
PAYROLL	$125	$225	$199.95
INVENTORY	$175	$275	$199.95
INVOICING	$150	$250	$199.95
COORDINATED INVENTORY/INVOICING ACCOUNTS RECEIVABLE	$449	$749	$599.95
COORDINATED AR-AP-GL	$375	$675	$599.95
COORDINATED AR-AP-GL with PAYROLL	$495	$899	$799.95
EXTENDED COORDINATED AR-AP-GL INVOICING/INVENTORY with PAYROLL	$799	$1299	$1199.95

H & E COMPUTRONICS INC.

••• EVERYTHING For Your TRS-80* Model I or Model III •••

* TRS-80™ is a trademark of Tandy Corp

- All orders processed within 24 Hours
- 30-Day money back guarantee
- Add $3.00 for shipping in UPS Areas
- Add $4.00 for C.O.D. or NON-UPS Areas
- Add $5.00 to Canada or Mexico
- Add exact postage to all other countries

REMSOFT, Inc.

Let Your TRS-80™ Teach You ASSEMBLY LANGUAGE

REMASSEM-1

Tired of buying book after book on assembly language programming and still not knowing your **POP** from your **PUSH?**

REMSOFT proudly announces a more efficient way, using your own TRS-80™ to learn the fundamentals of assembly language programming at YOUR pace and at YOUR convenience.

Our unique package, **"INTRODUCTION TO TRS-80™ ASSEMBLY PROGRAMMING"**, will provide you with the following:

- ★ Ten 45-minute lessons on audio cassettes.
- ★ A driver program to make your TRS-80™ video monitor serve as a blackboard for the instructor
- ★ A display program for each lesson to provide illustration and reinforcement for what you are hearing
- ★ A textbook on TRS-80™ Assembly Language Programming
- ★ Step-by-step dissection of complete and useful routines to test memory and to gain direct control over the keyboard, video monitor, and printer.
- ★ How to access and use powerful routines in your Level II ROM

This course was developed and recorded by Joseph E. Willis and is based on the successful series of courses he has taught at Meta Technologies Corporation, the Radio Shack Computer Center, and other locations in Northern Ohio. The minimum system required is a Level II, 16K RAM.

LEARN TRS-80™ ASSEMBLY LANGUAGE DISK I/O

REMDISK-1

Your disk system and you can really step out with **REM**SOFT's Educational Module, **REMDISK-1**, a "short course" revealing the details of DISK I/O PROGRAMMING using assembly language.

Using the same format as our extremely popular introduction to assembly language programming, this **"ASSEMBLY LANGUAGE DISK I/O PROGRAMMING"** course includes:

- ★ Two 45-minute lessons on audio cassette.
- ★ A driver program to make your TRS-80™ video monitor serve as a blackboard for the instructor.
- ★ A display program for each lesson to provide illustration and reinforcement for what you are hearing.
- ★ A booklet of comprehensive, fully-commented program listings illustrating sequential file I/O, random-access file I/O, and track and sector I/O.
- ★ A diskette with machine-readable source codes for all programs discussed, in both Radio shack EDTASM and Macro formats
- ★ Routines to convert from one assembler format to the other.

This course was developed and recorded by Joseph E Willis, for the student with experience in assembly language programming, it is an intermediate-to advanced-level course. Minimum hardware required is a Model I Level II, 16K RAM one disk drive system.

COMPUTRONICS

50 N. PASCACK ROAD
SPRING VALLEY, NEW YORK 10977

AMERICAN EXPRESS · Master Charge · VISA

24 HOUR ORDER LINE
(914) 425-1535

NEW TOLL-FREE ORDER LINE
(OUTSIDE OF N.Y. STATE)
(800) 431-2818

PLEASE SEND ME:

- ☐ REMASSEM-1 (TRS-80 MODEL I CASSETTE) $69.95
- ☐ REMASSEM-1 (TRS-80 MODEL I DISKETTE) $74.95
- ☐ REMASSEM-1 (TRS-80 MODEL III CASSETTE) $74.95
- ☐ REMASSEM-1 (TRS-80 MODEL III DISKETTE) $79.95
- ☐ REMDISK-1 (TRS-80 MODEL I DISKETTE) $29.95
- ☐ REMDISK-1 (TRS-80 MODEL III DISKETTE) $34.95

CREDIT CARD NUMBER .... EXP DATE ....

SIGNATURE .... NAME ....

ADDRESS .... CITY .... STATE .... ZIP ....

*** ALL PRICES AND SPECIFICATIONS SUBJECT TO CHANGE ***

Write your first program with the help of a flying saucer.

Alternate Course—Part II

Michael A. Duffin
1507 East Ave.
Berwyn IL 60402

One of the hardest things in data processing is writing your first program.

To help alleviate this problem I gave my class Flying Saucer (Program Listing 1) to enter into the computer. The intended results of this program can be seen in Fig. 1.

Those numbers on the left are line numbers, and the computer uses them so it knows what line to execute first. They may be typed into the computer in any order, and the computer will place them in the correct sequence. If you make a mistake, just type the line over again.

One thing you must remember is that when you are finished typing on each line you must hit the white Enter key. This tells the computer that you are finished entering a line. If you want to make sure that you have typed in all of the lines correctly then type LIST into the computer. This causes the computer to list all of the lines with line numbers that you have typed into the computer. When you are ready to see your picture, type RUN into the computer. Don't forget to hit the Enter key after you have typed in list or run.

As far as the instructions used in the program, CLS means clear the screen. This causes the computer to erase everything that's on the screen.

PRINT@ tells the computer to print something at a specified spot on the screen. If you look at a TRS-80 Video Display Worksheet you will see the numbers 0 through 63 at the top of the sheet. At the right of the sheet are the numbers 63, 127 and so on. At the left are the numbers 0, 64, 128 and so on.

These numbers are used with the PRINT@ instruction. The number represents the corresponding positions on the screen. The numbers 0 through 1023 are valid for this instruction. The second operand of the PRINT@ instruction is the character or groups of characters that you want printed at the specified location. If we look at line 20 of the program:

```
20 PRINT@ 158,"()"
```

we see that the characters ( ) will be printed starting at position 158. These characters must be included in quotes.

One other point of interest about this program concerns lines 70, 80 and 90. The starting position in lines 20 through 60 was determined by taking the number to the left of the characters for that line and then adding the number from the top that is just above the first character of that line. Thus, in line 20, the numbers 128 + 30 (158) represent the starting position. Since the computer can add more reliably than we can, lines 70, 80, and 90 let the computer add the two numbers to determine the

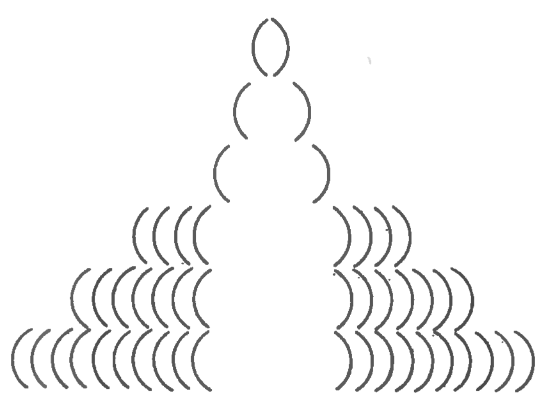


Fig. 1

```
5 REM ***   FLYING SAUCER ***
10 CLEAR
20 PRINT@ 158,"()"
30 PRINT@ 221,"(  )"
40 PRINT@ 284,"(    )"
50 PRINT@ 344,"(((    ))))"
60 PRINT@ 384+21,"(((((((      )))))))"
70 PRINT@ 448+18,"(((((((((((      )))))))))))"
80 PRINT@ 576+24,"THIS   IS   A"
90 PRINT@ 640+14,"FLYING                    SAUCER"
```

```
5 REM *** FLYING SAUCER MODIFIED ***
10 CLS
20 PRINT@ 158,"()"
30 PRINT@ 221,"(  )"
40 PRINT@ 284,"(    )"
50 PRINT@ 344,"((((      ))))"
60 PRINT@ 384+21,"(((((((      )))))))"
70 PRINT@ 448+17,"(((((((((((      )))))))))))"
80 PRINT@ 576+24,"THIS   IS   A"
90 PRINT@ 639+14,"FLYING                    SAUCER"
100 PRINT@ 960+30,"!!"
110 FOR X=1 TO 100
120 NEXT X
130 GOTO 100
```

Please call (213) 883-8594.

Our Epson prices are so low, we're not allowed to print them.

EPSON MX-80 .... $CALL

80 cps/9x9 matrix/Lower case with true descenders/Bi-directional & Logic seeking/Adjustable tractor/ Expanded printing/Block graphics/ Forms control/Compressed printing/ Double-strike printing/Correspondence quality/Emphasized printing mode/Standard parallel interface

EPSON MX-80 F/T ...... $CALL

Same features as the MX-80 plus Friction Feed. Adjustable removable tractor is standard for ease of handling forms and single sheets.

EPSON MX-100 F/T ..... $CALL

Same features as the MX-80 & MX-80 F/T but on 15½ inch carriage for printing 132 columns with standard 10 cpi font or 232 columns in the compressed character font. The MX-100 is complete with Dot Resolution Graphics

EPSON INTERFACES & OPTIONS

TRS-80 MODEL I, III CABLE	$	30	SERIAL INTERFACE (2K BUFFER)	$	149
TRS-80 MODEL I Keyboard Interface	$	95	SERIAL CABLE Male to Male	$	30
TRS-80 MODEL II CABLE	$	30	DOT RESOLUTION GRAPHICS	$	90
APPLE INTERFACE & CABLE	$	100	REPLACEMENT RIBBON	$	13
IEEE 488 INTERFACE	$	60	REPLACEMENT PRINT HEAD (Quiet type)	$	40
SERIAL INTERFACE	$	70	EPSON SERVICE MANUAL	$	40

Alpha Byte STORES

We built a reputation on our prices and your satisfaction.

We guarantee everything we sell for 30 days. If anything is wrong, just return the item and we'll make it right. And, of course, we'll pay the shipping charges.

We accept Visa and Master Card on all orders. COD accepted up to $300.00. We also accept school purchase orders.

Please add $2.00 for standard UPS shipping and handling on orders under 50 pounds, delivered in the continental U.S. Call us for shipping charges on items that weigh more than 50 pounds. Foreign, FPO and APO orders please add 15% for shipping. California residents add 6% sales tax.

✓69

31245 LA BAYA DRIVE, WESTLAKE VILLAGE, CALIFORNIA 91362

starting position.

After this brief explanation of the program I instructed my students to type the program into their machines and run it. Then I asked them to modify the program so that the picture on the screen matched the picture on the worksheet. At this time all modifications were accomplished by retyping the line that was in error.

Most of the students had no problems with this program except for the fact that a few of them weren't very good at typing, but then again neither am I.

Once everyone had completed the exercise I gave them the following four lines:

```
100  PRINT@ 960+30,"!!"
110  FOR X=1 TO 100
120  NEXT X
130  GOTO 100
```

Along with these lines I gave them the following instructions:

1) Add lines 100 and 130 only to the saucer program.

2) Type RUN

a)What happens? Why?

b)What does line 130 do?

c)How do you stop the program?

3) Now type in lines 110 and 120. (The program should now consist of lines 10 through 130).

4) Type RUN

a)How has the program changed?

5) Change the number 100 in line 110 to 1000.

6) Type RUN

a)How has the program changed?

```
##################
############
######
##
######
############
##################
############
######
##
######
############
##################
```

Fig. 2

```
5 CLS
6 REM *** SYMMETRIC DESIGN BY ANDREA BAAR ***
10 PRINT @ 86, "xxxxxxxxxxxxxxxxxx"
15 PRINT @ 153, "xxxxxxxxxxxx"
20 PRINT @ 220, "xxxxxx"
25 PRINT @ 286, "xx"
30 PRINT @ 348, "xxxxxx"
35 PRINT @ 383+26,"xxxxxxxxxxxx"
40 PRINT @ 447+23, "xxxxxxxxxxxxxxxxxx"
45 PRINT @ 511+26, "xxxxxxxxxxxx"
50 PRINT @ 575+29, "xxxxxx"
55 PRINT @ 639+31, "xx"
60 PRINT @ 703+29, "xxxxxx"
65 PRINT @ 767+26, "xxxxxxxxxxxx"
70 PRINT @ 831+23, "xxxxxxxxxxxxxxxxxx"
```

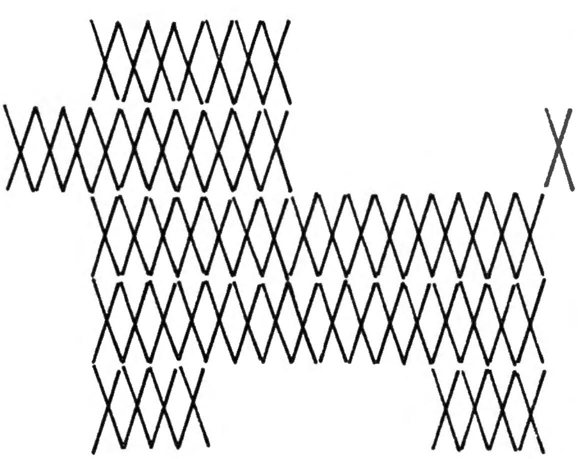


Fig. 3

```
10 CLS
20 PRINT@ 280, "XXXXXXX"
30 PRINT@ 341, "XXXXXXXXXX          X"
40 PRINT@ 408, "XXXXXXXXXXXXXXXX"
50 PRINT@ 447+25, "XXXXXXXXXXXXXXX"
60 PRINT@ 511+25, "XXXX          XXXX"
70 PRINT@ 639+22, "THIS IS A DOG"
```

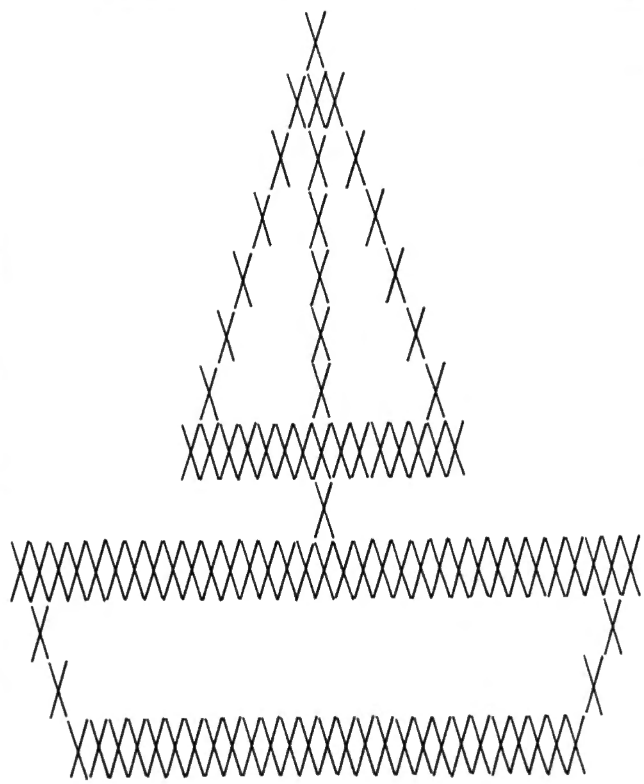


Fig. 4

```
5 REM *** SHIP BY LIRI HODJA ***
10 CLS
20 PRINT@ 95,"X"
30 PRINT@ 158,"XXX"
40 PRINT@ 221,"X X X"
50 PRINT@ 284,"X  X  X"
60 PRINT@ 319+28,"X   X   X"
70 PRINT@ 383+27,"X    X    X"
80 PRINT@ 447+26,"X     X     X"
90 PRINT@ 511+25,"XXXXXXXXXXXXXXX"
100 PRINT@ 575+32,"X"
110 PRINT@ 639+15,"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  X"
120 PRINT@ 703+16,"X                                  X"

130 PRINT@ 767+17,"X                                X"
140 PRINT@ 831+18,"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
```

7) Change the number 1000 in line 110 to 10.

8) Type RUN

a)How has the program changed?

b)What do lines 110 and 120 do?

c)How does the program work?

For those of you who don't have a TRS-80 handy, I'll explain the answers to the questions above.

2a. What happens is the flying saucer flies up off the screen. Then, what seems to happen is the characters "!!" stay on the screen.

2b. Line 130 goes back to line 100 and prints !! again at position 990.

2c. There are a couple of ways the program can be stopped. One way of course is to turn off or unplug the machine. The big disadvantage in doing this is that you have to retype the entire program into the machine so you can answer question number 3. A better method of stopping this program or any program is to hit the Break key.

4a. The saucer still flies off the screen but it does this at a slightly slower rate. This time it can also be seen that the characters !! seem to be blinking.

6a. By changing the number in lines 110 to 1000 the saucer flies up off the screen at a much slower speed and, yes, this time the characters !! are definitely blinking.

8a. This time the saucer flies off the screen at a much faster rate.

```
            #      #
            #      #
            ########
            #      #
             ######
               # #
          #####  #####
          #           #   #
          #           # #
          #           ##
      ####                ####
############################
```

Fig. 5

```
10 CLS
20 PRINT@ 63+25, "#      #"
30 PRINT@ 127+25, "#      #"
40 PRINT@ 191+25,"########"
50 PRINT@ 255+25,"#      #"
60 PRINT@ 319+26,"######"
70 PRINT@ 383+28,"# #"
80 PRINT@ 447+23,"#####  #####"
90 PRINT@ 511+23,"#           #   #"
100 PRINT@ 575+23,"#           # #"
110 PRINT@ 639+23,"#           ##"
120 PRINT@ 703+19,"####                ####"
130 PRINT@ 767+19,"############################"
```

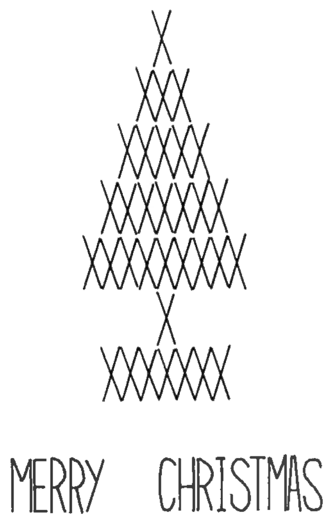


Fig. 7

```
5 REM ***   CHRISTMAS TREE BY CAROL ANN VOJTIK ***
10 CLS
20 PRINT@ 157,"X"
30 PRINT@ 191+29,"XXX"
40 PRINT@ 191+64+28,"XXXXX"
50 PRINT@ 319+27,"XXXXXXX"
60 PRINT@ 383+26,"XXXXXXXXX"
70 PRINT@ 447+30,"X"
80 PRINT@ 511+27,"XXXXXXX"
90 PRINT@ 639+22,"MERRY   CHRISTMAS"
```

8b. My students easily ascertained that line 110 somehow controls the speed that the saucer moves. The relationship of line 120 and the actual mechanics of a For...Next loop evaded most of them since they had never been exposed to a For...Next loop before. What lines 110 and 120 really do is add 1. Line 110 starts with X= then line 120 causes X to be incremented until X equals the number after "to" in line 110.

8c. Line 10 blanks the screen. Lines 20 through 90 build the saucer and write the message. Line 100 causes the characters "!!" to be written on the last line of the screen. Lines 110 and 120 cause the saucer to move up slower. Line 130 goes back to line 100 and the instructions 100 through 130 keep repeating because of this.

By the way, line 130 is an example of an unconditional branch. Lines 110 and 120 are an example of a conditional branch. An unconditional branch is always executed. (e.g. Line 130 always goes back to line 100 while the program is executing).

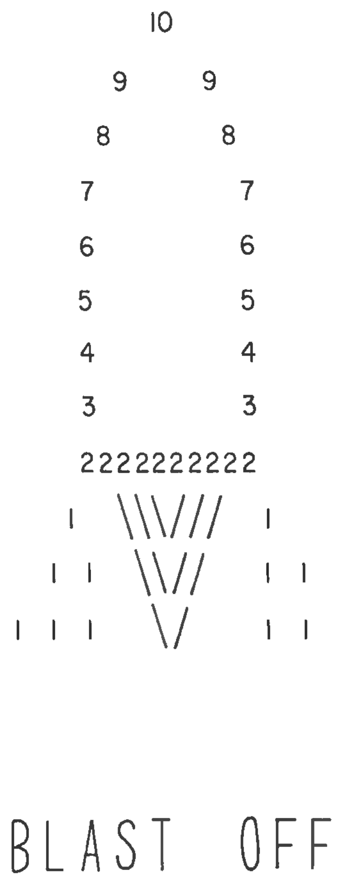


Fig. 6

```
5 REM *** SPACE SHIP BY TONY PUSATERI ***
10 CLS
20 PRINT@ 31,"10"
30 PRINT@ 64+29,"9    9"
40 PRINT@ 156,"8      8"
50 PRINT@ 219,"7        7"
60 PRINT@ 283,"6        6"
70 PRINT@ 347,"5        5"
80 PRINT@ 411,"4        4"
90 PRINT@ 475,"3        3"
100 PRINT@ 539,"2222222222"
110 PRINT@ 602,"1  <<<>>>  1"
120 PRINT@ 664,"1 1   <<>>   1 1"
130 PRINT@ 726,"1 1 1    <>  1 1 1"
140 PRINT@ 798,"1111"
150 PRINT@ 862,"1111"
160 PRINT@ 926,"1111"
170 PRINT@ 983,"B L A S T    O F F"
180 GOTO 180
```

✓See List of Advertisers on page 354

A conditional branch only occurs when certain conditions are true (e.g., lines 110 and 120 cause X to increment until X equals the value following the "to" in line 110. When X equals this value then the next line is executed).

You're probably wondering what caused that saucer to fly up off of the screen. Before explaining this to my students I felt they needed to know a little bit about hardware and software.

Hardware is, basically, the physical machine itself. Software are the programs that are put inside the machine. Software can be divided into three groups: operating system software, application programs and translators. Operating system software allows the various pieces of hardware to interact and allows a programmer to talk to the computer. An application program is one that performs a specific function or group of functions. These functions are not usually system related and they are usually written to perform a job. A translator is the software which takes an application program and converts it so the machine can understand it. A translator takes what the user types in (i.e. the source code) and converts it to machine code (i.e., the object code).

On the TRS-80 when the PRINT@ instruction is executed the cursor is automatically placed on the line following the one on which the characters were printed by the translator. (The cursor indicates where you are on the screen.)

When you type something on the last line on the screen, the operating system automatically causes all of the material on that screen to be moved up one line when the Enter key is hit or if the data typed in extends more than the length of one line.

Line 100 causes a combination of both of these to occur. The translator prints !! on the last line and then attempts to place the cursor on the next line. The system thinks the data has extended to the next line so it causes the other material on the screen to be moved up one line. This makes the saucer seem to fly away.

On a microcomputer there are two types of memory where these programs are found, RAM and ROM. Translators are operally found in ROM (i.e., Read Only Memory). This means that a pro-
Memory). This means that a pro-
grammer can read the material in this type of memory, but he or she can't change it. Applications programs are usually stored in RAM (i.e., Random Access Memory). This type of memory can be either read by or written to by an applications program.

At the conclusion of the above exercises I asked my students to make some sort of picture on a Video Display Worksheet and then use the PRINT@ instruction to program their diagrams. Some of the diagrams and their associated programs are included at the end.

The last program in this article is called Hidden Pictures, written by Carol Vojtik. However, as I'm sure you have surmised, this program was written at the end of the class.

Lines 150 and 2810 are of special interest and can be used in any program. What they do is they each cause the program to wait until a key is hit.

As you can see from Carol's two programs, the students have come a long way in their ability to use the Basic language.■

Program Listing 1

```
1 REM ********************************************
2 REM *
3 REM *     HIDDEN PICTURES BY CAROL VOJTIK
5 REM *
7 REM **********************************************
10 CLS
20 PRINT @ 87, "HIDDEN PICTURES"
30 PRINT " "
40 PRINT "I WILL DRAW A MAZE"
50 PRINT "SEE IF YOU CAN FIND:"
60 PRINT " "
70 PRINT "AN ARROW"
80 PRINT "A BEARS HEAD"
90 PRINT "A BUG"
100 PRINT "A DUCK"
110 PRINT "A SHOVEL"
120 PRINT "A TEE-PEE"
130 FOR X=1 TO 3000: NEXT
140 PRINT "HIT ANY KEY TO CONTINUE"
150 A$=INKEY$: IF A$="" THEN 150
160 CLS
180 FOR X=15360 TO 15447
200 POKE X, 166
210 NEXT X
220 POKE 15448, 175
230 POKE 15449, 182
240 POKE 15450, 190
250 POKE 15451, 167
260 FOR X=15452 TO 15461
270 POKE X, 166
290 NEXT X
310 POKE 15462, 190
320 POKE 15463, 167
330 POKE 15464, 166
340 POKE 15465, 182
350 FOR X=15466 TO 15495
360 POKE X, 166
370 NEXT X
380 POKE 15496, 167
390 POKE 15497, 191
400 POKE 15498, 167
410 FOR X=15499 TO 15508
420 POKE X, 166
430 NEXT X
440 POKE 15509, 174
450 POKE 15510, 174
460 POKE 15511, 174
470 POKE 15512, 191
480 POKE 15513, 191
485  POKE 15514, 174
486 POKE 15515, 174
490 FOR X=15516 TO 15524
500 POKE X, 166
520 NEXT X
540 POKE 15525, 190
550 POKE 15526, 166
560 POKE 15527, 166
570 POKE 15528, 166
580 POKE 15529, 166
590 POKE 15530, 174
600 POKE 15531, 166
610 FOR X=15532 TO 15537
630 POKE X, 182
640 NEXT X
650 FOR X=15538 TO 15560
660 POKE X, 166
670 NEXT X
680 POKE 15561, 191
690 FOR X=15562 TO 15577
700 POKE X, 166
710 NEXT X
720 POKE 15578, 175
730 POKE 15579, 182
740 FOR X=15580 TO 15588
750 POKE X, 166
760 NEXT X
780 POKE 15589, 183
790 FOR X=15590 TO 15595
800 POKE X, 166
810 NEXT X
820 POKE 15596, 174
830 POKE 15597, 182
840 FOR X=15598 TO 15601
850 POKE X, 166
860 NEXT X
870 POKE 15602, 175
880 POKE 15603, 182
890 FOR X=15604 TO 15624
900 POKE X, 166
910 NEXT X
920 POKE 15625, 191
930 FOR X=15626 TO 15667
940 POKE X, 166
950 NEXT X
960 POKE 15668, 167
970 POKE 15669, 174
980 POKE 15670, 174
990 POKE 15671, 174
1000 POKE 15672, 182
1010 FOR X=15673 TO 15686
1020 POKE X, 166
1030 NEXT X
1040 POKE 15687, 191
1050 POKE 15688, 191
1060 POKE 15689, 191
1070 POKE 15690, 191
1080 POKE 15691, 191
1090 FOR X=15692 TO 15704
1100 POKE X, 166
1110 NEXT X
1120 POKE 15705, 174
1130 POKE 15706, 167
1140 POKE 15707, 174
1150 POKE 15708, 174
1160 POKE 15709, 182
1170 POKE 15710, 182
1180 FOR X=15711 TO 15718
1190 POKE X, 166
1200 NEXT X
1210 POKE 15719, 167
1220 FOR X=15720 TO 15735
1230 POKE X, 166
1240 NEXT X
1250 POKE 15736, 182
1260 POKE 15737, 167
1270 FOR X=15738 TO 15769
1280 POKE X, 166
1290 NEXT X
1300 POKE 15770, 187
1310 POKE 15771, 166
1320 POKE 15772, 166
1330 POKE 15773, 166
1340 POKE 15774, 166
1350 POKE 15775, 183
1360 POKE 15776, 183
1370 POKE 15777, 174
1380 POKE 15778, 174
1390 POKE 15779, 166
1400 POKE 15780, 166
1410 POKE 15781, 166
1420 POKE 15782, 166
1430 POKE 15783, 166
1440 POKE 15784, 175
1450 POKE 15785, 182
1460 FOR X=15786 TO 15796
1470 POKE X, 166
1480 NEXT X
1490 POKE 15797, 167
1500 POKE 15798, 167
1510 POKE 15799, 167
1520 FOR X=15800 TO 15832
1530 POKE X, 166
1540 NEXT X
1550 POKE 15833, 174
1560 POKE 15834, 182
1570 POKE 15835, 174
1580 POKE 15836, 174
1590 POKE 15837, 167
1600 POKE 15838, 167
1610 FOR X=15839 TO 15849
1620 POKE X, 166
1630 NEXT X
```

Program continues

```
1640 POKE 15850, 175
1650 FOR X=15851 TO 15859
1660 POKE X, 174
1670 NEXT X
1680 POKE 15860, 167
1690 FOR X=15861 TO 15876
1710 POKE X, 166
1720 NEXT X
1730 FOR X=15877 TO 15880
1740 POKE X, 174
1750 NEXT X
1760 POKE 15881, 190
1770 POKE 15882, 174
1780 POKE 15883, 167
1790 POKE 15884, 167
1800 POKE 15885, 167
1810 POKE 15886, 175
1820 POKE 15887, 174
1830 POKE 15888, 182
1840 POKE 15889, 182
1850 FOR X=15890 TO 15941
1860 POKE X, 166
1870 NEXT X
1880 POKE 15942, 167
1890 POKE 15943, 183
1900 POKE 15944, 175
1910 POKE 15945, 166
1920 POKE 15946, 174
1930 FOR X=15947 TO 15953
1940 POKE X, 166
1950 NEXT X
1960 POKE 15954, 175
1970 FOR X=15955 TO 15980
1980 POKE X, 166
1990 NEXT X
2000 POKE 15981, 182
2010 FOR X=15982 TO 15990
2020 POKE X, 166
2030 NEXT X
2040 POKE 15991, 182
2050 FOR X=15992 TO 16005
2060 POKE X, 166
2070 NEXT X
2080 POKE 16006, 174
2090 POKE 16007, 174
2100 POKE 16008, 167
2110 POKE 16009, 174
2120 POKE 16010, 182
2130 FOR X=16011 TO 16016
2140 POKE X, 166
2150 NEXT X
2160 POKE 16017, 182
2170 FOR X=16018 TO 16042
2180 POKE X, 166
2190 NEXT X
2195 POKE 16043, 174
2200 POKE 16044, 191
2210 FOR X=16045 TO 16053
2220 POKE X, 174
2230 NEXT X
2240 POKE 16054, 183
2250 FOR X=16055 TO 16073
2260 POKE X, 166
2270 NEXT X
2280 POKE 16074, 190
2290 POKE 16075, 167
2300 POKE 16076, 166
2310 POKE 16077, 166
2320 POKE 16078, 182
2330 POKE 16079, 174
2340 POKE 16080, 167
2350 FOR X=16081 TO 16108
2360 POKE X, 166
2370 NEXT X
2380 POKE 16109, 167
2390 FOR X=16110 TO 16118
2400 POKE X, 166
2410 NEXT X
2420 POKE 16119, 167
2430 FOR X=16120 TO 16142
2440 POKE X, 166
2450 NEXT X
2460 POKE 16143, 167
2470 POKE 16144. 167
2480 POKE 16145, 167
2490 POKE 16146, 167
2500 POKE 16147, 175
2510 POKE 16148, 174
2520 POKE 16149, 174
2530 POKE 16150, 167
2540 POKE 16151, 175
2550 POKE 16152, 190
2560 POKE 16153, 167
2570 FOR X=16154 TO 16200
2580 POKE X, 166
2590 NEXT X
2600 POKE 16201, 182
2610 FOR X=16202 TO 16265
2620 POKE X, 166
2630 NEXT X
2640 POKE 16266, 167
2650 POKE 16267, 174
2660 POKE 16268, 174
2670 POKE 16269, 174
2680 POKE 16270, 182
2690 POKE 16271, 182
2700 POKE 16272, 182
2710 POKE 16273, 182
2720 POKE 16274, 174
2730 POKE 16275, 174
2740 POKE 16276, 174
2750 POKE 16277, 167
2760 POKE 16278, 167
2770 FOR X=16279 TO 16285
2780 POKE X, 166
2790 NEXT X
2800 PRINT @927, "HIT ANY KEY TO SEE THE RESULTS"
2810 B$=INKEY$: IF B$="" THEN 2810
2820 FOR X=15360 TO 15445
2823 POKE X, 32
2826 NEXT X
2830 POKE 15446, 130
2840 POKE 15447, 164
2850 POKE 15448, 169
2860 POKE 15449, 150
2870 POKE 15450, 152
2880 POKE 15451, 129
2890 FOR X=15452 TO 15461
2900 POKE X, 32
2910 NEXT X
2920 POKE 15462, 152
2930 POKE 15463, 131
2940 POKE 15464, 164
2950 POKE 15465, 144
2960 FOR X=15466 TO 15495
2970 POKE X, 32
2980 NEXT X
2990 POKE 15496, 131
3000 POKE 15497, 191
3010 POKE 15498, 131
3020 FOR X=15499 TO 15508
3030 POKE X, 32
3040 NEXT X
3050 POKE 15509, 136
3060 POKE 15510, 140
3070 POKE 15511, 140
3080 POKE 15512, 191
3090 POKE 15513, 191
3100 POKE 15514, 140
3110 POKE 15515, 140
3120 POKE 15516, 132
3130 FOR X=15517 TO 15524
3140 POKE X, 32
3150 NEXT X
3160 POKE 15525, 154
3170 POKE 15526, 32
3172 POKE 15527, 32
3175 POKE 15528, 32
3180 POKE 15529, 130
3190 POKE 15530, 140
3200 POKE 15531, 164
3210 FOR X=15532 TO 15537
3220 POKE X, 176
3230 NEXT X
3240 FOR X=15538 TO 15560
3250 POKE X, 32
3260 NEXT X
3270 POKE 15561, 191
3280 FOR X=15562 TO 15574
3290 POKE X, 32
3300 NEXT X
3305 POKE 15574, 160
3310 POKE 15575, 134
3320 POKE 15576, 32
3330 POKE 15577, 32
3340 POKE 15578, 137
3350 POKE 15579,  144
3360 FOR X=15580 TO 15588
3370 POKE X,  32
3380 NEXT X
3390 POKE 15589, 149
3400 FOR X=15590 TO 15595
3410 POKE X, 32
3420 NEXT X
3430 POKE 15596, 168
3440 POKE 15597, 148
3450 FOR X=15598 TO 15601
3460 POKE  X, 32
3470 NEXT X
3480 POKE 15602, 137
3490 POKE 15603, 176
3500 FOR X=15604 TO 15624
3510 POKE X, 32
3520 NEXT X
3530 POKE 15625, 191
3540 FOR X=15626 TO 15652
3550 POKE X, 32
3560 NEXT X
3570 POKE 15653, 130
3580 POKE 15654, 164
```

Program continues

✓See List of Advertisers on page 354

```
3590 FOR X=15655 TO 15667
3600 POKE X, 32
3610 NEXT X
3620 POKE 15668, 131
3630 POKE 15669, 140
3640 POKE 15670, 140
3650 POKE 15671, 140
3660 POKE 15672, 176
3670 FOR X=15673 TO 15686
3680 POKE X, 32
3690 NEXT X
3700 FOR X=15687 TO 15691
3710 POKE X, 191
3720 NEXT X
3730 FOR X=15692 TO 15704
3740 POKE X, 32
3750 NEXT X
3760 POKE 15705, 170
3770 POKE 15706, 131
3780 POKE 15707, 140
3790 POKE 15708, 140
3800 POKE 15709, 176
3810 POKE 15710, 176
3820 FOR X=15711 TO 15718
3830 POKE X, 32
3840 NEXT X
3850 POKE 15719, 165
3860 FOR X=15720 TO 15735
3870 POKE X, 32
3880 NEXT X
3890 POKE 15736, 176
3900 POKE 15737, 133
3910 FOR X=15738 TO 15768
3920 POKE X, 32
3930 NEXT X
3940 POKE 15769, 162
3950 POKE 15770, 153
3960 POKE 15771, 132
3970 POKE 15772, 32
3973 POKE 15773, 32
3976 POKE 15774, 32
3980 POKE 15775, 179
3990 POKE 15776, 179
4000 POKE 15777, 140
4010 POKE 15778, 140
4020 FOR X=15779 TO 15783
4030 POKE X, 32
4040 NEXT X
4050 POKE 15784, 137
4060 POKE 15785, 149
4070 FOR  X=15786 TO 15795
4080 POKE X,  32
4090 NEXT X
4100 POKE 15796, 160
4110 POKE 15797, 135
4120 POKE 15798, 131
4130 POKE 15799, 131
4140 FOR X=15800 TO 15832
4150 POKE X, 32
4160 NEXT X
4170 POKE 15833, 170
4180 POKE 15834, 176
4190 POKE 15835, 140
4200 POKE 15836, 140
4210 POKE 15837, 131
4220 POKE 15838, 131
4230 FOR X=15839 TO 15848
4240 POKE X, 32
4250 NEXT X
4260 POKE 15849, 130
4270 POKE 15850, 141
4280 FOR X=15851 TO 15859
4290 POKE X, 140
4300 NEXT X
4310 POKE 15860, 129
4320 FOR X=15861 TO 15876
4330 POKE X, 32
4340 NEXT X
4350 POKE 15877, 172
4360 POKE 15878, 140
4363 POKE 15879, 140
4366 POKE 15880, 140
4370 POKE 15881, 152
4380 POKE 15882, 140
4390 POKE 15883, 131
4400 POKE 15884, 131
4410 POKE 15885, 131
4420 POKE 15886, 141
4430 POKE 15887, 140
4440 POKE 15888, 180
4450 POKE 15889, 176
4460 FOR X=15890 TO 15941
4470 POKE X, 32
4480 NEXT X
4490 POKE 15942, 131
4500 POKE 15943, 179
4510 POKE 15944, 143
4520 POKE 15945, 32
4530 POKE 15946, 140
4540 FOR X=15947  TO 15953
4550 POKE X, 32
4560 NEXT X
4570 POKE 15954, 169
4580 FOR X=15955 TO 15979
4590 POKE X, 32
4600 NEXT X
4610 POKE 15980, 160
4620 POKE 15981, 144
4630 FOR X=15982 TO 15990
4640 POKE X, 32
4650 NEXT X
4660 POKE 15991, 176
4670 FOR X=15992 TO 16005
4680 POKE X, 32
4690 NEXT X
4700 POKE 16006, 138
4710 POKE 16007, 140
4720 POKE 16008, 131
4730 POKE 16009, 140
4740 POKE 16010, 176
4750 FOR X=16011 TO 16016
4760 POKE X, 32
4770 NEXT X
4780 POKE 16017, 176
4790 POKE 16018, 134
4800 FOR X=16019 TO 16042
4810 POKE X, 32
4820 NEXT X
4830 POKE 16043, 174
4840 POKE 16044, 157
4850 FOR X=16045 TO 16053
4860 POKE X, 140
4870 NEXT X
4880 POKE 16054, 179
4890 FOR X=16055 TO 16073
4900 POKE X, 32
4910 NEXT X
4920 POKE 16074, 152
4930 POKE 16075, 129
4940 POKE 16076, 32
4950 POKE 16077, 32
4960 POKE 16078, 176
4970 POKE 16079, 140
4980 POKE 16080, 131
4990 FOR X=16081 TO 16107
5000 POKE X, 32
5010 NEXT X
5020 POKE 16108, 130
5030 POKE 16109, 129
5040 FOR X=16110 TO 16118
5050 POKE X, 32
5060 NEXT X
5070 POKE 16119, 131
5080 FOR X=16120 TO 16135
5090 POKE X, 32
5100 NEXT X
5110 POKE 16136, 160
5120 POKE 16137, 134
5130 FOR X=16138 TO 16142
5140 POKE X, 32
5150 NEXT X
5160 FOR X=16143 TO 16146
5170 POKE X, 131
5180 NEXT X
5190 POKE 16147, 140
5200 POKE 16148, 140
5210 POKE 16149, 140
5220 POKE 16150, 131
5230 POKE 16151, 137
5240 POKE 16152, 156
5250 POKE 16153, 129
5260 FOR X=16154 TO 16199
5270 POKE X, 32
5280 NEXT X
5290 POKE 16200, 137
5300 POKE 16201, 176
5310 FOR X=16202 TO 16213
5320 POKE X, 32
5330 NEXT X
5340 POKE 16214, 160
5350 POKE 16215, 134
5360 FOR  X=16216 TO 16264
5370 POKE X, 32
5380 NEXT X
5390 POKE 16265, 130
5400 POKE 16266, 131
5410 POKE 16267, 140
5420 POKE 16268, 140
5430 POKE 16269, 140
5440 FOR X=16270 TO 16273
5450 POKE X, 176
5460 NEXT X
5470 POKE 16274, 140
5480 POKE 16275, 140
5490 POKE 16276, 140
5500 POKE 16277, 131
5510 POKE 16278, 129
5520 FOR X=16279 TO 16380
5530 POKE X, 32
5540 NEXT X
5550 PRINT @927, "THATS IT!"
5560 END
```

✓See List of Advertisers on page 354

Animate your TRS-80 by packing those strings.

Picture This

Dan Keen
Dave Dischert
RD 1 Box 432
State Highway
Cape May Court House, NJ 08210

We have this conception of a man who bought the first TRS-80 that rolled off the production line, took it home, spent several minutes with it, stumbled onto something that isn't in the owner's manual, and proceeded to write a game called "Android Nim" that soon became a well known standard in the world of microcomputing.

We're sure things didn't come that easy to Leo, but he sure makes graphics look easy. He proved the TRS-80 is capable of a lot more than Radio Shack gives it credit for. Smooth, rapidly executing graphics are possible using Basic.

There have been a few articles scratching the surface of this technique, but only enough to whet our appetites. We will follow an animated scene from start to finish using string packing procedures. All you need is a Level II machine, with or without Disk Basic, with any amount of memory.

```
10 A$="A"
20 K=VARPTR(A$)
30 AD=PEEK(K+2)*256+PEEK(K+1)
40 POKEAD,191
50 PRINTA$
```

Program Listing 1.

What Is Packing?

This procedure stores ASCII characters in the space where a string normally goes. The 256 ASCII characters are actually machine code which represents letters, numbers, graphics and instructions. The Basic interpreter section of the computer is not used—one reason why printing with string packing is so swift.

Packing machine code makes graphics faster, conserves memory space, eliminates the need for USR functions to access the code (which gets tricky in Disk Basic since you must know an entry address to use it), eliminates the need to answer the memory size question (programs are not limited to any size K), and makes it easy to CLOAD from tape since the program is in Basic. (Machine-code tapes are notorious for their difficult loading.)

Program Listing 1 is the bare minimum needed to set up a simple packed string. The number POKEd in line 40 can be anything from zero to 255. The graphic code for lighting six blocks is 191. A$ can equal anything; only the length is important. If you change line 40 to POKE AD,191 :POKE AD+,191, A$ must have two characters, A$ = "AB".

Run Listing 1. Printing A$ did not display an A, but instead acted as if the computer was instructed to print CHR$(191). Now list the program and look at line 10. It changed! It has been packed and really is no longer a string. It now reads: 10 A$ = "USING". The machine code number 191 is there though only the Basic word which represents that number is visible.

Since A$ doesn't really equal the word using, just typing in "using" won't accomplish the same thing. Packing takes up no string space at all—the code is

```
10 CLS
200 F1$="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
202 K=VARPTR(F1$):AD=PEEK(K+2)*256+PEEK(K+1)
204 DATA176,176,176,176,176,128,26,24,24,24,24,24,24,24,24,184,191,179,191,191,191,191,189,184,149,128,26,24,24,24,24,24,24,24,24,24
,24,24,130,143,185,191,191,191,191,135,130,133,128
206 FORP=0TO48:READX:POKEAD+P,X:NEXTP
210 F2$="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
212 K=VARPTR(F2$):AD=PEEK(K+2)*256+PEEK(K+1)
214 DATA176,176,176,176,176,128,26,24,24,24,24,24,24,24,24,184,191,179,191,191,191,191,189,184,134,128,26,24,24,24,24,24,24,24,24,24
,24,24,130,143,185,191,191,191,191,135,130,137,128
216 FORP=0TO48:READX:POKEAD+P,X:NEXTP
220 F3$="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
222 K=VARPTR(F3$):AD=PEEK(K+2)*256+PEEK(K+1)
224 DATA176,176,176,176,176,128,26,24,24,24,24,24,24,24,24,184,191,179,191,191,191,191,189,176,140,128,26,24,24,24,24,24,24,24,24,24
,24,24,130,143,185,191,191,191,191,135,128,131,128
226 FORP=0TO48:READX:POKEAD+P,X:NEXTP
230 F4$="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
232 K=VARPTR(F4$):AD=PEEK(K+2)*256+PEEK(K+1)
234 DATA176,176,176,176,176,128,26,24,24,24,24,24,24,24,24,184,191,191,191,191,191,191,189,184,149,128,26,24,24,24,24,24,24,24,24,24
,24,24,130,143,185,191,191,191,191,135,130,133,128
236 FORP=0TO48:READX:POKEAD+P,X:NEXTP
1000 CLS:LO=502:SW=0
1010 PRINT@LO,F1$;:GOSUB1100
1020 PRINT@LO,F2$;:GOSUB1100
1030 PRINT@LO,F3$;:GOSUB1100
1035 IFLO<452THEN2000
1040 SW=SW+1:IFSW<2THEN1010ELSEPRINT@LO,F4$;:GOSUB1100:SW=0:GOTO1020
1100 FORQ=1TO15:NEXTQ:LO=LO-1:RETURN
```

Program Listing 2.

POKEd right into the program line. So, although there are five letters in A$, it does not necessarily take up five bytes. You can add: 5 CLEAR Ø, and the program will still run. You can delete lines 20, 30, and 40, save the leftover two-line program on tape or disk, and always get a perfect run when reloaded. Once a string is packed, you no longer need the POKE statements.

Let's create a cartoon. Keep in mind these hints: Always be sure to put in the characters that the string is set equal to quotation marks. It won't matter in Listing 1, but if there were more POKEs, leaving off the end quote would result in destruction of the next few lines of the program. Also, incorporate blanks or spaces (ASCII codes 32 or 128—both do the same) into your animated subject to erase behind it when moving it across the screen. CLS is too slow and choppy to get smooth movement.

```
10 CLS
200 F1$="DEFDEFDEFDEFDEFENDCLEARUSINGCONTUSINGUSINGUSINGUSINGTOCLEAELSEENDRESETIFCLOADUSINGUSINGUSINGUSINGNEXTRESETCMDEND"
202 K=VARPTR(F1$):AD=PEEK(K+2)*256+PEEK(K+1)
204 DATA176,176,176,176,176,128,26,24,24,24,24,24,24,24,24,184,191,179,191,191,191,191,189,184,149,128,26,24,24,24,24,24,24,24,24,24
,24,24,130,143,185,191,191,191,191,135,130,133,128
206 FORP=0TO48:READX:POKEAD+P,X:NEXTP
210 F2$="DEFDEFDEFDEFDEFENDCLEARUSINGCONTUSINGUSINGUSINGUSINGTOCLEARRANDOMENDRESETIFCLOADUSINGUSINGUSINGUSINGNEXTRESETINPUTEND"
212 K=VARPTR(F2$):AD=PEEK(K+2)*256+PEEK(K+1)
214 DATA176,176,176,176,176,128,26,24,24,24,24,24,24,24,24,184,191,179,191,191,191,191,189,184,134,128,26,24,24,24,24,24,24,24,24,24
,24,24,130,143,185,191,191,191,191,135,130,137,128
216 FORP=0TO48:READX:POKEAD+P,X:NEXTP
220 F3$="DEFDEFDEFDEFDEFENDCLEARUSINGCONTUSINGUSINGUSINGUSINGTODEFLETENDRESETIFCLOADUSINGUSINGUSINGUSINGNEXTENDSETEND"
222 K=VARPTR(F3$):AD=PEEK(K+2)*256+PEEK(K+1)
224 DATA176,176,176,176,176,128,26,24,24,24,24,24,24,24,24,184,191,179,191,191,191,191,189,176,140,128,26,24,24,24,24,24,24,24,24,24
,24,24,130,143,185,191,191,191,191,135,128,131,128
226 FORP=0TO48:READX:POKEAD+P,X:NEXTP
230 F4$="DEFDEFDEFDEFDEFENDCLEARUSINGUSINGUSINGUSINGUSINGUSINGTOCLEAELSEENDRESETIFCLOADUSINGUSINGUSINGUSINGNEXTRESETCMDEND"
232 K=VARPTR(F4$):AD=PEEK(K+2)*256+PEEK(K+1)
234 DATA176,176,176,176,176,128,26,24,24,24,24,24,24,24,24,184,191,191,191,191,191,191,189,184,149,128,26,24,24,24,24,24,24,24,24,24
,24,24,130,143,185,191,191,191,191,135,130,133,128
236 FORP=0TO48:READX:POKEAD+P,X:NEXTP
1000 CLS:LO=502:SW=0
1010 PRINT@LO,F1$;:GOSUB1100
1020 PRINT@LO,F2$;:GOSUB1100
1030 PRINT@LO,F3$;:GOSUB1100
1035 IFLO<452THEN2000
1040 SW=SW+1:IFSW<2THEN1010ELSEPRINT@LO,F4$;:GOSUB1100:SW=0:GOTO1020
1100 FORQ=1TO15:NEXTQ:LO=LO-1:RETURN
```

Program Listing 3.

We will make a fish swim across the screen from right to left. As he swims, his tail will wiggle and he will occasionally blink.

To accomplish this, we will create a series of compressed strings, each a different variation of the fish, then print them in sequence, printing and incrementing as we go along. We avoid the CLS command since it causes the action to look rough. Instead, a space is added to the tail (actually two spaces, one underneath the other since the fish is two lines tall) so that the trailing blanks are printed along with the fish. You don't see the blanks, but their effect is to erase any characters left on the screen from the last printing. This trick brings us closer to real cartoon action.

"We will follow an animated scene from start to finish using string packing procedures."

Note the use of ASCII codes 26 and 24 to downshift and backspace. Any size object can be stored in one string using these codes.

In our animated fish scene, we use four variations. Three have different shaped tails, to give the illusion of swimming, and the fourth is the same as the first except that the eye is closed to simulate a blink. The variable SW is used as a switch to select between the blinking and unblinking variations.

The four fish are set up in lines 200 through 236. It takes a few seconds for the machine to run these lines. Lines 1000–1100 create the action on the screen. Program Listing 2 is the program as you must type it in.

Program Listing 3 is the program after you run it once. As stated earlier, once the program is run, you can delete lines 202, 204, 206, and so forth (all the POKE, VARPTR, and data statements). The program can then be saved or CSAVEd and will run correctly. No one will ever know your original program was twice as long.

It is possible to refine the fish movement further. Rather than incrementing a character space at a time, several variations of the fish could advance the fish one graphic block. The fish would move forward twice for each PRINT @ position. Instead of 40 usable frames there would be 80.

We hope this takes some of the mystery out of the string packing process. ■

TUNE-UP YOUR TRS-80

Ever wanted
to do things to your TRS-80
that Radio Shack said couldn't be done?
How about reverse video, high-resolution graphics, a high-speed clock, and audible keystrokes?

Not enough? How about turning an 8-track into a mass storage device, making music, controlling a synthesiser, individual reverse characters, and a real-time clock?

If the thought of using a screwdriver gives you the shivers then you can turn to the software section. Learn how to make BASIC programs auto-execute, reset memory size, pack program lines with machine code, and generate sound effects.

The Custom TRS-80 and Other Mysteries is packed with page after page of practical information, and tested software. Get a copy and turn your TRS-80 into a supercomputer! Available from your local *IJG* dealer for only $29.95.

IJG

1260 West Foothill Blvd.,
Upland, California 91786
(714) 946-5805

TRS-80 is a trademark of Tandy

MAKE BASIC PERFORM LIKE A CHAMPION

BASIC
is not nearly
as slow as most programmers think

BASIC Faster and Better shows you how to supercharge your BASIC, with 300 pages of fast, functions and subroutines.

You won't find any trivial poorly designed 'check-book-balancing' programs in this book – it's packed with *useful* programs.

Tutorial for the beginner, instructive for the advanced, and invaluable for the professional, this book doesn't just talk . . . it shows how! All routines are also available on disk, so that you can save hours of keyboarding and debugging.

The #1 disk *BFBDEM* contains all the demonstration programs, and #2 *BFBLIB* has all the library functions.

BASIC Faster & Better is $29.95, and the two program disks are $19.95 each.

Get the book and/or disks direct from *IJG*, or your local *IJG* dealer.

IJG ✓37

1260 West Foothill Blvd.,
Upland, California 91786
(714) 946-5805

TRS-80 is a trademark of Tandy

Cassette loading may seem square to you but your computer likes it that way.

The Wave Shaper

R.B. Shreve
2842 Winthrop Rd.
Shaker Heights, OH 44120

Few would argue the desirability of a backup file for irreplacable data, or backup copies of programs representing the investment of many hours of time or substantial sums of money. Cassettes are less expensive backup files than floppies. There is widespread misunderstanding, however, of what makes the volume control setting on a CTR so critical when loading files and disagreement on the easiest way to ensure good loads.

In "No More TRS-80 Cassette Woes" by Paul Goetz and David Miller (*73 Magazine*, September, 1979, p. 96), the authors point out that it is not the amplitude of the pulses put out by the cassette recorder that cause loading problems, but their duration. I have verified this by monitoring the output of several recorders with an oscilloscope under conditions producing good and bad loads. The wave shapes differ from Miller and Goetz' illustrations, probably due to different audio circuit characteristics, but the results were consistent with their conclusions.

Fig. 1 shows the output waveform of a CTR-80 recorder when loading a tape made on the same recorder with my TRS-80. Fig. 1a is the wave with a volume control setting between four and six, which gave a satisfactory load. Fig. 1b is with a setting above eight, which did not. Obviously, if you want to add an oscilloscope to your peripheral equipment, you could get a good load every time. You can buy a lot of disks for the price of a good 'scope, however.

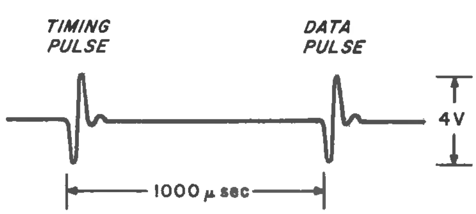


Fig. 1a. Satisfactory Load Setting

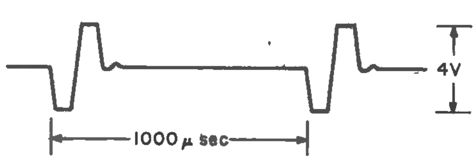


Fig. 1b. Unsatisfactory Load Setting

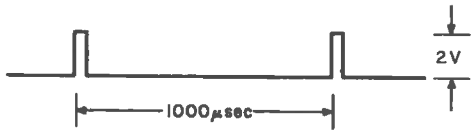


Fig. 1c. Wave-shaping Device Output

Fig. 1. Output Wave forms of CTR-80 Recorder

Why go to the trouble of installing a monitor? Goetz and Miller have the right idea; modify your recorder's output before it gets to the computer.

Digital equipment doesn't like sloping wave shapes that rise and fall gradually (measured in terms of time as a computer sees it), but thrives on short pulses with sharp edges and square corners. Let's give our TRS-80 what it wants. The E-Z Loader described by Goetz and Miller does it very well, but it is more complex than necessary. My device converts the CTR-80 output to the waveform shown in Fig. 1c no matter whether it looks like Fig. 1a or Fig. 1b coming out of the recorder, and it is easier to assemble and less expensive.

An Inexpensive Wave-shaping Device

Here's the parts list, with prices:

1-SN74121 Monostable Multivibrator IC	$ .35
1-2N2222 NPN Transistor	.25
2-Mylar capacitors: 0.1 and .01 mfd (20¢ each)	.40
5-one fourth watt 5% Resis-	

tors: two at 1K, 1 each 2.2K, 4.7K, and 10K ohms: (5¢ each)	.25
1-Silicon Diode, such as 1N914 or 1N4148	.10
Total Cost:	$ 1.35

The unit requires a five-volt power source, and draws pulses that probably exceed 50ma. when connected to the TRS-80. If you want to install it inside the cassette recorder and power it from the recorder, you will need a voltage regulator. The one shown in the circuit diagram uses the following parts:

1- LM340T-5 Regulator	$ 1.25
1- 1000mfd 16V. Electrolytic Capacitor	.54
1- 200mfd 16V. Electrolytic Capacitor	.23
	$ 2.02

The entire unit, including regulator, can be assembled on a one-inch by three-inch piece of perforated board, and located in the recorder battery compartment. I added a miniature plug that fits the recorder ear jack, and a matching jack to take the plug on the wire to the computer. This avoids rewiring the existing jack, which is hard to get at, and permits the recorder to operate normally if I want to use it for something besides computer tapes.

The circuit is shown in Fig. 2. Layout is not critical; arrange the parts to fit your mounting board and the space. You can use an IC socket if you wish; it is easier than making soldered connections to the IC pins. You could make a PC board if you want to be fancy.

Operation

The secret is the SN74121. This is an IC that might have been designed just for this purpose. It has three inputs, two of which (pins 3 and 4) are designed to be triggered by square waves. The third, pin 5, can be triggered by a slow-rising wave shape like our cassette output. Whichever input triggers it, it will output a single square pulse at pin 6. Pulse duration is independent of anything that happens after it is triggered, and is determined by the timing resistor and capacitor connected to pins 10 and 11.

According to the manufac-

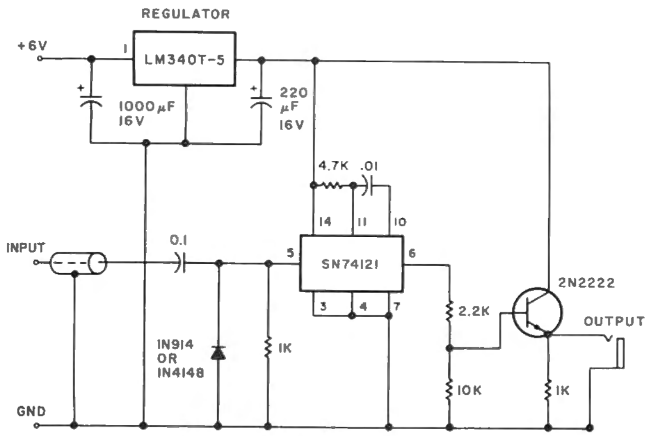


Fig. 2. Circuit

turer's data sheet for an SN74121, the values shown in Fig. 2 should produce a pulse of approximately 32 microseconds duration, whether the triggering wave looks like Fig. 1a or 1b. In other words, the setting of the recorder volume control is disregarded as long as it is high enough to trigger the IC.

Actual pulse duration will vary from this figure, depending on how close the components you use come to the design values. This is no problem; the tape should load satisfactorily with any pulse up to 150 microseconds. The voltage divider and transistor in the output circuit limit the pulse amplitude to approximately 2V while providing enough current for the low impedance load of the TRS-80 input. The resistor from pin 5 to ground returns the device input to ground potential after it is triggered and the diode protects it against any negative voltage that might occur by accident.

Installation in the CTR-80

Installation is easy in the battery compartment of your recorder:

- Remove the back of the recorder, by removing the three Phillips-head screws, one in the bottom of the battery compartment, and one at each corner opposite the battery.

- Drill a hole in the side of the back cover next to the battery compartment, of a size to mount the new output jack, and on the same side of the recorder as the other inputs and outputs. Drill another small hole alongside the first for the wire that will connect to the input plug.

> *"Goetz and Miller have the right idea; modify your recorder's output before it gets to the computer."*

- Mount the jack. I made it easier by cutting away some of the thin plastic wall of the battery compartment on which the contact springs are mounted, using a small pair of diagonal cutting pliers. If you do this, first trim off or tape up any loose battery wiring.

- Locate the points on the recorder circuit board at which the +6V and ground connections will be made. Fig. 3 shows part of the foil layout of the CTR-80 circuit board. The Phillips-head screw on the left will help you identify the ground bus. The two brown wires and one red wire at the top will help locate the +6V point—the small pad to which the red wire and one brown wire connect. Run leads from these points to the battery compartment and connect them to the wave shaper. Connect its output to the output jack. Run the wire for the input plug through its hole and connect it.

- While you have the back off the recorder, you may want to solder another silicon diode such as a rectifier diode or 1N914 across the points where the shielded wire to the drive motor is connected to the circuit board. This has nothing to do with the installation of the wave-shaping device, but is good protection against voltage spikes generated when the motor is stopped. These spikes can put a glitch in the middle of a tape if the computer stops the recorder, as it may when saving data under program control.

- Reinstall the back of the recorder.

- Put a small piece of sponge rubber or other padding in the bottom of the battery compartment, rest the wave shaper on it, and reinstall the battery compartment cover. The padding on the cover will hold the device in place.

That should do it. Connect the recorder to your TRS-80 and try loading a cassette with the level control set at six or higher. It should load on the first try. Note that this device will not load tapes saved on TRS-80s with modified speeds. No one in the Cleveland area has had any problems in either construction or operation to my knowledge. If you do, drop me a line.■

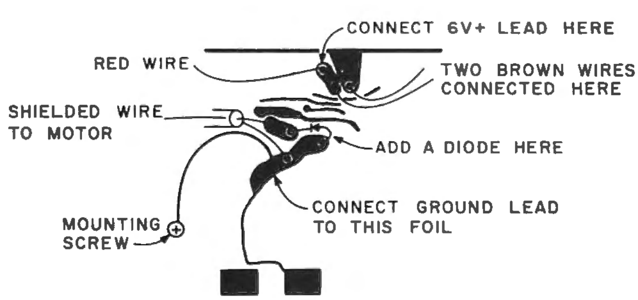


Fig. 3. Foil Layout—CTR-80 Circuit Board

Why don't you do it in machine language?

High Speed Sorts

Richard R. Robson
13 Stilwell
Fort Leavenworth, KS 66027

It all started when my friend Phil told me the sort in his mailing list program took forever to complete. He hated to add any records to his files because the Add program automatically called up the Sort program, resulting in endless hours of a churning and gurgling Basic sort.

In any sort, swapping string variables is a fact of life. In a TRS-80 Level II Basic sort, extensive string swapping is nothing but trouble.

Look at the simple version of a sort of the array TG$( ) in Program Listing 1. Each time a TG$(n) or SVE$ is assigned a new value, Level II Basic searches for the next available chunk of string storage space to store the new value. The previous value of the string variable remains in its original location in memory, and the pointer for TG$(n) is updated to point to the newly found chunk of storage space containing the latest value. This can't go on forever; this spendthrift method soon leads to an out-of-string-space condition.

When this occurs, Level II calls the garbage collector routine. This routine causes all normal processing to stop while it goes through string space, collecting the space occupied by superceded values of string variables. Having done this, it then reorganizes the string space so all the current values of string variables are neatly lined up from the top of available memory downward. At the same time, it updates all the internal pointers to these values. During all this, the resident program watches from the sidelines.

In a sort of any size, the computer could call the garbage collector several times during each pass through an array, but there is a better way. Machine code is the answer.

Sorting Out Sorts

The usual approach to sorting involves moving or switching blocks of data. The approach taken here leaves the string undisturbed in memory. But how do you resequence the data if you never touch it? Simple: You don't rearrange the data in the TRS-80's memory; you merely rearrange the internal pointers to the data.

When you ask Level II (or Disk Basic) for VARPTR(TG$(1)), you get an answer which represents the memory location where TG$(1) is stored. If that memory location is abbreviated as M, then M – 3 (read as "M minus three") contains the length of TG$(0), and M + 3 contains the length of TG$(2)

M + 1 and M + 2 point, respectively, to the LSB (least significant bit) and MSB (most significant bit) of the memory address where the actual value of TG$(1) is stored. Similarly, M – 2 and M – 1 contain the LSB and MSB of the location where data is stored for TG$(0). Table 1 illustrates this concept and indicates that, given VARPTR (TG$(1$)), we can tell a machine code program all it needs to know about the TG$( ) array.

Let's say that while our sort is going through the TG$( ) array, it is looking at TG$(1) and TG$(2). Table 1 tells us that their respective lengths are at M and M + 3 and that their data can be found at the addresses whose LSB and MSB are found at M + 1,

Given: M = VARPTR (TG$(1))
Then:

Memory Location	Contents
M−11	"3" signifying String
Variable	
M−10	"G" (of "TG")
M−9	"T" (" ")
M−8	who knows?
M−7 & M-6	who knows
M−5	LSB of $ of elements in TG$ ( )
M−4	MSB of $ of elements in TG$ ( )
M−3	LEN(TG$(0))
M−2	LSB of address of TG$(0) data
M−1	MSB of address of TG$(0) data
M	LEN(TG$(1))
M+1	LSB of address of TG$(1) data
M+2	MSB of address of TG$(1) data
M+3	LEN(TG$(2))
M+4	LSB of address of TG$(2) data
M+	MSB of address of TG$(2) data
.	.
.	.
.	.
M+3*n −3	LEN(TG$(n))
M+3*n −2	LSB of address of TG$(n) data
M+3*n −1	MSB of address of TG$(n) data

Table 1 VARPTR Structure

M+2 and M+4,M+5. With those addresses and lengths, we can compare each byte of TG$(1) with the corresponding byte of TG$(2) and continue to do so until we hit a not equal condition, the end of the shorter of the two strings, or the simultaneous end where the lengths are equal.

There are three possible outcomes. First, the unequal case: Assuming that our desired sequence is ascending (low values to high values), if TG$(1) is greater in value then TG$(2), we want to switch the two strings to establish an ascending sequence. If this is not the case, leave well enough alone; they are already in ascending sequence.

The second case occurs when both strings are the same length and are exactly equal in value. Don't worry; ascending sequence exists whenever TG$(1) is less than or equal to TG$(2).

The third case occurs when we compare the two strings byte for byte, and continue to get an equal condition. Suddenly we reach the end of the shorter of the two strings. The shorter of the two strings is really the less-than string and it should be switched or not, determined as was the first case.

A sort in machine code can buzz along at a pretty good clip, but it is still quite cumbersome when it has to move large, variable length blocks of data.

Look closely at Table 1. Why not switch the pointers instead of switching the data to which they point? That way, you move only three bytes at a time and you don't have to emulate the garbage collector to accommodate switching variable length blocks. More importantly, by rearranging the pointer for TG$( ), you make things much nicer for yourself when you return to the Basic program once the sort is done. On return, TG$(1) will contain the data with the lowest values and TG$(n) will have the string with the highest values. To list those strings in ascending sequence, dump them out one at a time with a simple For...Next loop. If you want descending instead of ascending sequence, dump the array with the following code:

```
FOR N = (highest subscript) TO 1 STEP − 1
PRINT TG$(N)
NEXT N
```

A Calling Program

Program Listing 2 is a sample calling program written for any 16K to 48K, disk or tape system. Lines 20 through 110 contain the decimal values of the machine-code sort program's instructions. Lines 130 and 140 determine what size system you have and compute a starting point for the sort program. In line 150, a dummy string is set up which in effect reserves 255 bytes at the top of memory into which the sort program is POKEd by line 160. This string (ZZ$) must never again be referred to in the Basic program to protect it from the garbage collector. Line 160 computes a hash total (T) of the decimal values. If you keyed in lines 20-110 correctly, the total of all 190 numbers would be 18,217. Lines 180 through 200 check this and, if you made a mistake, allow you to check your work and make necessary corrections.

Lines 230-250 determine whether your system is a tape or a disk system and set up the proper USR interface. Once you are sure your data lines are correct, you can delete lines 170-220.

The rest of the program sets up a test array, prints it in its original sequence, sorts it using the call at line 390, and then prints the sorted results in ascending and descending sequence. Lines 510-550 let you enter 18 values and sort them. The actual sort takes place quickly with little delay between the dump of the unsorted array and the dump after the sort (less than one second).

The key to this speed is the machine-code sort routine. Program Listing 3 is an Assembly listing of this relocatable program. The first part receives the value of VARPTR(TG$(1)) passed by the calling Basic program and saves it in Start. Five is then subtracted to get the M-5 pointer to the number of array elements. The element count is reduced by two (once because we don't use TG$(0) and once because the bubble sort stops short of the last element—(TG$(n)). The adjusted element count is then stored in ELEMS. The last part of the paragraph stores LEN(TG$(0)) in TGNOTG for later reference.

Once the housekeeping is done, the program gets down to business in Sort. First it sets up the IX index register to contain VARPTR(TG$(1)). It also loads IY with the adjusted element count and initializes the switch flag to zero.

In DOTAG, the program loads HL with the address of byte 1 of the second of the two strings and loads DE with byte 1 of the first of the two strings to be compared. The B and C registers are loaded with the lengths of the two strings. To illustrate this, assume we are comparing TG$(1) and TG$(2). The C register would contain LEN(TG$(1)) and DE would point to byte 1 of TG$(1). Similarly, B contains LEN(TG$(2)) and HL points to byte 1 of TG$(2).

In EXAMIN, we compare the two strings byte for byte. We come back to EXAMIN again

```
10 '              FIGURE 1 - BASIC SORT
20 '
30 CLEAR 1000
40 DATA SMITH,BAKER,ABLE,JONES,M,HOTEL,KILO,GEORGE,ZULU,CABIN
50 DATA DOG,FOX-TERRIER,FOXTERRIER,SMITHA,WERE,MHJKI,XYZ,ABC
60 T=18: DIM TG$(T)
70 LPRINT,"FIGURE 1 - PROGRAM OUTPUT & TIMING USING BASIC STRING
 SORT":LPRINT"
  ":LPRINT,"BEFORE SORTING",TIME$
80 FOR N = 1 TO T: READ TG$(N): LPRINT TG$(N): NEXTN
90 '
100 LPRINT,"BEGIN SORTING",TIME$
110 GOSUB160: REM PERFORM SORT IN BASIC
120 LPRINT,"END SORTING",TIME$
130 FOR N = 1 TO T:LPRINT TG$(N): NEXT N: REM PRINT SORTED ARRAY

140 END
150 '              SORT ROUTINE
160 '              (IN BASIC)
170 FLAG = 0
180 FOR N = 1 TO T-1
190 IF TG$(N) > TG$(N+1) THEN GOSUB 230: REM SWITCH ROUTINE
200 NEXT N
210 IF FLAG <> 0 THEN GOTO 170: REM LOOP TIL NO SWITCHES MADE
220 RETURN: REM FINALLY PASSED THRU ARRAY WITH NO SWITCHES
230 '              ROUTINE TO SWITCH VALUES
240 FLAG = 1:REM SET FLAG TO INDICATE AT LEAST 1 SWITCH MADE
250 SVE$ = TG$(N):       REM SAVE TG$(N)
260 TG$(N) = TG$(N+1):   REM MOVE N+1 TO N
270 TG$(N+1) = SVE$:     REM MOVE N TO N+1 VIA SVE$
280                      REM NOTE STRING VARIABLES ASSIGNED NEW V
ALUES
290 RETURN
```

Program Listing 1

and again until we arrive at one of the three outcomes discussed earlier. If the two strings are already in the proper sequence, the program falls into Next, where it increments the IX register to point to the next VARPTR and checks to see if we have reached the next to last VARPTR. If not, we loop back to DOTAG; otherwise, the program checks to see if we made any switches along the way, and if so, goes back to Sort.

The program returns to Basic only after it has made a complete pass through the array without performing a single switch (flag = 0).

The next busiest section of the program is Switch. Here the program physically switches the length/LSB/MSB triads when the comparisons in EXAMIN dictate a rearrangement of the sequence. Upon exit from this section Flag is set equal to one, indicating a switch has been made. The data (strings) are never switched—only the pointers are switched, requiring movement of two sets of three bytes each.

Tag Sorts

One often wants to sort a large file, for instance, a mailing list. The sort is usually keyed on one small field out of a record that could exceed 100 characters in length. Phil's mailing list, for example, had a couple hundred records, each 85 characters long; but his sort key was only six bytes long.

If Phil tried to read all those records into memory to sort them on the six byte keys, it would be a tight fit and a very long sort. Instead, he could easily fit the six byte keys into memory and sort on them. Once he sorted the array of keys, they would be in sequence.

As he reads in the records from his disks, he uses a record number (RN) uniquely associated with each record. This record number can be converted into two ASCII (American Standard Code for Information Interchange) bytes by the following code:

```
MSB = INT(RN/256)
LSB = RN - (256*MSB)
KEY$ = CHR$(LSB) + CHR$(MSB)
In Disk BASIC: KEY$ = MKI$(RN)
```

While saving the desired field (for example, FLD$) in the array, Phil saves the record key as well by concatenating the six-byte field with the two-byte tag:

```
TG$(N) = FLD$ + KEY$
```

After the TG$( ) array is sorted, each element in the array will be in ascending sequence and will carry the tag with it to its parent record out on the disk. The array element can then be broken out back into the field component and the tag component:

```
FLD$ = LEFT$(TG$(N),LEN(TG$(N)) - 2)
KEY$ = RIGHT$(TG$(N),2)
RN = ASC(LEFT$(KEY$,1))
RN = RN + 256*ASC(RIGHT$(KEY$,1))
In Disk BASIC: RN = CVI(KEY$)
```

The tag sort technique can also be used in another fashion. Assume an in-memory information system used the following arrays for name (NM$( )), phone (PH$( )), and street (S$( )), and we wanted to display the data in phone-number sequence. If we did a regular sort on PH$( ), the phone numbers would be in sequence but the rest of the data would not.

For example, before the sort, NM$(1) = ADAMS, PH$(1) = 999 – 9999, NM$(n) = ZULU, and PH$(n) = 111 – 111. After the sort, PH$(1) would be 111 – 111, PH$(n) would be 999 – 9999; NM$(1) would remain ADAMS and NM$(n), ZULU. But now ADAMS has ZULU's phone number and vice versa. The tag sort would fix that. Examine the following code:

```
FOR I = 1 TO n
MSB = INT(I/256):LSB = I - 256*MSB
TG$(I) = PH$(I) + CHR$(LSB) +
CHR$(MSB)
NEXT I
(call sort on TG$( ) )
FOR I = 1 TO n
KEY$ = RIGHT$ (PH$(I),2)
KEY = ASC (LEFT$(KEY$,1))
KEY = KEY + 256*ASC(RIGHT$
(KEY$,1))
PRINT NM$(KEY),PH$(KEY), S$(KEY)
NEXT I
```

Program Listing 2

```
1 REM            FIGURE 2 - SAMPLE PROGRAM TO LOAD & CALL A/L SOR
T
2 '
3 '                        31 AUG 80
4 '
10 CLEAR1000:CLS:PRINT"WAIT...SETTING UP SORT A/L PROGRAM"
20 DATA205,127,10,34,42,65,17,251,255,25,229,221,225,221,110,0,2
21,102,1,43
30 DATA43,34,40,65,221,126,2,50,44,65,42,42,65,229,221,225,42,40
,65,229
40 DATA253,225,62,0,50,39,65,221,110,4,221,102,5,221,94,1,221,86
,2,221
50 DATA78,0,221,70,3,58,44,65,254,0,40,14,120,254,3,56,2,5,5,121
60 DATA254,3,56,2,13,13,26,190,0,40,4,56,18,48,39,19,35,5,13,121

70 DATA254,0,40,7,120,254,0,40,25,32,231,253,43,253,229,193,221,
35,221,35
80 DATA221,35,120,177,32,177,58,39,65,254,0,32,153,201,229,213,2
21,229,225,17
90 DATA29,65,1,3,0,237,176,221,229,209,1,3,0,237,176,33,29,65,1,
3
100 DATA0,237,176,62,1,50,39,65,209,225,24,195,70,73,78,73,83,72
,76,68
110 DATA9,65,44,40,68,69,41,228,2,8
120 '
130 TM=PEEK(16598)+256*PEEK(16599):IFTM>32767THENTM=TM-65536
140 TM=TM-190:REM TM=(TOP-MEM) - (SIZE OF A/L PROGRAM) = START P
OINT FOR PROGRAM
150 ZZ$=STRING$(255,0):REM SET UP DUMMY STRING TO HOLD PROGRAM
160 FORI=0TO189:READ J:POKE (TM+I),J:T=T+J:NEXTI
170 REM LINE ABOVE POKES PROGRAM AND COMPUTES "HASH" TOTAL
171 '
172 REM LINES 180THRU 200 CAN BE DELETED ONCE YOU HAVE THE DATA
CORRECTLY TYPED
173 '
180 L=10:REM NEXT LINE CHECKS HASH TOTAL, IF WRONG, DISPLAYS EAC
H DATA LINE FOR RECONCILIATION -- DELETE ONCE DATA ARE OK
185 '
190 IFT<>18217 THEN PRINT"ERROR IN DATA...CHECK DATA STATEMENTS"
:RESTORE:FORI=0TO189STEP20:L=L+10:PRINT"DATALINE";L:FORJ=0TO19:R
EADK:PRINTUSING"### ";K;:NEXTJ:READK:PRINTUSING"### ";K:INPUT"<E
NTER> FOR NEXT LINE";K:NEXTI:STOP
195 '
200 PRINTTM:REM JUST FOR DEBUGGING, THIS LINE CAN BE DELETED
205 '
206 REM LINES 210-220 ARE NICE TO HAVE, BUT CAN BE DELETED
207 '
210 TS=PEEK(16599):REM FIND OUT TYPE SYSTEM 16,32 OR 48K
220 IFTS>191 THEN TS$=" 48K " ELSE IF TS > 127 THEN TS$=" 32K "
ELSE IF TS < 128 THEN TS$= " 16K "
230 BB=PEEK(16549):REM BB IS MSB OF BEGINNING BASIC ADDRESS ASSU
MED TO BE 66 FOR TAPE SYSTEM
235 '
236 REM THE PRINT STATEMENTS ON LINES 240 & 250 CAN BE DELETED,
TOO
240 IF BB<>66 THEN DEFUSR1=TM:PRINT" NOW SET UP FOR";TS$;"DISK S
YSTEM'S USR1"
250 IFBB=66 THEN POKE16526,65:POKE16527,PEEK(16599):PRINT"NOW SE
T UP FOR";TS$;"TAPE SYSTEM'S USR"
260 '
270 REM THIS IS END OF MODULE TO SET UP THE SORT PROGRAM
280 REM FOR ANY SIZE SYSTEM, TAPE OR DISK
290 '
300 'TEST DATA GOODIES FOLLOW
310 '
320 DATASMITH,BAKER,ABLE,JONES,M,HOTEL,KILO,GEORGE,ZULU,CABIN,DO
G,FOX-TERRIER,FOXTERRIER,SMITHA,WERE,MHJKI,XYZ,ABC
330 A=18:DIMTG$(A):REM SET UP TEST ARRAY
340 FORI=1TO A
350 READTG$(I)
360 PRINTTG$(I)
370 NEXTI
375 '
376 REM          LINE 380 IS JUST SHOWBIZ
377 '
380 IFBB<>66 THEN PRINT"BEGINNING SORT AT ";TIME$
382 '
383 '
384 REM                   LINE 390 IS THE ACTUAL SORT CALL
385 REM                   (NOTE IT USES BB TO DETERMINE IF
386 REM                   SYSTEM IS TAPE OR DISK AND ASSUMES
387 REM                   SORT ARRAY IS TG$(N)  )
388 '
389 '
390 IFBB=66THENX=USR(VARPTR(TG$(1))) ELSE X=USR1(VARPTR(TG$(1)))
:REM USR1 FOR DISK, USR FOR TAPE
391 '
392 '
393 '
395 REM LINE 400 IS MORE SHOWBIZ
396 '
400 IFBB<>66 THEN PRINT"ENDING SORT AT  ";TIME$
410 REM DISPLAY ARRAY IN ASCENDING ORDER
420 FORI=1TOA
430 PRINTTG$(I)
440 NEXTI
450 INPUT"<ENTER> FOR DESCENDING ORDER";I
460 REM NOW IN DESCENDING ORDER
470 FOR I = A TO 1 STEP -1
480 PRINT TG$(I)
490 NEXTI
495 '
500 REM ROLL YOUR OWN SORT ARRAY
510 INPUT"<ENTER> TO INPUT 18 NEW STRINGS & SORT";I
520 FORI=1TO18
530 PRINT"ENTER STRING #";I;:INPUT TG$(I)
540 NEXT        I
550 GOTO 390
560 END
```

One potential problem with the tag sort is that it adds two bytes to the target string and could lead to a bad sequence in the case of strings of unequal lengths. To visualize this, take the case where TG$(255) = ABC and TG$(256) = ABCD. The concatenated strings (in ASCII decimal) would be:

65 66 67 255 0 ("ABC" + KEY$ of 255)
65 66 67 68 1 1 ("ABCD + KEY$ of 156)

Comparing the two strings, the sort program finds equality through the first three bytes but calls for a switch upon examining byte four of each string (255 is greater than 68). The result places ABCD before ABC, a bad sequence.

Refer again to Listing 3 and the paragraphs labeled CHTAG, ADJB and ADJC. These paragraphs adjust the lengths over which the byte-for-byte comparisons are to be made, to avoid comparing tag bytes with data bytes. These adjustments are made only if the calling program had previously assigned some value to TG$(0). If TG$(0) is left unused, as a null string (zero length), then the respective lengths are not adjusted. To enable a tag sort as opposed to a normal sort, we must intialize TG$(0) to some dummy value.

Out of Sorts

For those who couldn't care less how it works, this sort technique can be implemented with the code shown in Program Listing 2, lines 20–160 and lines 230–250. The actual sort call (assuming the target array is TG$( )) can be placed anywhere in the Basic program and is one of the following two calls (depending upon whether the system is a tape or disk system):

X = USR(VARPTR(TG$(1)))... Tape system
X = USR1(VARPTR(TG$(1)))... Disk system

I have a homebrew mailing list program with around 120 records on the disk. I often sort it on fields ranging in size from 15 to 25 bytes. Not counting the time it takes to read in the fields from the disk or to access and print the records after I have done the tag sort, the sort itself takes about five seconds. Unfortunately, I can't give you any comparative times for a pure Basic sort on my files—I don't have the time. ■

Program Listing 3

```
                00010 ;                    FIGURE 3...USR ROUTINE
                00020 ;
                00100 ;          SORT UTILITY (TAGSRT/SRC & TAGSRT/OB
J)
                00110 ;RELOCATABLE VERSION (30 AUG 80)
                00120 ;GIVENS:
                00130 ;  (1)  BASIC CALLING PROG WANTS TO SORT STR
ING ARRAY TG$(N)
                00140 ;  CALLING VIA "X=USR1(VARPTR(TG$(1)))".
                00150 ;  (2)  IF SORT IS A TAGSORT (WITH 2-BYTE TA
G CONCATENATED
                00160 ;  TO EACH ARRAY ELEMENT), THEN TG$(0) MUST
BE ASSIGNED
                00170 ;  SOME VALUE, OTHERWISE TG$(0) MUST BE NULL

                00180 ;
                00190 ;ON RETURN:
                00200 ;  POINTERS IN ARRAY TG$(N) WILL HAVE BEEN R
EARRANGED SO
                00210 ;THAT THE ARRAY IS ORDERED IN ASCENDING SEQU
ENCE.
                00220 ;
                00221 ;PROGRAM MAY BE LOADED INITIALLY TO 32000-32
180
                00222 ;THEN RELOCATED TO WHEREVER USER DESIRES
                00223 ;USING PEEK&POKE
                00224 ;
4127            00230          ORG      4127H ;USE ARITH WK AREA OF
ROM/RAM
411D            00240 SAVE     EQU      411DH
4127 00         00250 FLAG     DEFB     0 ; "DID WE SWITCH" FLAG: 1=
YES, 2=NO
4128 0000       00260 ELEMS    DEFW     0000 ;HOLDS # OF ELEMENTS (M
INUS 1) IN ARRAY
412A 0000       00270 START    DEFW     0000 ;HOLDS VARPTR (TG$(1))
412C 00         00280 TGNOTG   DEFB     0 ;0=NO TAGSORT, >0 MEANS DO
 A TAGSORT
0A7F            00290 GETARG   EQU      0A7FH ;ROM ROUTINE TO PASS V
ARPTR ARGUMENT TO HL
7D00            00295          ORG      32000 ;CAN BE ANYWHERE
                00296 ;INITIAL LOCATION OF PROGRAM; RELOCATABLE
7D00 CD7F0A     00300 BEGIN    CALL     GETARG  ;GET POINTER INTO HL

7D03 222A41     00310          LD       (START),HL ;(START)=VARPTR(T
G$(1))
7D06 11FBFF     00320          LD       DE,-5
7D09 19         00330          ADD      HL,DE ;POINT TO # OF ARRAY E
LEMENTS
7D0A E5         00340          PUSH     HL
```

program continues

program continued

```
 7D0B DDE1       00350          POP     IX
 7D0D DD6E00     00360          LD      L,(IX+0) ;GET # OF ARRAY ELE
MENTS
 7D10 DD6601     00370          LD      H,(IX+1)
 7D13 2B         00380          DEC     HL ;LESS 1 (FOR TG$(0))
 7D14 2B         00390          DEC     HL ;LESS 1 FOR SORT
 7D15 222841     00400          LD      (ELEMS),HL ;SET # OF ARRAY E
LEMS LESS 2
 7D18 DD7E02     00410          LD       A,(IX+2) ;GET LEN(TG$(0))
 7D1B 322C41     00420          LD      (TGNOTG),A ;SAVE IT
 7D1E 2A2A41     00430 SORT     LD      HL,(START) ;GET 1ST VARPTR I
NTO IX
 7D21 E5         00440          PUSH    HL
 7D22 DDE1       00450          POP     IX
 7D24 2A2841     00460          LD      HL,(ELEMS)
 7D27 E5         00470          PUSH    HL
 7D28 FDE1       00480          POP     IY ;# OF ELEMS IN IY
 7D2A 3E00       00490          LD      A,0
 7D2C 322741     00500          LD      (FLAG),A
 7D2F DD6E04     00510 DOTAG    LD      L,(IX+4) ;POINTER TO N+1
 7D32 DD6605     00520          LD      H,(IX+5)
 7D35 DD5E01     00530          LD      E,(IX+1) ;POINTER TO N
 7D38 DD5602     00540          LD      D,(IX+2)
 7D3B DD4E00     00550          LD      C,(IX) ;LEN(TG$(N))
 7D3E DD4603     00560          LD      B,(IX+3) ;LEN(TG$(N+1))
 7D41 3A2C41     00570 CHKTAG   LD      A,(TGNOTG) ;CHECK IF TAGSORT

 7D44 FE00       00580          CP      0
 7D46 280E       00590          JR      Z,EXAMIN ;NOT A TAGSORT
                 00600 ;TAGSORT, SO: MUST ADJUST B&C BY -2
 7D48 78         00610 ADJB     LD      A,B
 7D49 FE03       00620          CP      3 ;B MUST BE >= 3 BYTES
 7D4B 3802       00630          JR      C,ADJC ;IF NOT, DON'T ADJUST
 IT
 7D4D 05         00640          DEC     B ;ELSE B=B-2
 7D4E 05         00650          DEC     B
 7D4F 79         00660 ADJC     LD      A,C ;SAME BUSINESS WITH C
 7D50 FE03       00670          CP      3
 7D52 3802       00680          JR      C,EXAMIN
 7D54 0D         00690          DEC     C
 7D55 0D         00700          DEC     C
                 00710          ;RETURN WITH B&C ADJUSTED BY -2 IF T
AGSORT
                 00720          ;ELSE RETURN WITH B&C UNTOUCHED.
 7D56 1A         00730 EXAMIN   LD      A,(DE)
 7D57 BE         00740          CP      (HL)
 7D58 00         00750          NOP ;(NOP JUST FOR DEBUGGING)
 7D59 2804       00760          JR      Z,AGN
 7D5B 3812       00770 LT       JR      C,NEXT ;IF N>=N+1 THEN NEXT
TAG
 7D5D 3027       00780 GT       JR      NC,SWITCH ;N<N+1
 7D5F 13         00790 AGN      INC     DE ;HERE, BOTH ARE = SO FAR.

 7D60 23         00791          INC     HL ;INCREMENTING BOTH POINTE
RS
 7D61 05         00800          DEC     B ;DEC LEN(TG$(N+1))
                 00810
 7D62 0D         00820          DEC     C ;DEC LEN(TG$(N))
 7D63 79         00830          LD      A,C
 7D64 FE00       00840          CP      0 ; AT END OF TG$(N)?
 7D66 2807       00850          JR      Z,NEXT ;YES, THEN SEQ IS OK
 7D68 78         00860          LD      A,B ;NO:
 7D69 FE00       00870          CP      0 ;AT END OF TG$(N+1)?
 7D6B 2819       00880          JR      Z,SWITCH ;YES, THEN OUT OF S
EQ
                 00890          ;BECAUSE TG$(N) IS LONGER AN
D BOTH
                 00900          ;WERE = UP TO LEN(TG$(N+1)).

 7D6D 20E7       00910          JR      NZ,EXAMIN
 7D6F FD2B       00920 NEXT     DEC     IY ;# OF ELEMS
 7D71 FDE5       00930          PUSH    IY
 7D73 C1         00940          POP     BC
 7D74 DD23       00950          INC     IX ;STEP IX REG TO NEXT VARP
TR
 7D76 DD23       00960          INC     IX
 7D78 DD23       00970          INC     IX
 7D7A 78         00980          LD      A,B ;CHECK TO SEE IF AT END
OF ARRAY
 7D7B B1         00990          OR      C
 7D7C 20B1       01000          JR      NZ,DOTAG ;NOT AT END OF ARRA
Y
 7D7E 3A2741     01010          LD      A,(FLAG) ;@ END, CHECK IF WE
 HAVE SWITCHED
 7D81 FE00       01020          CP      0
 7D83 2099       01030          JR      NZ,SORT ;WE HAD SWITCHED...M
AKE ANOTHER PASS
 7D85 C9         01040          RET ;FULL PASS WITH NO SWITCHES...BA
CK TO BASIC
                 01100 ;ROUTINE TO SWITCH NTH AND N+1TH ELEMENTS
 7D86 E5         01110 SWITCH   PUSH    HL ;POINTS TO VARPTR T$(N)
 7D87 D5         01120          PUSH    DE
 7D88 DDE5       01130          PUSH    IX ;==> VARPTR(TG$(N))
 7D8A E1         01140          POP     HL ;==> VARPTR(TG$(N))
 7D8B 111D41     01150          LD      DE,SAVE
 7D8E 010300     01160          LD      BC,3 ;SIZE OF VARPTR
                 01170          ;VARPTR IS:
                 01180            ;(1) STRING LENGTH
                 01190            ;(2) LSB OF ADDRESS OF STRING
                 01200            ;(3) MSB "     "    "   "
 7D91 EDB0       01210          LDIR    ;MOVE VARPTR(N) TO 3-BYTE SA
VE AREA
                 01220          ;NOW HL ==> VARPTR T$(N+1)
 7D93 DDE5       01230          PUSH    IX ;==> VARPTR(TG$(N))
 7D95 D1         01240          POP     DE ;AND DE ==> VARPTR T$(N)
 7D96 010300     01250          LD      BC,3
 7D99 EDB0       01260          LDIR ;MOVE VARPTR(N+1)TO VARPTR(N)
 7D9B 211D41     01270          LD      HL,SAVE
                 01280          ;DE ALREADY ==> VARPTR T$(N+1)
 7D9E 010300     01290          LD      BC,3
 7DA1 EDB0       01300          LDIR ;MOVE SAVED(VARPTR(TG$(N))) TO
VARPTR(TG$(N+1))
 7DA3 3E01       01310          LD      A,1 ;TURN "SWITCH" FLAG ON
 7DA5 322741     01320          LD      (FLAG),A
 7DA8 D1         01330          POP     DE
 7DA9 E1         01340          POP     HL
 7DAA 18C3       01350          JR      NEXT
 7DAC 46         01360 EOJ      DEFM    'FINISH'
      49 4E 49 53 48
 0000            01370          END
 00000 TOTAL ERRORS
```

Where one will learn from another.

Erudite Arrays

George Barnes, Ph.D.
Nevada Department of Education
400 West King Street
Capitol Complex
Carson City, NV 89710

Many of the statistical applications I have for my Level I TRS-80 require multiple arrays and matrices. To handle these applications, I developed a simple system for using the one array in Level I as a series of arrays or as a multidimensional matrix. Basically, I use various portions of the one available array for different purposes.

Multiple Arrays

If you need to set up two arrays, and can set an upper limit X as the number of items in one of the arrays, then one array can take the form A(I) and the other array can take the form A(X+I). The variable I identifies the item in either array with which you are concerned. This system can be expanded to produce the effect of any number of arrays. For example, if four arrays of X length are desired, they could be designated A(I), A(X+I), A(2×X+I), A(3×X+I), where (I) represents the number of the entry in any array.

If you want to use arrays with different lengths within the same program, you should alter this system slightly. For example, if you want three arrays which include 10, 100 and 500 numbers, your designations might be A(I), A(10+I) and A(10+100+I).

Matrices

As you can see, multiple arrays are fairly simple, but what do you do if you need to set up a matrix? You basically do the same thing to set up a matrix that you do to set up arrays, only you do it more times. To set up an (X, Y) matrix with X values of 1 through A and Y values of 1 through B, you need B arrays of 1 through A. Any cell (X,Y) in such an array can be designated A(X+Y×A).

To visualize this more clearly look at the 3 by 4 matrix in Table 1. Notice that in the one array available, the array Y=1, X=1 to 3 sits on top of the array Y=2, X=1 to 3, which sits on top of the array Y=3, X=1 to 3. You will also notice that the array positions A(0) through A(3) are not used. For programs which don't come close to using your computer's memory capacity, it won't be necessary to correct this. When it is necessary to

		X attributes (A=3)		
		X=1	X=2	X=3
Y attributes (B=4)	Y=1	A(X+Y*A)= A(1+1*3)= A(4)	A(X+Y*A)= A(2+1*3)= A(5)	A(X+Y*A)= A(3+1*3)= A(6)
	Y=2	A(X+Y*A)= A(1+2*3)= A(7)	A(X+Y*A)= A(2+2*3)= A(8)	A(X+Y*A)= A(3+2*3)= A(9)
	Y=3	A(X+Y*A)= A(1+3*3)= A(10)	A(X+Y*A)= A(2+3*3)= A(11)	A(X+Y*A)= A(3+3*3)= A(12)
	Y=4	A(X+Y*A)= A(1+4*3)= A(13)	A(X+Y*A)= A(2+4*3)= A(14)	A(X+Y*A)= A(3+4*3)= A(15)

Table 1

conserve memory, cells in an (X,Y) matrix can be designated A(X + Y × A − A − 1).

Many statistical applications require three or four dimensional matrices. To create these you can expand upon the two-dimensional matrix designations just described. For example, any cell (X, Y, Z) in a three-dimensional matrix can be designated A(X + A × Y + A × B × Z), and any cell (X, Y, Z, W) in a four-dimensional matrix can be designated A(X + A × Y + A × B × Z + A × B × C × W), where the maximum values of X, Y and Z are A, B and C. If memory space is at a premium, positions in three and four dimensional matrices can be written as A(X + A × Y + A × B × Z − A − 1) and A(X + A × Y + A × B × Z + A × $B × C × W − A − 1).

Matrix Subroutines

As you can see, the expressions which represent matrix positions are sometimes rather long, and they can eat up memory space fast if they are repeated many times in a program. The solution to this problem is to handle them with a subroutine. First set the dimensions of your matrix, and identify a particular cell's coordinates using the following variables.

Number of dimensions in matrix	Dimensions of matrix *	Cell Coordinates
2	A	X, Y
3	A, B	X, Y, Z
4	A, B, C	X, Y, Z, W

*One less than the total number of dimensions is all that needs to be set.

Use the command GOSUB 10000 to obtain a value H such that A(H) which represents the cell which you have identified with cell coordinates.

For two dimensional matrices:

```
10000 H=X+Y×A−A−1 : RET.
```

For three dimensional matrices:

```
1000 H=X+A×Y+A×B×Z−A−1 : RET.
```

For four dimensional matrices:

```
10000 H=X+A×Y+A×B×Z+A×B×C×W−A−1 : RET.
```

The subroutines which do this task are marvelously short, as you can see, and they give the Level I TRS-80 tremendous power for statistical applications. ■

These utilities do more than aid Level I to Level II conversions.

Split and Splice

Dr. Stephen Mills
404 Wilson Avenue
Kinston, NC 28501

As programs grow, so do management and debugging problems, and soon the programmer is feeling the lack of text manipulation options in the Level I format. For instance, suppose you are 8K into a program and have worked out a 1K subroutine useful elsewhere. Or suppose you're having trouble with lines 1000–1100, and everything else above and below these lines is a nuisance to the debugging operation. Or suppose you're on the verge of going to Level II, and you have some very long programs to convert. Will there be enough RAM to load your Level I program along with the conversion tape program?

What do you do? Just delete part of the programs in question and work with or CSAVE the rest. To do this in Level I, type the number of the lines you want to delete with no spaces or program material following the number, and the lines are deleted from the program text. The method is simple, but it soon becomes very tedious.

It was ultimately the conversion question which led me to write Split, a program which cut my long Basic programs into pieces which could be converted separately and merged again into a Level II program. But Split is equally serviceable for the other problems I mentioned, since its function is simply to split a program in two, saving the discarded portion to cassette (if you so desire) and leaving the remaining lines in the computer.

Split is simple to use. Just put the Split cassette in the recorder with the Basic program that requires surgery already in memory. CLOAD Split on top of it—a process which takes only seven seconds. Then enter a breakpoint line number, and specify whether you want the lower lines saved on tape or tossed into the bit bucket. If you want to save it, Split will turn on the cassette and CSAVE everything below the breakpoint line. Afterward, your program list will contain only the breakpoint line and those above it, which you can rework, CSAVE or split further. This utility makes life easier for the ambitious Level I programmer. You can make separate recordings of subroutines and program variants without hesitation.

> *"You can make separate recordings of subroutines and program variants without hesitation."*

The perfect companion for Split is Splice. Once you have your Basic programs broken down into modules for editing and debugging, you'll want to be able to reassemble them. Splice modifies Level I's CLOAD operation, so your loaded program does not cancel the one previously in memory. You can merge Basic programs with a touch of the Enter key and return to Basic control by pressing Break. The newly loaded programs are stacked end-to-end sequentially.

The catch, as you may have guessed, is that Split and Splice cannot be written in Level I Basic. Level I is absolutely intolerant of system self-modifications like these. You will need T-Bug or an appropriate editor/assembler tape to produce your own copies. With T-Bug, you could produce Split and Splice tapes just by punching in the values from the listings provided. However, I shall be doing more than just explaining the workings of Split and Splice.

I'll discuss general techniques for using machine language programs to complement and facilitate Level I Basic. If you're writing long programs in Level I and have a hobbyist's interest in computing, assembly language is probably the next frontier for you to conquer. I'll assume that the reader knows some of the principles of assembly or machine language programs, although I won't be presupposing sophisticated practical experience or knowledge of TRS-80 hardware and Level I ROM.

The Trouble with Level I

To understand how Split and Splice work, it helps to know how Level I Basic programs are managed in your computer. Level II stores its Basic statements in a condensed, one-byte code. With one exception, Level I Basic is stored exactly as you see it in the List. Each line is a string of ASCII alphanumeric characters. Each line ends with the decimal value 13 (0D hexadecimal), which signifies the Enter key and operates like the carriage return of a typewriter, resetting your display to the next line.

The one exception is an important one—the line number itself. It is not stored as you type it or as it is displayed but is converted to its hexadecimal equivalent. Every line number is a signed, two-byte number. The largest number that can be represented this way (in 15 bits, plus one sign bit) is decimal 32767, which is why this is the largest number you can use for a Basic line. (Incidentally, the space which is automatically inserted between the number and the program statement is not stored; it is a consequence of how Level I ROM handles output conversions.)

This difference between Level I and Level II should explain the relative slowness of Level I execution; for in Level I, the text you typed in must be decoded every time it is executed, while in Level II it is partially decoded in the storage format. It also explains why the merging technique discussed in "APPEND It!" (February 1980, *80 Microcomputing*) cannot be applied to Level I.

Level I programs are always stored in RAM beginning at address 4200H or 16896 decimal. The section of memory dedicated to the video display ends at 3FFFH or 16383. What about those 513 bytes in between? Basic does use the area below 4200H for some of its operations, but the text of the program itself never goes there. When a program is not running, though, most of the space between video memory and the Basic text is free. There are two good reasons why our assembly language programs ought to go in this area. One is that Split and Splice are then operable with any size memory and any program length, since this area is protected.

However, there are seven addresses in this area which should be respected, since they are concerned with the computer's general housekeeping requirements. The first six are actually three two-byte codes, occupying 4068H-406DH. 4068H contains the address of the current position of the video cursor. Next, at 406AH, the computer stores the address of the end of available memory; this is 7FFFH for a 16K system and 4FFFH for 4K. The value is apparently determined when the computer powers up, and is not recalculated after a reset or 01C9H reentry, so if you change this address, Level I ROM won't be able to determine how much RAM is available (of course, one could deliberately lower this value to protect a high-core machine language program). 406CH contains the address of the last byte plus one of the resident Basic program.

After you type New, this address has the value 4200H, just the way it was before anything was loaded. The program text is still there, but ROM loses track of it when 406CH is reset. Split and Splice involve deliberate manipulation of the value stored at 406CH. Finally, 4090H contains a one-byte reference code to keep the status of output port 0FFH in order. This port controls cassette operation, and the 64-character mode of the video display.

I mentioned a second reason for storing our programs below 4200H. This concerns one of the obstacles facing any assembly language programming for a

Level I computer: You must load something in addresses 41FEH and 41FFH, the two bytes immediately before the start of Basic. As long as two bytes must go there, we might as well assemble the whole code below 4200H. Split and Splice have been tailored to fit snugly between 4090H and 4200H.

Level I does not have a System format. The only way of getting non-Basic programs into the computer and running them is via tape, and it is important for the Level I user to know how this is done. When you type CLOAD and the TRS-80 goes into its input routine, a value is stored at the two bytes just below 4200H. This value is the address to which the program counter will go when the CLOAD is completed. Normally, with a Basic program, this is an address in ROM (more about it later). But, if you CLOAD a tape which inputs a different value to this address, you can take control away from ROM. This is exactly what T-Bug and the System tape of Radio Shack's Editor/Assembler do; they overwrite the stored ROM address with the desired RAM address—the starting address of the machine language code.

There are two more special problems before Split and Splice can work. First, we can't put Split in the desired area because T-Bug, or the System tape, is already there. If you try to produce a copy of Split using T-Bug, you'll be changing T-Bug itself, and the System tape contains the 500-baud input program necessary to get an assembled program into Level I. The solution is, of course, to assemble at a higher address, transfer the program down and produce a 250-baud tape for future use. For this purpose we have a subroutine, applicable not only to Split but to any assembly language program for producing a second-generation program that can be loaded and run without messing around with the T-Bug or System tape.

The second problem is important for programs like Split, which interact with Basic. We have to place the program's starting address at 41FEH. So it would seem we have to write a program ending at 41FFH and insert the required two-byte value in the last two locations. But unfortunately, Level I ROM's CLOAD technique prevents us from doing that. When a tape is CLOADed, the actual storage is one byte longer than the program itself. The last byte recorded by CSAVE is a checksum byte and when the computer finishes loading, it compares its tally of the input with the one claimed by the tape to determine if everything loaded correctly. If they don't tally, you get that all too familiar What? message after the load.

Unfortunately, ROM doesn't just look at the checksum; it stores it right after the end of the program. So if we simply assembled our start address word to go into 41FEH-41FFH, the checksum would automatically get POKEd into 4200H—which contains the low-order byte of our Basic program's first line number.

We could, of course, require inputting the Basic program after Split, but that would make Split very inconvenient to use. A better solution is to incorporate the checksum into the program itself. We formally end the program at 41FEH, which will contain the low-order byte of our two-byte address, and let the checksum byte fall in 41FFH—making sure that the checksum equals the high-order byte of our starting address. This little feat is also handled by our block-transfer and dump subroutine. In fact, the editor/assembler makes it so easy that we can do it without ever knowing what the actual values are!

Split

You can either duplicate the assembly code with an editor/assembler or just key in the machine language opcodes using T-Bug. Split (like Splice) is tailored to 255 bytes for readability to make it easy to compare its assembly location (from 4A00H) to its final working location (4100H). Note that Split itself ends at 4AFEH; the lines from 4AFF are the subroutine for producing the final product. If you use the editor/assembler, assemble Split, load your

Rose ©

You've Got TOTAL ACCESS ©

(specializing in TRS80 *)

TO YOUR COMPUTER HARDWARE & SOFTWARE NEEDS. CALL ROSE TODAY!

I've Got

★ **Disk Drives**
- TA400 (40-T) $289
- TA 800 (80-T) $419
- TA400 Flippy $319
- TA800 Flippy $439
- All with silver case and power supply

★ **CABLES**
- 2-Drive $23.95
- 4-Drive $33.95
- Extender Cable $14.95

★ **BARE DRIVES**
- TA400B $259
- TA800B $389
- TA400B Flippy $289
- TA800B Flippy $409

★ **EPSON PRINTERS**
- MX70 .... $379.00
- MX80 .... $475.00
- MX80 F/T $589.00
- Parallel Cables $29.95

★ **OPERATING SYSTEMS**
- TRSDOS 2.3 Disk & Manual ... $17.95
- LDOS $129.95
- NEWDOS/80 .... $129.95
- DOSPLUS 3.3, 3.3D $89.95

★ **IRON**
- Disk Drive Power Supply, Single $37
- Disk Drive Case (silver) and Base $19
- Memorex Diskettes (bx of 10) $25

90 day warranty on drives. Add $5.00 freight per drive in Cont. US. UPS COD charge $1.40. There is also a 15 day FREE TRIAL on TA drives. If not completely satisfied I'll refund your money (less shipping). I'll take exception to improper use or mishandling.

★ **USERS GROUPS**
When your club makes a group buy, be sure to call me for a price.
Rose

TRS80 * COMPUTERS

★ **COMPUTERS**
- 26-1061 Mod III, LI, 4K $595
- 26-1062 Mod III, LII, 16K $850
- 26-1063 Mod III, 32K/2 Disk/RS232 .... $2150
- 26-4002 Mod II, 64K $3300
- 26-3001 Color Computer, 4K... $330
- 26-3002 Color Computer, 16K.. $510
- 26-3501 Pocket Computer $212

★ **DISK EXPANSION, MOD II**
- 26-4160 One Disk $977
- 26-4162 Three Disk $1998
- LOBO drive units also available.

★ **PERIPHERALS**
- 26-1140 ØK Exp. Interface $255
- 26-1141 16K Exp. Interface $305
- 26-1142 32K Exp. Interface $355
- 26-1172 D.C. Modem II $135
- 26-1206 CTR-80 Recorder $51
- 26-3503 P.C. Interface $42

★ **PRINTERS**
- 26-1165 Line Printer V $1581
- 26-1166 Line Printer VI $986
- 26-1158 Daisy Wheel II $1694*

* LIMITED AVAILABILITY

The complete line of Radio Shack products is available through TA with standard RS limited warranty. Call me for price and delivery. Just 'cause you don't see it, don't mean we ain't got it.
ROSE

★ **SOFTWARE**
We have Adventure, Big 5, Circle J, Med Systems, SBSG, Snapp, Radio Shack and much more. Call or write. DO IT NOW!

ROSE'S SPECIAL OF THE MONTH

★ PRINTERS ★

EPSON MX70 ... $369.00

EPSON MX80 ... $475.00

STARWRITER 45cps DAISYWHEEL .. $1795.00

Parallel Cable for TRS80 for any of the above $29.95

You can't live without a bunch of these!

Rose's T-SHIRTS
With any order of $1,000.00 or more, I'll send you one of my **"MINI-FLOPPY" T-Shirts!**

REAL TIME CLOCK-CALENDER
T-Timer© $89.95

DRIVE TABS
Disk drives need identity! No more "that drive"! Make 'em be somebody - call 'em 0, 1, 2 and 3. They'll never forget ya!
- Set of 4 .... $4.50
- Set of 2 .... $2.75

ORDER NOW! TOLL FREE 800-527-3582

Write or call Rose TOLL FREE at **1-800-527-3582** (Texas residents call 214-234-1770). Please use the toll free lines for orders and literature requests only. Technical help or service use the Texas line. You can pay by VISA or MASTERCARD, you can send check or money order (allow a couple of weeks for personal checks to clear) or order COD (we ship COD's cash, certified check or money order only). Rose will take American money in just about any form. Add freight (UPS where possible) on all orders under $1000. If you buy $1000 or more Rose eats the freight! Texas residents cough up 5% sales tax. Allow 2-4 weeks for delivery. Order today - I need the money!

TOTAL ACCESS©
P.O. BOX 3002
RICHARDSON, TX 75080

214-234-1770

* TRS80 & Radio Shack are trademarks of Tandy Corp.
© Copyright 1981 TOTAL ACCESS

✓6

TOACC/23
TOACC1/24

✓See List of Advertisers on page 354

500-baud version with the System tape and hit the slash key. If you produce it using T-Bug, after you've typed in the whole code and double-checked it, type J 4AFF. Either way, the CRT should read Ready Cassette, which means that the block-transfer and the checksum adjustment are complete, and the computer is ready to save your program to cassette.

Insert a blank cassette and Enter, and a copy will be recorded. The Ready Cassette message will return when the copy has been made, and you can make as many as you like. You'll probably want to make a whole string of them, since this eliminates the constant rewinding and leader delay. Exit the recording loop by pressing the Break key.

If you understand assembly language, you can follow the workings of Split in the mnemonics and notes of Program Listing 1. Remember that the assembly has to be modified for the post-transfer location. For that purpose, I have defined the label DIF for the displacement the program receives by transfer. DIF is used to adjust other assembly addresses for JP, CALL and some internal LD instructions, while relative jumps (JR) are calculated correctly without the displacement value.

Lines 150–380 are video display and cassette operation subroutines. Initially, Split loops through a keyboard input ROM routine, waiting for the value of 0DH, the Enter key. It restarts if Clear is pressed—a necessary safeguard since Clear automatically functions when ROM's key scan routine is used. It is useful to know something about the operation of the routine at 0B40H in ROM, since the prime or alternate registers contain some useful information. The active (non-prime) A register contains the input byte, or zero if nothing is entered during the scan. The B' register contains the ASCII code for the character pressed, a value which may or may not equal that in A, depending on the nature of the character and whether or not Shift is depressed. The D'E' register pair has the value of the keyboard address line of the key pressed. For our purposes, the important detail is that H'L' contains the cursor location.

After Enter is hit, the key scan loop falls through to line 480, which converts the string of numerals typed in and displayed into a hexadecimal, 16-bit number. It scans video memory backward from the cursor location to find the string and converts it directly from memory, using the H'L' pair, so there is no need to fetch the cursor location out of 4068H. This calculation aborts if it produces overflow, a negative number or encounters a non-numeric input. In other words, it must get a number between one and 32767 to pass to the next stage. If successful, beginning at line 940, Split searches the resident Basic program for the correct line. This is easy since the byte 0DH will only occur at the end of a line. The program scans the Basic text, comparing the two bytes after an 0DH with the calculated number. It will not take the first line of Basic (hence the start at 4202H in line 950) and does not scan quite to the end of the program (which is fetched from 406CH) to avoid reading the final carriage return and a coincidental match with the garbage bits that follow it. If the search fails, the program restarts.

When the program gets the address of the chosen line of Basic, the subroutine CTON is called. This asks if the lines below the breakpoint are to be saved on tape. In this case, the input comes not from ROM's key scan routine, but is taken directly from keyboard addressing. If Enter is pressed, the program goes to ROM's turn-on-cassette routine at 0FE9H. Note this is done by a JP rather than a Call. This causes the program counter to return to line 1180 directly after the ROM routine ends. If Break or Clear is pressed, the program restarts, allowing you to change your breakpoint. If one of the vertical arrow keys is pressed, the cassette is not turned on, and the value three is stored at 41A6H. 41A6H is where the displacement value of the relative jump at line 1220 is located. As Split is recorded, this value is zero, which means line 1220 does nothing. If the value is changed to three, the CSAVE Call at line 1230 is skipped over.

Finally, everything from the breakpoint to the end of the Basic program is block-moved down to 4200H. Radio Shack's published reentry point for Level I is 01C9H. So what's this 0EECH in line 1360? Well, that's a branching point that ROM goes to after a CLOAD. For our program, it is necessary to store the new end-of-program address at 406CH. Reentering Basic at this address gets this done for us automatically. After this is done, ROM evaluates the Z-flag following the checksum test and goes to 01C9H if everything is okay. Since there is a test here, lines 1320–1340 of Split prepare for it by checking the last byte transferred, which should be the final carriage return of the Basic program. If it isn't, it reflects something seriously wrong, and you'll get the old What? message. But this is one case in which you're not likely to see it pop up.

Lines 1470 and 1480, by the way, self-test your Split to make sure it loaded correctly. If not, control goes to ROM address 08C9H (the old What? again). The resident Basic program will not be messed up by the bad CLOAD (unless the addresses were misread, but that will probably hang up the computer and require a reset), and the end-of-program marker will not be changed.

The ADJ subroutine (lines 1630 to the end) accomplishes the relocation of Split and the checksum adjustment. Note that the stack pointer is initially reassigned out of the way; otherwise, the transfer will crash into the stack defined by Basic. During the transfer, every byte moved is subtracted from the accumulator to calculate to checksum adjustment. Why subtract? The checksum is something of a misnomer for the final byte of a Basic tape dump. The final byte stored on tape is not the sum of the bytes of the program, but the two's complement negation of the sum. The CLOAD routine test is not performed by a CP instruction between the value read and the value reached during the load. Rather, the checksum byte is added to the rest. This should give a final total of zero if everything went right. Hence we keep a running subtract of the bytes as they are moved.

Finally, the value 41H is subtracted from that result. 41H is the checksum byte we want at 41FFH—the high byte of Split's starting address. This value is then POKEd into a convenient unused zero byte in the transferred Split, and Split is ready to be CSAVEd by a call to the ROM routine. The Break key will get you out of the CSAVE loop, and the Enter key will keep making copies of Split.

Splice

Splice is the other side of the editing coin. It, too, loads into the safe zone below 4200H. You can CLOAD it with your computer's memory empty, but since Splice is made with the same checksum adjusted method as Split, you can also begin with your first Basic program already in memory. Splice is not a one-shot program like Split; it loops continuously until you Break, so you can merge as many programs as you want. Splice is very useful for reassembling programs debugged in modules and getting complex blocks of data statements ap-

Program Listing 1

```
                00010          ;SPLIT
                00020          ;LEVEL I BASIC EDITING UTILITY
                00030          ;BY STEPHEN MILLS
                00040          ;
                00050          ;SHORT UTILITY TO DIVIDE A LEVEL I BASIC
                00060          ;PROGRAM AT A SELECTED LINE NUMBER
                00070          ;LINES BELOW BREAKPOINT ARE SAVED TO
                00080          ;CASSETTE, AND CONTROL IS RETURNED TO
                00090          ;BASIC MONITOR WITH UPPER LINES.
                00100          ;THIS EDTASM VERSION ASSEMBLES AT 4A00H
                00110          ;TRANSFERS TO 4100H AND GENERATES A
                00120          ;250 BAUD LEVEL I LOADABLE COPY
4A00            00130          ORG     4A00H
0900            00140 DIF      DEFL    STACK-41FEH     ;DISPLACEMENT
4100            00150 WRITE    DEFL    $-DIF
4A00 46         00160 HWRITE   LD      B,(HL)  ;GET BYTE CT
4A01 D5         00170          PUSH    DE      ;SAVE DE
4A02 23         00180          INC     HL      ;BUMP VIDEO PTR
4A03 7E         00190          LD      A,(HL)  ;GET CHARACTER
4A04 D7         00200          RST     10H     ;LEVEL I DISPLAY
4A05 10FB       00210          DJNZ    HWRITE+2
4A07 3E0D       00220          LD      A,0DH   ;FOR CAR RET
4A09 D7         00230          RST     10H     ;WHEN DONE
4A0A D1         00240          POP     DE
4A0B C9         00250          RET
410C            00260 CTON     DEFL    $-DIF
4A0C 21CB41     00270          LD      HL,CSMS ;CASSETTE
4A0F CD0041     00280          CALL    WRITE   ;DISPLAY IT
4A12 3A4038     00290 PAUSE    LD      A,(3840H)       ;KEYSCAN
4A15 E61F       00300          AND     1FH     ;FORGET UPPER BITS
4A17 28F9       00310          JR      Z,PAUSE
4A19 1F         00320          RRA     ;SEE IF ENTER
4A1A DAE90F     00330          JP      C,0FE9H ;FINISH IN ROM
4A1D E603       00340          AND     3       ;BREAK OR CLEAR?
4A1F 2006       00350          JR      NZ,START        ;ABORT IF SO
4A21 3E03       00360          LD      A,3     ;RELATIVE JUMP #
4A23 32A641     00370          LD      (41A6H),A       ;LOAD USECAS+1
4A26 C9         00380          RET
4127            00390 LSTART   DEFL    $-DIF
4A27 310042     00400 START    LD      SP,4200H        ;REINIT STACK
4A2A 2100F7     00410          LD      HL,STMS         ;MESSAGE
4A2D CD0041     00420          CALL    WRITE
4A30 CD400B     00430 WTZ      CALL    0B40H   ;KEYSCAN
4A33 FE0C       00440          CP      0CH     ;CLS?
4A35 28F0       00450          JR      Z,START ;YES-START OVER
4A37 FE0D       00460          CP      0DH     ;CAR RETURN?
4A39 20F5       00470          JR      NZ,WTZ  ;NO,SCAN AGAIN
4A3B D9         00480          EXX     ;GO TO PRIME REGISTERS
4A3C 3E20       00490          LD      A,20H   ;ASCII FOR BLANK
4A3E 2B         00500 NOBL     DEC     HL      ;DEC VIDEO POINTER
4A3F BE         00510          CP      (HL)    ;IS IT BLANK?
4A40 28FC       00520          JR      Z,NOBL  ;YES-KEEP GOING
4A42 D9         00530          EXX     ;CHANGE REGISTERS
4A43 AF         00540          XOR     A       ;CLEAR A
4A44 5F         00550          LD      E,A     ;CLEAR E
4A45 4F         00560          LD      C,A     ;CLEAR C
4A46 57         00570          LD      D,A     ;ZERO D
4A47 D9         00580 BUMP     EXX
4A48 7E         00590          LD      A,(HL)
4A49 2B         00600          DEC     HL
4A4A D9         00610          EXX
4A4B FE20       00620          CP      20H     ;SEE IF END OF
4A4D 2829       00630          JR      Z,BREAK ;INPUT
4A4F D630       00640          SUB     30H     ;SUB ASCII
4A51 38D4       00650          JR      C,START ;AND TEST
4A53 FE0A       00660          CP      10      ;FOR VALID
4A55 30D0       00670          JR      NC,START        ;INPUT
4A57 6F         00680          LD      L,A     ;STORE #
4A58 2600       00690          LD      H,0     ;CLEAR HI REG
4A5A 79         00700          LD      A,C     ;GET DECIMAL CT
4A5B 0C         00710          INC     C       ;AND BUMP IT
4A5C D5         00720          PUSH    DE      ;SAVE CURRENT SUM
4A5D B7         00730 SHIFT    OR      A       ;IS IT 0?
4A5E 2811       00740          JR      Z,DONE
4A60 29         00750          ADD     HL,HL   ;DOUBLE #
4A61 38C4       00760          JR      C,START ;CHECK OVERFLOW
4A63 E5         00770          PUSH    HL      ;TRANSFER TO
4A64 D1         00780          POP     DE      ;ANOTHER REG
4A65 29         00790          ADD     HL,HL   ;DOUBLE RESULT
4A66 38BF       00800          JR.     C,START ;CHECK OVERFLOW
4A68 29         00810          ADD     HL,HL   ;AGAIN FOR
4A69 38BC       00820          JR      C,START ;TIMES 8
4A6B 19         00830          ADD     HL,DE   ;HL = 10 TIMES #
4A6C 38B9       00840          JR      C,START ;OVERFLOW?
4A6E 3D         00850          DEC     A
4A6F 20EC       00860          JR      NZ,SHIFT
4A71 D1         00870 DONE     POP     DE      ;RESTORE SUM
4A72 19         00880          ADD     HL,DE   ;SUM ADDRESS
4A73 38B2       00890          JR      C,START ;OVERFLOW?
4A75 EB         00900          EX      DE,HL   ;PUT SUM IN DE
4A76 18CF       00910          JR      BUMP    ;GO FOR NEXT DIGIT
4A78 CB7C       00920 BREAK    BIT     7,H     ;IS # NEGATIVE?
4A7A 20AB       00930          JR      NZ,START        ;CANCEL IF SO
4A7C 2A6C40     00940          LD      HL,(406CH)      ;END OF PROG
4A7F 010242     00950          LD      BC,4202H        ;BASIC TEXT
4A82 C5         00960          PUSH    BC      ;HERE AFTER LINE #
4A83 ED42       00970          SBC     HL,BC   ;GET BYTE CT
4A85 E5         00980          PUSH    HL      ;AND MOVE IT
4A86 C1         00990          POP     BC      ;TO BC
4A87 E1         01000          POP     HL      ;START ADR IN HL
4A88 0B         01010          DEC     BC      ;FORGET CHECKSUM
4A89 0B         01020 CMPR     DEC     BC      ;AND CAR RET
4A8A 0B         01030          DEC     BC      ;ADJUST COUNT
4A8B 23         01040          INC     HL      ;AFTER TEST ADJUSTMENT
4A8C 3E0D       01050          LD      A,13    ;END OF LINE INDICATOR
4A8E EDB1       01060          CPIR    ;RESTART IF LINE NOT
4A90 2095       01070          JR      NZ,START        ;FOUND
4A92 7E         01080          LD      A,(HL)  ;GET LOW ADR
4A93 23         01090          INC     HL      ;POINTS TO HI-BYTE
4A94 BB         01100          CP      E       ;MATCH E?
4A95 20F2       01110          JR      NZ,CMPR ;RECYCLE
4A97 7E         01120          LD      A,(HL)  ;GET HI ADDR
4A98 BA         01130          CP      D       ;SEE IF MATCH
4A99 20EE       01140          JR      NZ,CMPR ;REDO IF NO MATCH
4A9B 2B         01150 MATCH    DEC     HL      ;BACK TO LOW ADR
4A9C EB         01160          EX      DE,HL   ;SET ASIDE
4A9D CD0C41     01170          CALL    CTON    ;START CASSETTE
4AA0 210042     01180          LD      HL,4200H        ;BASIC ADR
4AA3 E5         01190          PUSH    HL      ;SAVE IT
4AA4 D5         01200          PUSH    DE      ;SAVE ADDRESS
41A5            01210 USECAS   DEFL    $-DIF   ;BRANCH MADE BY CTON
4AA5 1800       01220          JR      $+2     ;DEFINES 0 JUMP
4AA7 CD4B0F     01230          CALL    0F4BH   ;SAVE PROGRAM 1
4AAA 2A6C40     01240          LD      HL,(406CH)      ;PROG TAIL
4AAD D1         01250          POP     DE      ;RESTORE ADDR
4AAE ED52       01260          SBC     HL,DE   ;CALC NEW PROG LENGTH
4AB0 E5         01270          PUSH    HL      ;MOVE TO BYTE
4AB1 C1         01280          POP     BC      ;COUNTER
4AB2 E1         01290          POP     HL      ;GET 4200H
4AB3 EB         01300          EX      DE,HL   ;SWAP
4AB4 EDB0       01310          LDIR    ;PROGRAM IS CUT DOWN
4AB6 2B         01320          DEC     HL      ;GET LAST TRANSFER
```

Program continues

Program continued

```
4AB7 7E       01330          LD     A,(HL)  ;AND SEE IF CAR RET
4AB8 FE0D     01340          CP     0DH     ;FOR 'CHECKSUM'
4ABA EB       01350          EX     DE,HL   ;GET END ADR IN HL FOR
4ABB C3EC0E   01360          JP     0EECH   ;RETURN TO BASIC
41BE          01370 STMS     DEFL   HISTMS-900H
4ABE 0C       01380 HISTMS   DEFB   12      ;MESSAGE LENGTH
4ABF 0D       01390          DEFB   0DH     ;CAR RET
4AC0 42       01400          DEFM   'BREAKPOINT?'
41CB          01410 CSMS     DEFL   $-900H
4ACB 1C       01420 HICSMS   DEFB   28      ;FINAL MESSAGE LENGTH
4ACC 3C       01430          DEFM   '<ENTER> TO CSAVE'
4ADC 0D       01440          DEFB   13      ;CAR RET
4ADD 5C       01450          DEFB   5CH     ;VERTICAL ARROW
4ADE 20       01460          DEFM   ' TO DELETE'
4AE8 CA2741   01470 TEST     JP     Z,LSTART        ;CHECKSUM
4AEB C2C908   01480          JP     NZ,08C9H        ;TEST
4AEE 0000     01490          DEFW   0       ;THIS IS
4AF0 0000     01500          DEFW   0       ;STACK
4AF2 0000     01510          DEFW   0       ;AREA
4AF4 0000     01520          DEFW   0
4AF6 0000     01530          DEFW   0       ;ALL
4AF8 0000     01540          DEFW   0       ;BYTES
4AFA 0000     01550          DEFW   0       ;EQUAL
4AFC 0000     01560          DEFW   0       ;ZERO
4AFE E8       01570 STACK    DEFB   TEST&0FFH       ;LO ADR BYTE
              01580          ;REMAINDER OF LISTING MOVES
              01590          ;ASSEMBLED PROGRAM TO BELOW
              01600          ;LEVEL I BASIC STARTING
              01610          ;ADDRESS AND CSAVES IT
              01620          ;FOR 250 BAUD LOADING
4AFF 31FE49   01630 ADJ      LD     SP,49FEH
4B02 11FE41   01640          LD     DE,41FEH
4B05 21FE4A   01650          LD     HL,4AFEH
4B08 01FF00   01660          LD     BC,0FFH ;BYTE CT.
4B0B AF       01670          XOR    A       ;CLEAR A
4B0C 96       01680 TRANS    SUB    (HL)    ;DO CHECKSUM
4B0D EDA8     01690          LDD            ;BLOCK MOVE
4B0F EA0C4B   01700          JP     PE,TRANS
4B12 D641     01710          SUB    41H     ;MODIFY
4B14 32FD41   01720          LD     (41FDH),A       ;POKE IT
4B17 21CB4A   01730 DUMPIT   LD     HL,HICSMS
4B1A 0611     01740          LD     B,17    ;PARTIAL MESSAGE
4B1C CD014A   01750          CALL   HWRITE+1        ;PREPARE CASSETTE
4B1F CD400B   01760 DLOOP    CALL   0B40H   ;KB INPUT
4B22 FE03     01770          CP     3
4B24 CAC901   01780          JP     Z,01C9H ;BREAK TO BASIC
4B27 FE0D     01790          CP     13      ;ENTER
4B29 20F4     01800          JR     NZ,DLOOP
4B2B CDE90F   01810          CALL   0FE9H   ;CASSETTE ON
4B2E 210041   01820          LD     HL,4100H        ;START ADR
4B31 11FF41   01830          LD     DE,41FFH        ;END+1 ADR
4B34 CD4B0F   01840          CALL   0F4BH   ;FOR LEVEL I
4B37 18DE     01850          JR     DUMPIT  ;REDO IF DESIRED
4AFF          01999          END    ADJ
```

pended to the program that uses them. The programs are stacked up in the order in which they are CLOADed. So you must load the pieces so their line numbers are ascending and don't duplicate or cross each other. Splice will give you an error message if the program you're trying to load is clearly not in Basic (i.e., it doesn't start at 4200H); your RAM is exhausted; or a checksum error occurs.

Unlike ROM's CLOAD routine, a badly loaded program is not added to the stack. That is, Splice tests the checksum before it modifies the end-of-program address 406CH and does not modify it on a checksum error. This maintains the integrity of your expanding program.

The heart of Splice begins at line 430. Basically, I have taken part of ROM's CLOAD routine and modified it to give the desired results. The critical part of the CLOAD concerns the addresses to be filled. After this part of the program is completed, control jumps to ROM (line 700) to finish the input and returns for the checksum test, end-of-program update and more merges. The address pushed in lines 430–440 is used for the ROM subroutine's return.

I could not call that subroutine at line 700 because of intervening stack operations (see line 470). There is a POP DE instruction near the end of the ROM routine (in 0F37H) which answers a PUSH DE at 0EF4H, an address skipped over by our modification of the CLOAD. Therefore, the return address had to be PUSHed before the DE PUSH to make the stack work correctly. The last-in, first-out stack action does not distinguish between register PUSHes and subroutine Call storage. Without this adjustment, our return address would have been POPped into DE at 0F37H, and program execution would have crashed into who knows what.

Normally during a CLOAD, the starting and ending addresses are read off the tape, and these values control the input.

the software source

Visit our new Retail Center at 20 Jericho Tpke., Jericho, Long Island, NY. Call (516) 333-2266 for hours & directions.

PROGRAMS UNLIMITED™

✓110

New! ORCHESTRA 85

TRS-80 MODEL I 16K TAPE OR DISK

Jon Bokelman and Software Affair score again! Now there's a music/sound generating system even more powerful and versatile than their extremely popular Orchestra 80!

Listen to what it does:

ORCHESTRA 85 plays in *stereo*, permits channel switching, allows stereo separation by both instrument and voice, and supports percussion effects.

ORCHESTRA 85 supports four voices (plus a fifth if a speed-up modification is installed).

ORCHESTRA 85 loads Orchestra 80 music (hundreds of songs), plays it in stereo. Compose! Orchestrate! Trade with others!

Listen to what's included:

Software: digital synthesizer, music language compiler, full screen editor, file manager, voice initialization utility.

Hardware: a single PC board that plugs into expansion connector on on keyboard or into screen printer port on expansion interface. Connects to any stereo amplifier system; external power source not required.

Unit does not void your Radio Shack warrantee. Comes complete with tape and disk versions of software, detailed instruction manual, sample music programs. (Speakers, amplifiers not included.)

All at a price that's music to your ears: ONLY $129.00

TRS-80 is a trademark of Tandy Radio Shack Corp.

New! MIZ' SPELL

TRS-80 MODEL I 48K DISK (Soon available for Models II & III)

Picture the fastest, most accurate schoolteacher you ever had, pointing out *all* your spelling errors so you can correct them instantly. This new spelling checker for Electric Pencil and Scripsit files is modeled after her: it's virtually impossible to misspell with MIZ SPELL!

Unlike other spelling checking programs:

MIZ SPELL will operate on *one* or *more* disc drives. (Some others require a minimum of two.)

MIZ SPELL will work with words of any length.(Others have limits on maximum word length.)

MIZ SPELL is interactive; you can correct mistakes as they are pointed out. (Some others require multiple, time-consuming passes.)

Check these features vs. other spelling checkers:

MIZ SPELL maintains separate dictionaries for specialized vocabularies, each with a 27,000 word capacity. (18,000 included.)

MIZ SPELL permits examination of unused dictionary space and allocation of space for new dictionaries.

MIZ SPELL learns new words from your text files as it check for spelling errors. (Words can also be deleted from the dictionary— but not accidentally, as with some other programs.)

MIZ SPELL offers such options as: ignore word in text, delete word in text, change word in text, add *all* remaining words, etc.

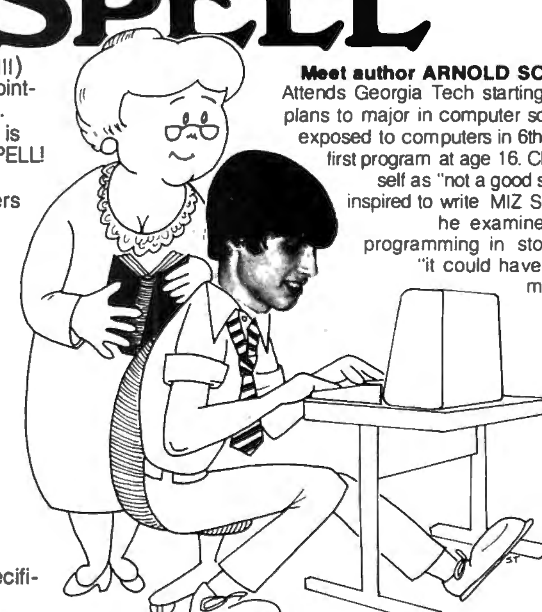
MIZ SPELL permits user configuration of system, including specification of default extensions for original and corrected text file.

Now check the low introductory price: ONLY $49.95

DEALER INQUIRIES INVITED. MIZ SPELL is published by SofPub, Inc. For information, contact Richard Taylor, President, Programs Unlimited at the address or phone numbers shown below.

Meet author ARNOLD SCHAEFFER

Attends Georgia Tech starting Fall, 1981; plans to major in computer science. First exposed to computers in 6th grade; sold first program at age 16. Characterizes self as "not a good speller"; was inspired to write MIZ SPELL when he examined available programming in store, decided "it could have been done much better."



ATTENTION, SOFTWARE AUTHORS. *SofPub Inc. may be the aggressive marketer you've been seeking for your programs. For details, contact Richard Taylor, Programs Unlimited at the address or phone below.*

HOW TO ORDER: Add $1.00 for postage and handling. NY Residents add sales tax.

CALL TOLL FREE: (800) 645-6038 (except NY*)
COMPUTER TO COMPUTER: (516) 334-3134
WRITE: Dept. 881M, Box 265, Jericho, NY 11753

master charge · AMERICAN EXPRESS · VISA

*NY State residents call 516- 997-8668 (Call collect for orders only, please)

FULL GUARANTEE

If for any reason dissatisfied with any purchase from PROGRAMS UNLIMITED, return merchandise in original condition and packaging for prompt, full refund.

✓See List of Advertisers on page 354

After each input byte, the value in DE (initially the start address) is incremented, and the computer looks for more input until the value in DE equals the end address. We cannot allow the values on tape to control the input, however, because the programs would simply be written over each other, always starting at 4200H. By Splice, the starting address is replaced by the old end-of-program value from 406CH. The length of the program is calculated from the values read from the tape and used to calculate the new end-of-program. With that work done, ROM's routine can handle the actual program bytes. One of the CLOAD asterisks is replaced by a plus sign to indicate that a merge is taking place (lines 520–530).

ADJ Subroutine

The ADJ subroutine does for Splice what it did for Split, although there are some differences in the way it was written. Split had a Ready Cassette program as part of its own message buffers, but Splice does not, so the Query message (line 1340) was added to the ADJ subroutine. Also, there is no difference in effect between simply subtracting the transferred program bytes and adding them with a negation of the final sum. As a matter of fact, ROM does it the latter way.

Using assembly language, it would be possible to generalize ADJ for any program you might want to load with the low-core, checksum, modified technique. You could then assemble a version of ADJ, save it on tape and load it after you've finished your assembly language program. This requires using some consistent labels, although the values will vary from one program to the next.

Your main program must define all JP, Call and LD instructions by reference to DIF. It must also provide a certain amount of free space below 4200H for stack operations. Even if your program does not leave the stack there, remember that it is working during the CLOAD itself and if part of your program goes there, it will be changed before your program is ever run. This is the purpose of the string of DEFW 0 lines in Split and Splice.

"In Split and Splice, the high-order byte of the entry point was 41H, but it need not be this value."

Let's assume you regularly label the first byte of the program Head and your entry point Entry. You can then use your labels, plus your knowledge of the constant values like 41FEH, to handle the block-transfer, checksum adjustment and cassette dump. There are certain logical operations which Radio Shack's Editor/Assembler supports which are useful here. The value that eventually goes into 41FEH, the low-order byte of your starting address, can be specified Tail, DEFB, Entry and 0FFH. The label Tail can be used to calculate the displacement, which is DIF, DEFL TAIL—41FEH.

In Split and Splice, the high-order byte of the entry point was 41H, but it need not be this value. To avoid manual assembly on this matter, you can use: HIBYTE DEFL ENTRY<-8, and the value HIBYTE in the check-

Program Listing 2

```
                00010           ;SPLICE
                00020           ;LEVEL I BASIC UTILITY
                00030           ;BY STEPHEN MILLS
                00040           ;ALLOWS STACKING OF SEVERAL BASIC
                00050           ;PROGRAMS FROM CASSETTE.
                00060           ;PROGRAMS  M U S T  BE ENTERED
                00070           ;SO THAT LINE NUMBERS ARE ASCENDING
                00080           ;(THOSE WITH LOWER ## FIRST)
                00090           ;AND EXCLUSIVE.
4A00            00100           ORG     4A00H
                00110           ;NOTE:  THIS EDTASM ASSEMBLER PLACES
                00120           ;PROGRAM AT 4A00H, BUT ON ENTRY IT
                00130           ;BLOCK-MOVES TO 4100H AND CREATES A
                00140           ;250-BAUD LEVEL I LOADABLE TAPE.
0900            00150 DIF       EQU     900H    ;DISPLACEMENT
4100            00160 BAD       EQU     4100H
4A00 0D         00170           DEFB    13      ;CAR RET
4A01 42         00180           DEFM    'BAD'
4104            00190 DISPLA    EQU     $-DIF
4A04 7E         00200 SHOW      LD      A,(HL)  ;GET ASCII
4A05 23         00210           INC     HL      ;AND BUMP
4A06 D7         00220           RST     10H     ;VIDEO DISPLAY
4A07 10FB       00230           DJNZ    SHOW    ;DISPLAY LENGTH
4A09 3E0D       00240           LD      A,0DH   ;IN B, MESSAGE ADDRESS
4A0B D7         00250           RST     10H     ;IN HL
4A0C C9         00260           RET     ;CARG RET AT END
410D            00270 CONTNU    EQU     $-DIF
4A0D 205E       00280           JR      NZ,FAIL ;CHECKSUM TEST
4A0F ED5B6A40   00290           LD      DE,(406AH)      ;MEM TOP
4A13 EB         00300           EX      DE,HL   ;SWAP FOR TEST
4A14 ED52       00310           SBC     HL,DE   ;SEE IF IT FITS
4A16 3855       00320           JR      C,FAIL  ;DON'T TAKE IT
4A18 ED536C40   00330           LD      (406CH),DE      ;UPDATE PROG END
4A1C 310042     00340 RESET     LD      SP,4200H        ;REINIT STACK
4A1F 062A       00350           LD      B,MESSL ;SET MAIN MESSAGE
4A21 217A41     00360           LD      HL,MESG ;GET ADDRESS
4A24 CD0441     00370           CALL    DISPLA  ;SHOW IT
4A27 3A4038     00380 PAUSE     LD      A,(3840H)       ;GET COMMAND MEM
4A2A CB57       00390           BIT     2,A     ;BREAK KEY
4A2C C2C901     00400           JP      NZ,01C9H
4A2F 1F         00410           RRA     ;FIRST BIT=ENTER
4A30 30F5       00420           JR      NC,PAUSE
4A32 210D41     00430 MERGE     LD      HL,CONTNU       ;RET ADDRESS
4A35 E5         00440           PUSH    HL      ;PUSH FOR CALL
4A36 CDE90F     00450           CALL    0FE9H   ;TURN ON CASSETTE
4A39 110042     00460           LD      DE,4200H        ;WHERE BASIC STARTS
4A3C D5         00470           PUSH    DE      ;BALANCE STACK
4A3D AF         00480           XOR     A
4A3E CD810F     00490           CALL    0F81H   ;CALL BYTE SEARCH
4A41 FEA5       00500           CP      0A5H    ;SYNC BYTE?
4A43 20F9       00510           JR      NZ,$-5  ;REDO IF NOT
4A45 3E2B       00520           LD      A,2BH   ;SET + FLASHER
4A47 32003C     00530           LD      (3C00H),A
4A4A 3D         00540           DEC     A       ;ASTERISK NOW
4A4B 32013C     00550           LD      (3C01H),A
4A4E CDA10F     00560           CALL    0FA1H   ;READ
4A51 BA         00570           CP      D       ;SHOULD MATCH
4A52 2019       00580           JR      NZ,FAIL ;QUIT IF NOT
4A54 CDA10F     00590           CALL    0FA1H   ;READ LO BYTE
4A57 BB         00600           CP      E       ;SHOULD BE 00
4A58 2013       00610           JR      NZ,FAIL
4A5A CDA10F     00620           CALL    0FA1H   ;LAST BYTE
4A5D 67         00630           LD      H,A     ;ADDRESS
4A5E CDA10F     00640           CALL    0FA1H   ;READ
4A61 6F         00650           LD      L,A     ;IN
```

sum adjustment calculation SUB HIBYTE.

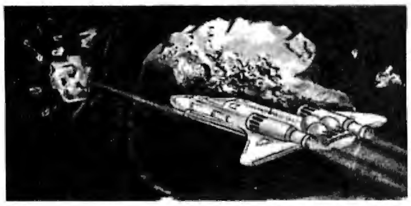
With ADJ you have a valuable tool for other assembly language programs, including other possibilities to facilitate working within the confines of Level I Basic. Other I/O operations possible are adding file name search to your CLOADs, speeding up your load rate or driving a line printer for hard copy dumps of your programs. Other editing aids, like line renumbering and relocation, are also available. It's not quite as easy as typing out a command in Level II, but the programs are so short that they load in an instant.

By the way, if you're wondering what happens if you merge programs so the line numbers are out of sequence, try it and see. You'll get a weird assortment of results. Evidently Level I can only do its sorting and assembling from the keyboard. When Splice loads them in a mixed state, they will simply stack up out of order. Some of them will even run like that. But if you have, say, a line 100 stacked above a line 200, you won't be able to access it by Basic command in most cases.

The search technique used for editing, GOTO and GOSUB expects the lines to be in order and will not proceed beyond the barrier set by an out-of-sequence line number. Also, lower-numbered lines are sometimes deleted after an out-of-sequence merge, but I don't understand the selectivity of this. Split can help you out of such messes, since it will read through a whole program looking for a specified line (but in case there are two or more line 100s, it will go with the first one). You may want to meet the challenge yourself by writing a sorting, line renumberer or additive line adjustment program.■

```
4A62 B7         ØØ66Ø          OR     A        ;RESET CARRY
4A63 ED52       ØØ67Ø          SBC    HL,DE    ;PROGRAM LENGTH
4A65 ED5B6C4Ø   ØØ68Ø          LD     DE,(4Ø6CH)        ;CURRENT END
4A69 19         ØØ69Ø          ADD    HL,DE    ;CALC NEW END
4A6A D218ØF     ØØ7ØØ          JP     NC,ØF18H          ;FINISH IN ROM
4A6D CDE4ØF     ØØ71Ø FAIL     CALL   ØFE4H    ;TURN OFF CASSETTE
4A7Ø Ø6Ø4       ØØ72Ø          LD     B,4      ;MESS LENGTH
4A72 21ØØ41     ØØ73Ø          LD     HL,BAD   ;MESSAGE
4A75 CDØ441     ØØ74Ø          CALL   DISPLA   ;PROGRAM LENGTH
4A78 18A2       ØØ75Ø          JR     RESET    ;NOT INCREASED
417A            ØØ76Ø MESG     EQU    $-DIF
4A7A ØD         ØØ77Ø MAINMS   DEFB   ØDH      ;INSTRUCTION
4A7B 5Ø         ØØ78Ø          DEFM   'PRESS:'
4A81 ØD         ØØ79Ø          DEFB   ØDH      ;DISPLAY
4A82 3C         ØØ8ØØ          DEFM   '<ENTER> TO MERGE
4A92 ØD         ØØ81Ø          DEFB   ØDH      ;CAR RET
4A93 3C         ØØ82Ø          DEFM   '<BREAK> WHEN DONE'
ØØ2A            ØØ83Ø MESSL    EQU    $-MAINMS
4AA4 ØØ         ØØ84Ø          NOP
41A5            ØØ85Ø HEAD     EQU    $-DIF
4AA5 ØC         ØØ86Ø HIHEAD   DEFB   ØCH      ;CLEAR SCREEN
4AA6 ØD         ØØ87Ø          DEFB   13       ;SPACE FOR FLASHER
4AA7 4C         ØØ88Ø          DEFM   'LEVEL I MERGE PROGRAM'
4ABC ØD         ØØ89Ø          DEFB   ØDH
4ABD 4C         ØØ9ØØ          DEFM   'LINE NUMBERS MUST BE SUCCESSIVE'
ØØ37            ØØ91Ø HEDL     EQU    $-HIHEAD
4ADC ØØ         ØØ92Ø          NOP
41DD            ØØ93Ø START    EQU    $-DIF
4ADD C2ECØE     ØØ94Ø          JP     NZ,ØEECH
4AEØ 31ØØ42     ØØ95Ø          LD     SP,42ØØH
4AE3 Ø637       ØØ96Ø          LD     B,HEDL   ;MESSG LENGTH
4AE5 21A541     ØØ97Ø          LD     HL,HEAD
4AE8 CDØ441     ØØ98Ø          CALL   DISPLA
4AEB C31C41     ØØ99Ø          JP     RESET-DIF
4AEE ØØØØ       Ø1ØØØ CKSUM    DEFW   Ø        ;CHECKSUM ADJ
4AFØ ØØØØ       Ø1Ø1Ø          DEFW   Ø        ;STACK
4AF2 ØØØØ       Ø1Ø2Ø          DEFW   Ø
4AF4 ØØØØ       Ø1Ø3Ø          DEFW   Ø
4AF6 ØØØØ       Ø1Ø4Ø          DEFW   Ø
4AF8 ØØØØ       Ø1Ø5Ø          DEFW   Ø
4AFA ØØØØ       Ø1Ø6Ø          DEFW   Ø
4AFC ØØØØ       Ø1Ø7Ø          DEFW   Ø        ;AREA
4AFE DD         Ø1Ø8Ø TAIL     DEFB   START-41ØØH       ;LOW ADR
4AFF 31FF4A     Ø1Ø9Ø ADJ      LD     SP,$     ;DON'T WANT STACK IN
4BØ2 11FE41     Ø11ØØ          LD     DE,41FEH          ;TRANSFER ZONE
4BØ5 21FE4A     Ø111Ø          LD     HL,TAIL
4BØ8 Ø1FFØØ     Ø112Ø          LD     BC,ADJ-4AØØH
4BØB AF         Ø113Ø          XOR    A        ;CLEAR A
4BØC 86         Ø114Ø BLOCK    ADD    A,(HL)   ;CHECKSUM
4BØD EDA8       Ø115Ø          LDD    ;BLOCK MOVE
4BØF EAØC4B     Ø116Ø          JP     PE,BLOCK
4B12 ED44       Ø117Ø          NEG    ;NEGATE A
4B14 D641       Ø118Ø          SUB    41H      ;MODIFY CHECKSUM
4B16 32EE41     Ø119Ø          LD     (CKSUM-DIF),A
4B19 213C4B     Ø12ØØ TLP      LD     HL,QUERY          ;ASK FOR
4B1C Ø6ØD       Ø121Ø          LD     B,13     ;25Ø BAUD COPIES
4B1E CDØ44A     Ø122Ø          CALL   SHOW     ;ASK ABOUT COPY
4B21 CD4ØØB     Ø123Ø QWAIT    CALL   ØB4ØH    ;KEYSCAN
4B24 FEØ3       Ø124Ø          CP     3        ;BREAK?
4B26 CAC9Ø1     Ø125Ø          JP     Z,Ø1C9H  ;BACK TO BASIC
4B29 FEØD       Ø126Ø          CP     13       ;ENTER?
4B2B 2ØF4       Ø127Ø          JR     NZ,QWAIT
4B2D CDE9ØF     Ø128Ø          CALL   ØFE9H
4B3Ø 21ØØ41     Ø129Ø          LD     HL,41ØØH
4B33 11FF41     Ø13ØØ          LD     DE,41FFH
4B36 CD4BØF     Ø131Ø          CALL   ØF4BH
4B39 AF         Ø132Ø          XOR    A        ;SET Z FLAG
4B3A 18DD       Ø133Ø          JR     TLP      ;LOOP FOR MORE COPIES
4B3C 43         Ø134Ø QUERY    DEFM   'CSAVE SPLICE?'
4AFF            Ø1999          END    ADJ
```

THE PROGRAM STORE



SPACE ROCKS

By Steven Kearns from Acorn
Gigantic antimatter rocks appear on the Tactical Display Screen of your spacecraft. You blast away with lasers and they just explode into smaller chunks. To score in this fast arcade game with sound, you must destroy the rocks. To stay in the game at all, you must avoid them!

To add to your woes, time bombs appear periodically. If their timers reach zero -- BOOM! And if that's not enough, the aliens will be glad to send out some spaceships loaded with antimatter torpedoes. Fire thrusters to move, shoot laser cannon, jump to hyperspace -- anything to avoid the onslaught. One or two players can compete, with five levels of difficulty.

16K protected tape...$19.95
32K protected disk...$19.95

GOLFER'S CHALLENGE

By Charles Butler from Liberty
Take on the formidable "Liberty Links" golf course by yourself or with a friend. The course changes each time you play, offering 18 holes replete with doglegs, trees, water hazards, sand traps, and out of bounds areas. You choose your club and the direction you wish the ball to go, but watch out for the breeze or a poor hit -- it can play havoc with your shot!

16K tape...$14.95 32K disk...$18.95

ALIEN ARMADA

By Waldron Hodsdon from Liberty
Hmmm. Looks like another "Space Invaders" type game. Nice neat ranks of aliens poised over your defensive base . . . but WATCH OUT! Here they come, swooping down with their bombs and Kamakazi-like dives. There are individual attackers plus group flights -- all intent on destroying your three bases before you destroy them. ALIEN ARMADA allows up to two players and has three levels of difficulty from beginner to expert.

16K tape...$14.95 32K disk...$18.95

TENPINS

By John Allen from Acorn
TENPINS brings you all the thrills of championship bowling. Up to four players participate, and the program automatically senses the skill of each. Beginners can simply position the ball and "roll" it while more skilled players can vary the force, roll a curve, and cause it to spin as it heads for the pins. All this -- plus 3-D graphics and sound effects -- adds up to a realistic and thoroughly challenging bowling game.

Protected tape...$14.95
Protected disk...$20.95

StarFighter

By Sparky Starks from Adventure Int.
As mercenary and galactic police officer, you must maintain the condition and control of all parts of your spacecraft. You sit at the controls while peering out of the digital spaceview port. Suddenly something appears on your screen: is it a Starpirate or a friendly merchant ship? You can't tell yet, and at this speed you may have only a fraction of a second to make an attack/no attack decision.

Speed and sound effects enhance the graphic action of this tactical game. Can you take the pressure?

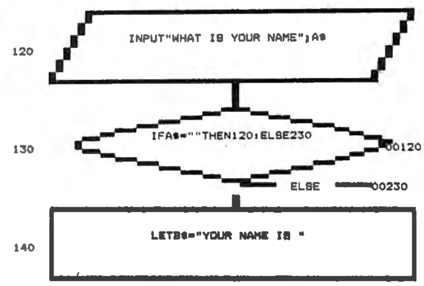
Model I & III, 16K tape...$24.95
Model I disk version......$29.95

HIGH FINANCE

By Leonard Kaufer from Cybernautics
It's almost literally "dog-eat-dog" in this manufacturing simulation. You and from one to five opponents test wits against each other and the market as competitors in the dog food business. You must buy raw materials at a good price to be able to sell at a profit. Sneaky moves by you or your opponents can drive up the cost of needed materials or depress the prices of finished products. You will be amazed at how a friendly game can turn into a cutthroat showdown of wills and wiles.

16K tape...$14.95

THE DOCUMENTER



From P80NUT Software
If you would love to be able to document your programs with a flowchart but lack the time, talent, or inclination, this program is for you. THE DOCUMENTER will produce a logical flowchart directly from any suitable BASIC program and print it on the screen or most lineprinters.

You get a flowchart and branch map that will help you follow program flow and aid your debugging efforts. Even memory-filling programs can be broken down into segments and flowcharted.

16K tape...$19.95 32K tape...$19.95
48K tape...$19.95 48K disk...$29.95

COLOR COMPUTER

The Program Store stocks a selection of programs for the TRS-80 Color Computer, too! For more information, send 50 cents for our Color Computer mini-catalog.

(CCT: Color Computer Tape
ECB: Extended Color Basic)

MINI-ARCADE 1: Computer Simulations
Five arcade games. 16K tape...$14.95

COLOR SPACE INVADERS: Spectral Assoc.
Machine language action. CCT...$21.95

COLOR METEOROIDS: Spectral Assoc.
Destroy deadly meteoroids. CCT...$21.95

COLOR SPACE WAR: Spectral Assoc.
Blow up the Death Star. CCT...$21.95

GAME OF LIFE: Spectral Assoc.
Classic rules of generation. CCT...$19.95

BLOCKADE: INTERPRO
Joystick arcade game. CCT/16K ECB..$19.95

GRAPHIC AIDE: Computer Simulations
Master illustration program. 16K...$10.95

TYPEWRITER: Computer Simulations
Simple word processing. 16K...$14.95

COCOBUG: Allen Gelder Software
Debugging monitor. 16K ECB...$19.95

SCREEN EDIT CONTROL SYSTEM: Datasoft
Screen editor & much more. 16K...$29.95

SIGMON: Datasoft
6809 monitor containing mini-assembler, disassembler and debugger. CCT..$29.95

COLOR COMPUTER DISASSEMBLER
Access to the ROM. 16K ECB...$19.95

MAGIC BOX: Spectral Assoc.
Use Model I/III BASIC programs. CCT $24.95

Visit Our Baltimore, MD Store: W. Bell Plaza – 6600 Security Blvd

TO ORDER CALL TOLL FREE 800 424-2738 For information Call (202) 363-9797

THE PROGRAM STORE ✓17
4200 Wisconsin Avenue NW, Dept. E09 Box 9609
Washington, D.C. 20016

MAIL ORDERS: Send check or M.O. for total purchase price, plus $1.00 postage & handling. D.C. residents, add 6% tax. Charge card customers: include all embossed information on card.

See List of Advertisers on page 354

A memory saving technique for Level II owners.

Delete and Save

R. Gene Langston
10755 Liscard Rd. S.
Jacksonville, FL 32216

After running some program instructions, I wanted to delete the block of statements that housed the instructions, enabling me to have more memory in which to load my arrays. I included, at the end of the instruction statements, a DELETE XXXX-XXXX line. Sure enough, as the program ran, it deleted the appropriate line numbers, but it also fell back into the command mode and I got a Ready prompt. I knew I could delete my line numbers and still stay in the program, but first I had to find out how the computer did it.

128	END	171	LSET	214	<
129	FOR	172	RSET	215	SGN
130	RESET	173	SAVE	216	INT
131	SET	174	SYSTEM	217	ABS
132	CLS	175	LPRINT	218	FRE
133	CMD	176	DEF	219	INP
134	RANDOM	177	POKE	220	POS
135	NEXT	178	PRINT	221	SQR
136	DATA	179	CONT	222	RND
137	INPUT	180	LIST	223	LOG
138	DIM	181	LLIST	224	EXP
139	READ	182	DELETE	225	COS
140	LET	183	AUTO	226	SIN
141	GOTO	184	CLEAR	227	TAN
142	RUN	185	CLOAD	228	ATN
143	IF	186	CSAVE	229	PEEK
144	RESTORE	187	NEW	230	CVI
145	GOSUB	188	TAB(	231	CVS
146	RETURN	189	TO	232	CVD
147	REM	190	FN	233	EOF
148	STOP	191	USING	234	LOC
149	ELSE	192	VARPTR	235	LOF
150	TRON	193	USR	236	MKI$
151	TROFF	194	FRL	237	MKS
152	DEFSTR	195	ERR	238	MKD
153	DEFINT	196	STRING$	239	CINT
154	DEFSNG	197	INSTR	240	CSNG
155	DEFDBL	198	POINT	241	CDBL
156	LINE	199	TIME$	242	FIX
157	EDIT	200	MEM	243	LEN
158	ERROR	201	INKEY$	244	STR$
159	RESUME	202	THEN	245	VAL
160	OUT	203	NOT	246	ASC
161	ON	204	STEP	247	CHR$
162	OPEN	205	+	248	LEFT$
163	FIELD	206	−	249	RIGHT$
164	GET	207	*	250	MID$
165	PUT	208	/	251	
166	CLOSE	209	↑	252	
167	LOAD	210	AND	253	.
168	MERGE	211	OR	254	!
169	NAME	212	>	255	ISA
170	KILL	213	=		

Table 1.

Here is some background on the TRS-80 statement format.

Go to your keyboard and type New. Now enter:

```
100 PRINT
```

Now, PEEK at the addresses that hold this statement. Level II memory stores the first address of the program at 17129. After entering the PEEK routine listed, you should get this result:

```
FOR X = 17129 TO 17136:PRINT X;" =";
PEEK(X):NEXT
          17129 = 239
          17130 =  66
          17131 = 100
          17132 =   0
          17133 = 178
          17134 =   0
          17135 =   0
          17136 =   0
```

The Next Statement

In Level II Basic the 17129 and 17130 addresses point to the beginning of the next statement. The first section (239) is the Least Significant Byte (LSB) and the second (66) is the Most Significant Byte (MSB). To convert this to something we can identify with, multiply the MSB by 256 and add the LSB. For example: (256*66) + 239 = 17135. This means that the next statement (if one is there) will begin at 17135.

The statement line number makes up the next two addresses, 17131 and 17132. The MSB and LSB are computed the same way as before. Line numbers up to 255 are obvious, and the rest are computed as shown.

Ignoring the next address for a moment, we find that 17134 holds a zero. This is used as an end-of-line indicator. The last two addresses, 17135 and 17136, are the locations (as computed above) where the next line would start if there was one. As you can see, this is the last statement in our program and these pointers are set to zero. This indicates that there are no more statements to follow. If there was another line, these two addresses would hold the pointer for the next statement, as did 17129 and 17130.

All of the addresses between the line number and the end of line indicator contain the text of the statement. Usually, there will be many more than the single one in our example. The text is composed of ASCII codes (see the Level II reference manual), and Level II statement/function storage format (see Table 1). Line 17133 contains the code for Print 178.

Suppose our line had been: 100 ONZGOTO200. Then our PEEKing would have revealed:

```
17129 = 244
17130 =  66
17131 = 100
17132 =   0
17133 = 161
17134 =  90
17135 = 141
17136 =  50
17137 =  48
17138 =  48
```

```
17139 =    0
17140 =    0
17141 =    0
```

I discovered that at least three things happen when the delete command is executed. First, the pointer at memory addresses 16633 and 16634 is assigned a new number. These pointers signal the end of a Basic program and the start of variable storage. MSB and LSB are computed as before. Second, the next statement pointers (of the first line deleted) are reset to zero. And third, the statement line number (of the first line deleted) is assigned a 4.

Now we must find the specific addresses that we wish to delete.

Let's assume that the instructions of a particular program are located in the last 60 lines, starting with line number 15000 and ending with 15600. The line preceding 15000 is: 14950 GOTO 100. To delete lines 15000 to 15600 (end of program) after displaying the information, we need to know the address of 15000. Edit 14950 as follows:

```
14950 GOTO 100%
```

Notice that a single character is added. You must use a character that has not been utilized in the program because we will search the memory for the ASCII code of this character. If it is a unique code, the search will be that much easier. Enter this command:

```
FOR X = 17129 TO 32767:IF PEEK(X) = 37
PRINT X ELSE NEXT
```

After a value is printed, verify that this is indeed the address that you want. If, for example, the address printed is 20248, verify this by Print PEEK (20247) and the value should be 48, the ASCII code for the zero that precedes it (GOTO 100%). If it is not verifiable, continue with your search until the character is found.

Once you have found the correct address, make a note of this address, A. Now delete the character you added in the line. Address A is now equal to the end-of-line indicator.

The next four addresses are the ones that we require for deletion.

To find the value of the address that we must set into 16633 and 16634, evaluate the pointers as follows:

- Divide address A+3 by 256. (A+3 is the address where the line number to be deleted begins.)
- Take the integer value of this division, B.
- Compute address A+3 minus 256*B. Let this value equal C.

Now we have address values of 'A', 'B' and 'C'.

Let's look at part of our example program and see what we can do with these values:

```
10    GOSUB  15000
15000  REM   INSTRUCTIONS BELOW
15600  RETURN
```

In order to delete lines 15000 to 15600, put the following line after line 15000 in any position where it will be executed:

```
POKE 16633,C:POKE 16634,B:POKE A + 1,
0:POKE A + 2,0POKE A + 3,4:POKE
A + 4,0:CLEAR 50
```

Put the values you found in place of A, B and C. The clear statement must be executed, but you can put in a value as required by your program.

These POKEs can be executed and all statements following them will be executed just as if everything was normal. After running the program, listing it will show that the lines have been deleted.

In summary:

- Add an unused character to the end of the last line not be be deleted.
- Search memory for that character; verify.
- Delete unused character.
- Note address A.
- Evaluate pointer (16633, 16634) line.
- Note values B, C.
- Add these executable statements after last line to be saved:

```
POKE 16633, C
POKE 16634, B
POKE A + 1, 0
POKE A + 2, 0
POKE A + 3, 4
POKE A + 4, 0
CLEAR 50
```
■

✓See List of Advertisers on page 354

The principles of operation at the user-programmer interface.

Going Pro

Nicolas Moss
108 Via Hermosa
Santa Cruz, CA 95060

With the introduction of the affordable personal computer, many people who never dreamed of seeing a computer are now programming. Several are very good at it. Many of the younger self-taught computerists decide to write software commercially. The ubiquity of the TRS-80 has created a big market for Basic programs, along with modifications to existing software (for the user who doesn't program). Very powerful software can be written at home for about $800. The sum makes part-time programming very appealing.

The purpose of this article is to tell you, the future part-timer, what to expect when you program for someone else.

The Rules

When you program for yourself, you know exactly what you want. You understand what you're doing because you have some background in the subject. You also know the limits of the machine. This, of course, changes when you write a program for someone else.

First, you do not really know what the user wants. How could you? Therefore, rule number 1: The user does not know what he wants.

Second, you will probably not know what the user is talking about: Computers have yet to be programmed to understand buzzwords. Rule number 2: When the user has a vague idea of what he wants, research the subject. This allows you to know what he means when he says, "Could you cross-reference my G.L. with my A.R. and A.P., and can you have it generate a categorized P. & L. sheet?"

Third, do not answer the user's buzzwords with your own. "Sure, I can do it, if your floppy can support file chaining and D.M.A., otherwise the interpreter will tie up the C.P.U. when it does string clean-up in high RAM." Rule 3: If you have to use buzzwords, use small, logical words that will not confuse the issue.

Rules 2 and 3 pose the biggest problem in the small-computer field. When a customer wants reasonably priced software, he shouldn't have to hire a third party to translate the conversation. The programmer must communicate properly with the user!

Rule 4 is derived from Rule 1: Application programs evolve. The first copy of a program will have to be changed radically to get the final copy. In the first program, avoid fancy tricks and leave plenty of room for modifications. Make the code easy to change.

Since the user doesn't really know what he wants, and is probably very non-technical, you will usually get a negative feeling about him. I therefore have Rule 5: The user is the boss; never criticize the boss! To help you cope with Rule 5, I suggest reading *Zen and the Art of Motorcycle Maintenance* by Robert M. Pirsig.

Although these rules sound simple enough, they are the hardest damn things to live by! It takes discipline and concentration.

So, for your first out-of-field programming experience, I suggest you choose to write a program for an accessible relative. This way, the user is close, and it is very hard to break Rule 5 if you are working for a relative; and if you do break it, affectionate understanding will keep the programmer-user bond from breaking.

Two last rules. After you satisfy the user, he will think you are a god. Rule 6: Programmers are not God! Rule 7: Under any circumstances, do not forget Rule 6! ■

Alone in your spacecraft, you are exploring an uncharted area when your Tactical Display Screen suddenly blossoms with gigantic moving objects. It would be bad enough if they were ordinary asteroids, but their size and behavior tell you they are much worse—huge chunks of deadly antimatter, capable of destroying your ship on contact!

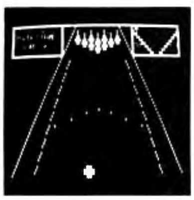
You fire your thrusters, hoping to escape the menacing onslaught, but here come more of the sinister shapes. You are surrounded, and the only way to avoid destruction is to fire your laser cannon. Aim . . . Fire . . . A direct hit! But instead of destroying the rock, your blast has exploded it into several smaller—but equally lethal—pieces. Now each must be separately eliminated.

The thought occurs to you that the sudden appearance of these rocks might be more than coincidence, and your suspicions are confirmed as time bombs enter the Tactical Display's field, their timer displays counting down toward your destruction! And finally the enemies themselves, in space ships firing antimatter torpedoes, join the fray.

Can you destroy or avoid the rocks and the enemy, and at the same time still the incessant ticking of the time bombs? With SPACE ROCKS, you can try. It's a realtime arcade game with sound effects to enhance the action. One or two players can compete, and there are five levels of difficulty to keep up the challenge as your skills increase.

Order SPACE ROCKS today for your TRS-80* model I or III. Available on 16K tape or 32K disk versions for just $19.95.

ACORN PROGRAMS ARE AVAILABLE AT FINE COMPUTER STORES EVERYWHERE



TENPINS

By John Allen

A realtime action game with 3-D graphics and sound, TENPINS brings the thrill of championship bowling to your TRS-80*

From one to four players participate, and the program senses the skill of each player. Beginners can simply "roll" the ball while more skilled players can vary the ball's force, direction, and spin.

TENPINS even has factors to simulate imperfections in the lanes, adding even more realism. Available for only $14.95 for 16K tape version, $20.95 for 32K disk.

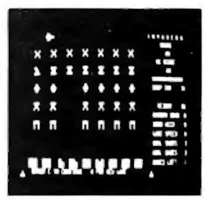
PINBALL

By John Allen

Get your flipper fingers ready for action in this realtime, machine language game.

Lots of sound and flashing graphics make this fast action game so much like the real thing that you'll have to remind yourself not to shake your TRS-80. Choose from five playing speeds to match your skill. Can you beat your friends' scores? Will you avoid the infamous "Bermuda Square?" Get PINBALL today and find out.

For TRS-80* model I or III. Just $14.95 for the 16K tape version or $20.95 on disk (32K).



INVADERS FROM SPACE

By Carl Miller

A fast machine language approach to this classic (and addictive) space game. The aliens drop bombs, move around, and try to overrun your bases.

You choose the speed, enemy bomb frequency and accuracy, and how many shots and bases you have. Unlike other games of this type, you can move your base and simultaneously fire at the invaders. Fun for all ages and skill levels, it has full sound effects for even more excitement.

Available for TRS-80* model I or III. Only $14.95 on 16K tape version or $20.95 on 32K disk.

*Trademark of Tandy Corp.

(202) 544-4259

Acorn
Software Products, Inc.

634 North Carolina Avenue, S.E., Washington, D.C. 20003

✓34

ALL ACORN ENTERTAINMENT SOFTWARE SUPPLIED FOR TRS-80* MODELS I & III

MAIL ORDERS: Include $2.00 shipping & handling (D.C. residents add 6% sales tax).
CHARGE CARD CUSTOMERS: Call (202) 544-4259 for fastest service.

✓See List of Advertisers on page 354

This mechanical horse race is heavy on positive reinforcement.

Math Derby

Michael A. Duffin
1507 East Ave.
Berwyn, IL 60402

One assignment I gave my introductory data processing students was to design and implement an educational game. Two students (Kay Keyes and Kathleen Schmaus) decided to write math games.

Kathy's game used the traditional "happy face" for a correct answer and "sad face" for an incorrect answer. The user specifies whether he or she wishes to add, subract, multiply or divide. To construct her figures, Kathy used the PRINT@ instruction. Kay's game, on the other hand, concentrated on the graphic abilities of the TRS-80. Kay graphically constructed a race track and horse using the POKE and PRINT@ instructions. In her game, if the user responds correctly to an addition problem, the horse moves to the finish line.

Learning from Students

Combining the ideas of these two students and implementing a few of my own, I constructed a viable Horse Race game that requires a student to be quick and accurate in answering a problem. It is an educational game designed to make math more interesting. It was written on a Level II TRS-80, and I believe it will work on a Level I machine as well, without any modification.

One to three players may play this game. A math problem is presented and each player has a horse on the screen as a playing piece. A player's horse moves only when the math problem is answered correctly. The player may pick the type of math operation he wishes (i.e., add, subtract, multiply or divide). A correct answer causes the horse to move, and the length of the move is determined by the type of operation chosen.

The Program

The most interesting routine in the program exists in lines 5120 through 5150. This routine lets the player enter his or her answer to a math problem via the INKEY$ instruction. One advantage of using this routine is that it allows the user to enter an answer that is more than one digit long. It does this by using an array, the VAL instruction in conjunction with the INKEY$ instruction, and concatenation.

Another advantage that this routine has is that it times the player's response. If I had used the Input instruction instead, the player could take out a calculator or take a short nap or a trip to Florida before he or she would have to respond to the problem presented by the program. The INKEY$ routine gives the player about 15 to 20 seconds to respond to the problem.

The best way to explain how the routine works is by using an example. Assume that the player has said that he or she would like a division problem and that the computer has then presented the following problem: 8/3 = ______. The answer is 2.67 rounded to two decimal places.

When the routine at lines 5120 through 5150 is entered, the first thing it does is initialize the J index to one. Line 5128 sets B$(1) to the first number that has been entered. If no number has been entered (B$(1) = " "), then J is not incremented and the program waits a bit and then checks again.

In our example when a digit is entered, B$(1) = 2 if a correct answer is being entered. At this time the condition at line 5128 is false, so the program falls

Program Listing 1. Horse Race (Math Game)—Duffin.

```
1 REM ****************************************************
2 REM *
3 REM *  '    HORSE RACE      (MATH GAME)
4 REM *
5 REM *  AUTHOR         MIKE DUFFIN
6 REM *
7 REM * (INSPIRED BY THE PROGRAMS WRITTEN BY KAY KEYES AND
KATHLEEN SCHMAUS AT MORTON COLLEGE IN BERWYN,ILLINOIS FOR
DPR 101...AN INTRODUCTION TO DATA PROCESSING.)
8 REM *
9 REM *******************************************************
10 REM *** A LIST OF VARIABLES APPEARS AT THE END OF THE PROGRAM
```

Program continues

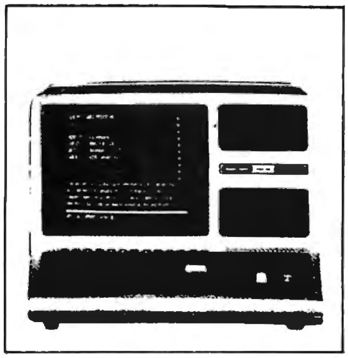
INTRODUCING:

SPECTRONICS, INC.

(A Wholly Owned Subsidiary of Level IV Products)

- OFFERING -

The Full TRS-80* Line at Discount Prices, and Extended Warranty Coverage



COLOR COMPUTER	
16K (R)	$440
16K (E)	$359
16K w/ext basic (R)	$529
16K w/ext basic (E)	$479

MOD III COMPUTER	
16K (R)	$859
48K 2-40 drives (R)	$2010
48K 2-40 drives (E)	$1859
48K 2-80 drives (E)	$2119
RS232 interface add	$89

MOD II COMPUTER	
32K (R)	$2950
64K (R)	$3375
Expansion Dr#1 (R)	$999
Expansion Dr#1-2-3 (E)	$695

NOTE: THE MOD III CAN BE ASSEMBLED WITH UP TO 4 MEG UNFORMATTED FLOPPY DISK STORAGE CAPACITY (4-80 TRK DOUBLE SIDED DRIVES) – CALL FOR A QUOTE ON THE SYSTEM TAILORED TO YOUR EXACT NEEDS.

PRINTERS	
Line Printer IV (R)	$849
Line Printer VI (R)	$986
Daisy Wheel II (R)	$1679
C. ITOH Starwriters	Call
Centronics All	Call
Epson All	Call
Anadex All	Call

DRIVES	COMPLETE	RAW
40 TRK MPI (E)	$309	$245
80 TRK MPI (E)	$429	$360
40/40 TRK MPI (E)	$445	$375
80/80 TRK MPI (E)	$579	$515

PERIPHERALS	
E.I. OK (R)	$249
E.I. 16K (R)	$359
E.I. 32K (R)	$469
E.I. 16K (E)	$305
E.I. 32K (E)	$362
16K RAM (200MS)	$ 39

DEALER'S INQUIRIES INVITED, CALL FOR OUR WHOLESALE DEPARTMENT (313) 525-6201

WARRANTY POLICY

Items including non-Radio Shack components (E) are covered by an extended six-month parts/labor warranty in our tech service center.

Items constructed from Radio Shack approved components (R) are covered by a 90-day parts/labor warranty at any Radio Shack store, plus an additional 90 days in our tech service center.

TERMS OF SALE

2% Discount allowed on cash pre-paid orders. Personal checks require two weeks for bank clearance.

Visa, Master Charge and C.O.D. orders accepted. Add $1.50 for C.O.D.'s.

Shipping/handling/insurance add 1.5% or $2.50 minimum.

Prompt shipments from stock.

SPECTRONICS, INC.
32461 SCHOOLCRAFT ROAD
LIVONIA, MICHIGAN 48150

MICHIGAN – (313) 525-6200
TOLL FREE – 800-521-3305

LEVEL IV PRODUCTS

*TRADEMARK OF TANDY CORP.

See List of Advertisers on page 354

through to the next line, which increments the J index (J = 2). This process repeats until the entire answer is entered or time runs out. Assuming the correct answer is entered within the allotted time, then:

B$(1) = "2"
B$(2) = "."
B$(3) = "6" and
B$(4) = "7"

Since these variables are interpreted by the computer as individual strings rather than a single numeric answer, we must reformat these responses. Line 5150 accomplishes this. The VAL instruction changes a string character to a numeric digit. The plus signs in the instruction concatenate the individual digits to a single answer.

The routine controls the amount of time a player has to enter his or her response through the variables F,H and M, which are used in the three-layer For...Next loops in this routine. If an extremely fast typist is playing this game, you might wish to decrease the value of H at line 5140 and increase one or both the outer loops at lines 5122 and 5124.

The routine at lines 4216 and 4217 clears the response from a previous question.

The instructions to the game are in lines 7000 through 7300. Once you are familiar with the game, instructions are no longer necessary. The only unusual thing about these routines is the way one exits from them. Lines 7038 and 7220 allow the instructions to stay on the screen until a player has finished reading them. The player then presses any key to see the next set of instructions. This is again accomplished via the INKEY$ instruction.

Below the instructions variables and their uses are listed. If you intend to modify the program the list will prove very useful.

The track is drawn by lines 90 through 4010. There is no particular reason that both PRINT@ instructions and POKE instruc-

Program continued

```
 ***
11 CLS
12 P(1)=328 : P(2)=P(1)+192 : P(3)=P(2)+192
13 INPUT "DO YOU NEED INSTRUCTIONS";A$
14 IF A$="Y" THEN GOSUB 7000 ELSE GOSUB 7230
20 PRINT@ 63 + 25, "HORSE   RACE"
90 FOR Y=15360+195 TO 15360+835 STEP 64
100    POKE Y,191
110 NEXT Y
990 REM BOTTOM LINE
999 FOR T=832TO895
1000 PRINT@T,CHR$(141);
1001 NEXT T
1002 FOR E=128+64 TO 191+64
1003 PRINT@E,CHR$(156);
1004 NEXT E
2020 FOR Y=15360+121+194 TO 15360+637+194 STEP 64
2030    POKE Y,191
2040 NEXT Y
4000 PRINT@ 896,"START"
4010 PRINT@895+56,"FINISH"
4090 REM *** HORSES ***
4100 FOR V=1 TO B
4110    GOSUB 6000
4120 NEXT V
4200          FOR V=1 TO B
4204 X=RND(10):Y=RND(10)
4205 REM *** CLEARS LINE ***
4206 PRINT@ 0,"
      "
4210 PRINT@ 0, "DO YOU WISH TO ADD, SUBTRACT ,MULTIPLY OR DIVIDE
! "N$(V):INPUT A$
4212 PRINT@ 0,"
      "
4214 REM CLEARS OLD ANSWER'
4216 FOR J=1 TO 6
4217    B$(J)=""
4218 NEXT J
4219 REM *** GOES TO APPROPRIATE ROUTINE:: ADD,SUB,MULT,DIV ***
4220 IF LEFT$(A$,1) = "S" THEN GOTO 4370
4330 IF LEFT$(A$,1) = "M" THEN GOTO 4500
4340 IF LEFT$(A$,1) = "D" THEN GOTO 4400
4350 IF LEFT$(A$,1) = "A" THEN GOTO 5100
4360 GOTO 4210
4370 PRINT@ 0, X"-"Y "="
4375 Z=X-Y
4380 GOTO 5115
4400 PRINT@ 0, X"/"Y "="
4410 Q=X/Y
4420 Z=INT(Q*100+.5)/100
4430 GOTO 5115
4500   PRINT@ 0, X"*"Y "="
4510 Z=X*Y
4520 GOTO 5115
5100 PRINT@ 0, X"+"Y"="
5110 Z=X+Y
5115 PRINT@ 64,"WHAT IS THE ANSWER "N$(V)
5118 REM ***** THE NESTED FOR NEXT LOOPS THAT FOLLOW ALLOW THE U
SER TO ENTER THE COMPLETE ANSWER VIA THE INKEY$ INSTRUCTION WITH
IN A SPECIFIED AMOUNT OF TIME *****
5119 REM ***** THE AMOUNT OF TIME MAY BE INCREASED BY INCREASING
 THE LIMITS OF ONE OR MORE OF THE FOR STATEMENTS *****
5120 J=1
5122 FOR F=1 TO 10
5124     FOR M=1 TO 10
5126         B$(J)=INKEY$
5128         IF B$(J)="" THEN GOTO 5140
5130             J=J+1
5140             FOR H=1 TO 10
5142             NEXT H
5144      NEXT M
5146 NEXT F
5148 REM *** D IS THE NUMERIC EQUIVALENT OF THE CONCATENATION OF
 THE INKEY$ VALUES .(D IS THE ANSWER SUPPLIED BY THE USER)
5150 D=VAL(B$(1)+B$(2)+B$(3)+B$(4)+B$(5))
5200 IF Z=D THEN GOTO 5300
5210 PRINT@ 0,D" IS THE WRONG ANSWER "N$(V)
5220 PRINT@ 64,"THE CORRECT ANSWER IS "Z
5222 FOR H=1 TO 550:NEXT H
5230 GOTO 5900
5300 PRINT@ 0,Z" IS THE CORRECT ANSWER  THATS VERY GOOD "N$(V)
5310 PRINT "VERY GOOD"
5311 FOR H=1 TO 550:NEXT H
5312 G=1
5320 IF LEFT$(A$,1)="M" THEN G=2
5330 IF LEFT$(A$,1)="D" THEN G=4
5390 REM *** MOVES THE APPROPRIATE HORSE 1 SPACE FOR AN ADDITION
 PROBLEM 2 SPACES FOR A MULT PROBLEM AND 4 SPACES  FOR A DIVISIO
N PROBLEM IF THE USERS ANSWER IS CORRECT ***
5400 FOR P= 1 TO G
5410     GOSUB 6000
5420 NEXT P
5900 NEXT V
5902 GOTO 4200
5910 END
5990 REM *** ERASES AND DRAWS THE HORSES ***
6000 PRINT@ P(V),CHR$(32);
6010 PRINT@ P(V)+1,CHR$(32);
6020 PRINT@ P(V)+2,CHR$(32);
6030 PRINT@ P(V)+60,CHR$(32);
6040 PRINT@ P(V)+61,CHR$(32);
6050 PRINT@ P(V)+62,CHR$(32);
6060 PRINT@ P(V)+63,CHR$(32);
6070 PRINT@ P(V)+64,CHR$(32);
6072 P(V)=P(V)+1
6080 PRINT@ P(V),CHR$(170);
6090 PRINT@ P(V)+1,CHR$(140);
6100 PRINT@ P(V)+2,CHR$(132);
6110 PRINT@ P(V)+60,CHR$(160);
6120 PRINT@ P(V)+61,CHR$(134);
6130 PRINT@ P(V)+62,CHR$(151);
6140 PRINT@ P(V)+63,CHR$(131);
6150 PRINT@ P(V)+64,CHR$(171);
6152 IF P(1)>376 OR P(2)>376+192 OR P(3)>376+192+192 THEN GOTO 6
```

Program continues

Program continues

```
160
6154 RETURN
6160 FOR M=1 TO 200:NEXT M
6164 PRINT@ 0,"
6168 PRINT@ 25,N$(V) " WINS "
6170 INPUT "DO YOU WANT TO PLAY AGAIN";A$
6180 IF LEFT$(A$,1)="Y" THEN GOTO 10
6190 END
7000 CLS:PRINT CHR$(23)
7010 PRINT "              HORSE RACE"
7020 PRINT
7030 PRINT "YOUR HORSE WILL ONLY MOVE IF YOU ANSWER A MATH PROBL"
EM CORRECTLY."
7035 PRINT
7037 PRINT "PRESS ANY KEY TO CONTINUE."
7038 A$=INKEY$ : IF A$="" THEN GOTO 7038
7039 CLS:PRINT CHR$(23)
7040 PRINT    "YOUR HORSE WILL MOVE"
7050 PRINT "1 SPACE IF YOU ADD OR SUBTRACT"
7060 PRINT "2  SPACES IF YOU MULTIPLY."
7070 PRINT "4   SPACES IF YOU DIVIDE."
7080 PRINT "    (DIVISION PROBLEMS MUST BE  ROUNDED TO 2 SIGNIFI
CANT DECIMAL PLACES.)"
7090 PRINT
7100 PRINT "PRESS ANY KEY TO CONTINUE."
7110 A$=INKEY$: IF A$="" THEN GOTO 7110
7120 CLS
7130 PRINT CHR$(23)
7140 PRINT "1 TO 3 PEOPLE MAY PLAY."
7150 PRINT
7210 PRINT "PRESS ANY KEY TO CONTINUE."
7220 A$=INKEY$:IF A$="" THEN GOTO 7220
7230 CLS
7240 INPUT "HOW MANY PEOPLE ARE PLAYING";B
7250 IF B<1 OR B>3 THEN GOTO 7120
7260 FOR Y=1 TO B
7270    PRINT
7280    PRINT "WHAT IS THE NAME OF PLAYER "Y
7282    INPUT N$(Y)
7290 NEXT Y
7292 CLS
7300 RETURN
7400 GOTO 7400
7410 REM ***************************************************
7420 REM *
7430 REM * VARIABLES USED IN THIS PROGRAM
7440 REM *
7450 REM ***************************************************
7460 REM       A$  1) TELLS COMPUTER IF USER WISHES TO ADD,
7470 REM              SUBTRACT MULTIPLY OR  DIVIDE.
7480 REM           2) LETS PROGRAM KNOW IF INSTRUCTIONS ARE
7482 REM              NEEDED
7490 REM
7500 REM       B$  ARRAY USED TO HOLD INKEY$ RESPONSES (INDEXED
7510 REM           BY V)
7520 REM
7530 REM       D   USERS REPONSE TO MATH QUESTION.
7532 REM
7540 REM       E   COUNTER USED TO BUILD BORDER ON TRACK.
7550 REM;
7560 REM       F   ONE OF 3 VARIABLES USED TO LIMIT THE USERS
7570 REM           RESPONSE TIME. (M AND H ARE ALSO USED.)
7580 REM
7590 REM       H   SAME AS F.
7600 REM
7610 REM       G   NUMBER OF SPACES THE HORSE WILL MOVE.
7620 REM           (BASED UPON A CORRECT REPONSE TO A SPECIFIC
7630 REM            TYPE OF MATH PROBLEM.)
7640 REM           FOR AN ADDITION OR SUBTRACTION PROBLEM  G=1
7650 REM           FOR A MULTIPLICATION PROBLEM            G=2
7660 REM           FOR A DIVISION PROBLEM                  G=4
7670 REM
7680 REM       J   INDEX USED WITH ANSWER ARRAY (B$).
7690 REM           USED TO ERASE THE PREVIOUS ANSWER.
7700 REM
7710 REM       M   SAME AS F
7720 REM
7730 REM       N$  ARRAY USED TO STORE PLAYERS NAMES.
7740 REM           INDEXED BY V.
7750 REM
7760 REM       P   ARRAY WHICH HAS THE CURRENT POSITION OF THE
7770 REM           USERS HORSE.
7780 REM
7790 REM       T   COUNTER USED TO PRINT THE BORDER.
7800 REM
7810 REM       V   INDEX USED WITH THE P ARRAY AND THE N$ ARRAAY
7820 REM
7830 REM       X   RANDOM NUMBER USED IN  THE COMPUTER
7840 REM           SELECTED MATH PROBLEM..
7850 REM
7860 REM       Y   1) SAME AS X.
7870 REM           2) USED TO PRINT THE VERTICAL BORDER AROUND
7880 REM              THE TRACK
```

Program Listing 2. Horse Race—Keyes.

```
1 REM ***************************************************
3 REM *
4 REM *    HORSE RACE    BY      KAY KEYES
5 REM *
7 REM ***************************************************
10 CLS
20 PRINT@ 63 + 25, "HORSE   RACE"
25 IF X= 15360 TO 16381
90 POKE 15360 + 127 + 4, 191
100 POKE 15360 + 191 + 4, 191
110 POKE 15360 + 255 + 4, 191
120 POKE 15360 + 319 + 4, 191
```

Program continued

tions were used to draw the graphics, except for variety. Line 12 sets the initial starting positions of the horses (the PRINT@ instruction is used to draw them). The horses are drawn and erased by the subroutine between lines 6000 and 6152. Once the neck of a horse has crossed the finish line, that person wins (lines 6152, and 6160 through 6168).

The math routines are found in lines 4204 through 5115 and 5200 through 5310. The numbers chosen for the computer by the program are one through 10. If you wish to change these numbers, change line 4204. The player is given the option to choose addition, subtraction, multiplication or division problems in line 4210. Lines 4220 through 4350 go to the correct routine based upon the player's selection. In our example, we would go to line 4400, which prints the problem on the screen. Line 4410 divides 8 by 3, and line 4420 rounds the answer to two decimal places. Regardless of what type of problem is chosen, the computer's final answer is stored in the variable Z.

Lines 5200 through 5310 inform the player if his or her answer was correct. If the answer was incorrect, the correct answer appears on the screen. If the answer is correct, the player's horse will move one space for an addition or subtraction problem, two spaces for a multiplication problem, or four spaces for a division problem. Lines 5312, 5320 and 5330 set the variable G accordingly. G is the varible to modify if you wish to weight the answers differently. (Some of the division problems are really difficult for the average student to figure out in the allotted time.) Lines 5400 through 5420 spin through the routine that erases and redraws the horses based upon the variable G, via a For. . . Next loop.■

Michael Duffin teaches data processing at Morton College in Cicero, Illinois.

Program continued

```
130 POKE 15360 + 383 + 4, 191
140 POKE 15360 + 447 + 4, 191
150 POKE 15360 + 511 + 4, 191
169 POKE 15360 + 575 + 4, 191
170 POKE 15360 + 639 + 4, 191
180 POKE 15360 + 703 + 4, 191
190 POKE 15360 + 767 + 4, 191
200 POKE 15360 + 831 +4, 191
999 FOR T=832TO895
1000 PRINT@T,CHR$(141);
1001 NEXT T
1002 FOR A=128  TO 191
1003 PRINT@A,CHR$(156);
1004 NEXT A
2020 POKE 15360 + 127 +188,191
2025 POKE 15360 + 63 + 188,191
2040 POKE 15360 + 191 + 188,191
2050 POKE 15360 + 255 + 188,191
2060 POKE 15360 + 319 + 188,191
2070 POKE 15360 + 383 + 188, 191
2080 POKE 15360 + 447 + 188,191
2090 POKE 15360 + 511 + 188, 191
3000 POKE 15360 +575 + 188,191
3010 POKE 15360 + 639 + 188,191
4000 PRINT@ 896,"START"
4010 PRINT@895+56,"FINISH"
5000 POKE 15360 + 255 +15,170
5010 POKE 15360 + 255+16,140
5020 POKE 15360 + 255 +17,132
5030 PRINT @330,CHR$ (160);
5040 PRINT@ 331,CHR$(134);
5050 PRINT@332,CHR$(151);
5060 PRINT@333,CHR$(131);
5070 PRINT@334,CHR$(171);
5080 X=RND(10)
5090 Y=RND(10)
5100 PRINT X"+"Y"=";
5110 Z=X+Y
5115 PRINT@460,"WHAT IS THE ANSWER";
5120 INPUT A
5130 IF Z=A GOTO 5150
5140 PRINT@460,"WRONG ANSWER. TRY AGAIN.";:GOTO 5120
5150 PRINT@460,"THAT IS THE CORRECT ANSWER"; A ;
5160 GOTO 6000
6000 PRINT@270,CHR$(32);
6010 PRINT@271,CHR$(32);
6020 PRINT@272,CHR$(32);
6030 PRINT@330,CHR$(32);
6040 PRINT@331,CHR$(32);
6050 PRINT@332,CHR$(32);
6060 PRINT@333,CHR$(32);
6070 PRINT@334,CHR$(32);
6080 PRINT@309,CHR$(170);
6090 PRINT@310,CHR$(140);
6100 PRINT@311,CHR$(132);
6110 PRINT@369,CHR$(160);
6120 PRINT@370,CHR$(134);
6130 PRINT@371,CHR$(151);
6140 PRINT@372,CHR$(131);
6150 PRINT@373,CHR$(171);
6160 PRINT@493, "****THE  WINNER****";
9999 GOTO 9999
```

```
1 REM ****************************************************
3 REM *
5 REM *     MATH GAME BY KATHLEEN  SCHMAUS
7 REM *
9 REM  ***************************************************
10 CLS
20 X=RND(10)
30 Y=RND(10)
31 INPUT "DO YOU WISH TO ADD, SUBTRACT, MULTIPLY OR DIVIDE";A$
32 IF A$="add" THEN GOTO 36
33 IF A$="SUBTRACT" THEN GOTO 37
34 IF A$="MULTIPLY" THEN GOTO 38
35 IF A$="DIVIDE" THEN GOTO 39
36 PRINT X "+" Y:Z=X+Y: GOTO 50
37 PRINT X "-" Y:Z=X-Y: GOTO 50
38 PRINT X "*" Y: Z=X*Y:GOTO 50
39 PRINT X "/" Y:Z=X/Y: GOTO 50
50 INPUT "WHAT IS THE ANSWER";Q
70 IF Q=Z GOSUB 80
71 IF Q<>Z GOTO 150
72 GOTO 10
80 PRINT@ 215, "#########"
90 PRINT@ 278, "#        #"
100 PRINT@ 342, "#  #   #  #"
110 PRINT@ 406, "#         #"
120 PRINT@ 470, "# #     # #"
130 PRINT@ 531, "#  #  #####  #  #"
140 PRINT@ 596, "##            ##"
142 FOR N=1 TO 500:NEXT N
145 RETURN
150 PRINT@ 215, "##########"
160 PRINT@ 278, "#         #"
170 PRINT@ 342, "#  #   #  #"
180 PRINT@ 406, "#         #"
190 PRINT@ 470, "#  #####  #"
200 PRINT@ 531, "#  # #     # #  #"
210 PRINT@ 596, "##            ##"
212 FOR N=1 TO 500:NEXT N
220 GOTO 50
```

Program Listing 3. Math Game—Schmaus.

✓See List of Advertisers on page 354

If your TRS-80 is so smart, then let it debug its own software.

"On Error . . . Fix"

N. C. Richardson
655 Carribean Road
Satellite Beach, FL 32937

One night at a local computer meeting we joked about the prospect of ever getting a Basic program command named "ON ERROR FIX"—a statement that would detect any and all errors and correct them on the first run without any bother to the programmer. To a limited degree, such a routine is possible.

Program Listing 1 will function on a Level II machine or one under control of a disk operating system. You can merge this routine with any program you are writing without conflict. Line zero will be before any line you would normally start with and must be executed before any errors are encountered. The rest of the routine lines start with 49990 and all are incremented by 10. This keeps the bulk of the routine out of your way and makes it easier to copy. The letter E is the only variable to avoid in your program.

The routine as shown will fix six types of errors, tell what type of error, what line it's in and if it made a deletion. If it can't fix the error, it will tell the type and line. In all cases, execution of your program will not stop until it encounters the command End or Stop. If you have a printer, change all the Print statements to LPRINT, so the error notes will not interfere with the monitor display.

Error Address

Here's how the program functions. Line 49990 has a colon (:), the use of which will be described in line 50020, and the common End is there in case you don't end your program's execution. This will keep the routine from trying to find errors in itself. Line 50000 sets the variable ER equal to the error number and EL equal to the error line number. 50010 causes a jump to 50160 on any error it cannot fix.

Line 50020 is the heart and brains of the routine. The addresses 16632 and 16631 contain the numerical data which points to an address when an error occurs. This address will either contain a zero prior to two pointer addresses and two line addresses of the line which contains the error, or it will be one that contains the colon (ASCII 58) immediately prior to the error in a multi-statement line. Thus the variable EA (error address) is set to the value to which 16632 and 16631 point, plus either five or one—so the routine won't see those leading addresses or the first colon. Line 50030 sets the variable E equal to the contents of EA (error address) and E1 equal to the contents of the next address. The variables in this line will be incremented to find the true error address and fix it.

Skipping over the actual error-correcting lines we come to line 50100. This line says that if the contents of the error address contain a colon or if one of the error-correcting lines has been successful (EF has been set), then stop the search and go to line 50120. If line 50100 is false, execution drops to line 50110 where EA (error address) is incremented by one and control is sent back to line 50030. This action continues until line 50100 is true. If one of the error-correcting lines (50040–50090) are successful we want to stop the scan, get a printout of the result and continue with main program execution. If none of the lines are successful, if a syntax error other than the one we have a line for occurs, then we can dump out by line 50100 when it finds the next colon. That is why we don't want the routine to see the first colon and, just in case there isn't another colon in your program, one is placed before the End statement in line 49990.

Line by Line

Line 50040 is the first error-correcting line. It says if the error number is one (Next Without For) and if the error address contains the code for the statement Next (internal code 178) then POKE a space (ASCII 32) in the error address and set EF (error flag variable) to equal two. Line 50050 checks to see if the error number is two (Syntax Error) and if so, whether the error address contains the Print statement and if the error address plus one contains an uppercase @. If all are true it will POKE a lowercase @ in the error address plus one and it will set EF = 1. This line helps explain why it is necessary to look at the error address and EA plus one on each pass. Line 50060 looks for a number three error (Return Without Error) and when it finds the address which contains Return (Internal code 146) it will POKE a space in its place. It will also set EF = 2. Line 50070 acts on an error four (Out of Data) and automatically executes a Restore command.

At 50080 we are looking for a number 11 error (Division By Zero). If such is the case and the error address contains the number 208, which is the internal code for division (/) and the next address contains a zero it will change it to a one. It sets EF = 1. The last of the error-fixing lines, 50090, looks for a number 19 error (Resume Without Error). If this is true and the error address contains 159 (Internal code for Resume), it will POKE a space in its place; it sets EF = 2.

Lines 50120 through 50340 simply print out the type of error that occurred and then hold the carriage return and line feed so that line 50350 can print "In Line" and the line number of the error. The EF (error flag) variable is used at 50360. If EF = 0, control goes to 50370. If EF = 1, control goes to 50380 and EF = 2 sends control to 50390, which prints "Not Fixed" and executes "Resume Next." Execution of your program will continue with the next statement after the error statement, and the error will remain. Line 50380 prints "Was Fixed," sets the EF variable flag back to zero and resumes execution by stating where an error had been. Line 50390 acts exactly as 50380 except it prints "Was Deleted."

Thus your program, no matter how many errors, will execute to the end. However, you will get a running report of every error as soon as it is encountered and some of your errors may be fixed permanently. Those uppercase @s are sometimes difficult to find, but this program will hunt them down.

Some fixes seem questionable. Would changing the denominator from zero to one work for you? If not, would some other number do? Change the line to fit your purpose, or delete line 50080. Would restoring your data automatically be detrimental? If so, take out line 50070. The other fix lines shouldn't give you any trouble.

If you have a disk system save the program in ASCII with the command save "ERROR FIX/ASC",A. Then you will be able to merge the program with your current program after it is written. Otherwise load Error Fix from tape before you start to enter your program.■

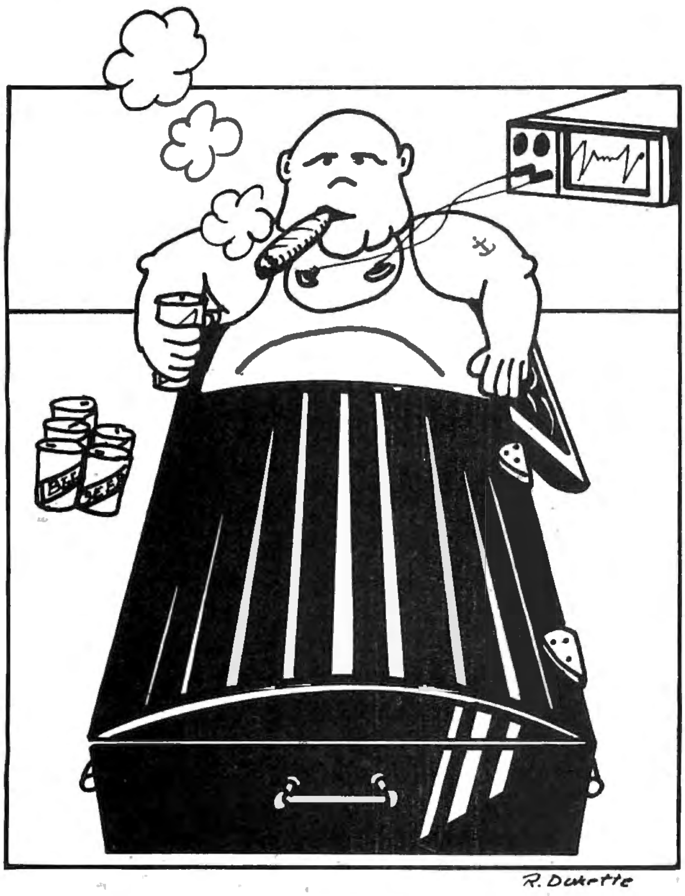
```
0 CLS:DEFINT E:ON ERROR GOTO 50000
49990 :END
50000 ER=ERR/2+1:EL=PEEK(16621)*256+PEEK(16620)
50010 IFER=1 OR ER=2 OR ER=3 OR ER=4 OR ER=11 OR ER=19 THEN 5002
0 ELSE 50160
50020 EA=PEEK(16632)*256+PEEK(16631):IF PEEK(EA)=0 THEN EA=EA+5
ELSE EA=EA+1
50030 E=PEEK(EA):E1=PEEK(EA+1)
50040 IFER=1 AND E=135 POKE EA,32:EF=2
50050 IFER=2 THEN IFE=178 AND E1=96 THEN POKE EA+1,64:EF=1
50060 IFER=3 AND E=146 POKE EA,32:EF=2
50070 IFER=4 THEN RESTORE:EF=1
50080 IFER=11 AND E=208 AND E1=48 THEN POKE EA+1,49:EF=1
50090 IFER=19 AND E=159 POKE EA,32:EF=2
50100 IFE=58 OR EF>0 GOTO 50120
50110 EA=EA+1:GOTO 50030
50120 IFER=1PRINT"NEXT WITHOUT FOR";
50130 IFER=2PRINT"SYNTAX ERROR";
50140 IFER=3PRINT"RETURN WITHOUT GOSUB";
50150 IFER=4PRINT"OUT OF DATA";
50160 IFER=5PRINT"ILLEGAL FUNCTION CALL";
50170 IFER=6PRINT"OVERFLOW";
50180 IFER=7PRINT"OUT OF MEMORY";
50190 IFER=8PRINT"UNDEFINED LINE";
50200 IFER=9PRINT"SUBSCRIPT OUT OF RANGE";
50210 IFER=10PRINT"REDIMENSIONED ARRAY";
50220 IFER=11PRINT"DIVISION BY ZERO";
50230 IFER=12PRINT"ILLEGAL DIRECT";
50240 IFER=13PRINT"TYPE MISMATCH";
50250 IFER=14PRINT"OUT OF STRING SPACE";
50260 IFER=15PRINT"STRING TOO LONG";
50270 IFER=16PRINT"STRING FORMULA TOO COMPLEX";
50280 IFER=17PRINT"CAN'T CONTINUE";
50290 IFER=18PRINT"NO RESUME";
50300 IFER=19PRINT"RESUME WITHOUT ERROR";
50310 IFER=20PRINT"UNPRINTABLE ERROR";
50320 IFER=21PRINT"MISSING OPERAND";
50330 IFER=22PRINT"BAD FILE DATA";
50340 IFER=23PRINT"DISK BASIC ONLY";
50350 PRINT" IN LINE ";EL;
50360 IF EF=1 THEN 50380 ELSE IF EF=2 THEN 50390
50370 PRINT" NOT FIXED":RESUME NEXT
50380 PRINT" WAS FIXED":EF=0:RESUME
50390 PRINT" WAS DELETED":EF=0:RESUME
```

Program Listing

✓See List of Advertisers on page 354

Prolong yours with this immortal program.

Lifespan



Joseph Wailand
1343 Millersport Highway
Williamsville, NY 14221

Numerous factors affect your life expectancy. Some of these can be controlled and some cannot. Sex, heredity and personality are important, but so are your life-style and habits. A thorough discussion of this subject can be found in *Dr. Kugler's Seven Keys to a Longer Life* by Hans J. Kugler.

This program will provide an estimate of your life span and perhaps help you identify those areas which can be improved. Extensive use of the INKEY$ function makes the program pleasant and easy to use. It is popular at parties and can be used to demonstrate your computer. The program is written in Level II Basic and takes less than 7K bytes of memory. ■

Program Listing

```
10 REM LIFE EXPECTANCY CALCULATION
15 REM INSPIRED BY AN ARTICLE
20 REM FROM PEOPLE'S ALMANAC 2, P.974, ARTICLE BY PETER PASSELL
30 REM PROGRAM WRITTEN BY JOSEPH WAILAND
40 CLS:PRINT"                    LIFE EXPECTANCY CALCULATION"
50 PRINT:PRINT"     THIS TEST PROVIDES AN APPROXIMATE CALCULATIO
N OF YOUR"
60 PRINT"LIFE EXPECTANCY IF YOU ARE BETWEEN 20 AND 65 YEARS OLD
AND"
70 PRINT"ARE REASONABLY HEALTHY."
80 FOR I=1 TO 6000:NEXT I
90 S=72
100 CLS:PRINT"GENDER":PRINT:PRINT
110 PRINT"ARE YOU A MAN OR A WOMAN?"
120 G$=INKEY$:IF G$=""THEN120
130 IF (G$<>"M") AND (G$<>"W") THEN 120
140 IF G$="M"THEN S=S-3 ELSE S=S+4
150 GOTO180
160 B$=INKEY$:IF B$=""THEN160
170 IF (B$<>"Y") AND (B$<>"N") THEN 160 ELSE RETURN
180 CLS:PRINT"LIFE-STYLE":PRINT
190 PRINT"DO YOU LIVE IN AN URBAN REGION WITH A POPULATION GREAT
ER"
200 PRINT"THAN 2,000,000?":PRINT:GOSUB 160
210 IFB$="Y"THEN S=S-2 ELSE GOTO240
220 PRINT"NERVOUS TENSION AND AIR POLLUTION ARE ASSOCIATED"
230 PRINT"WITH CITY LIVING.":PRINT:GOTO270
240 PRINT"DO YOU LIVE IN A TOWN WITH A POPULATION UNDER 10,000"
250 PRINT"OR ON A FARM?":PRINT:GOSUB160
260 IFB$="Y"THEN S=S+2
270 PRINT"DOES YOUR OCCUPATION INVOLVE MOSTLY DESK WORK?":PRINT:
GOSUB160
280 IFB$="Y"THEN S=S-3 ELSE 300
290 GOTO330
300 PRINT"DOES YOUR WORK REQUIRE HEAVY PHYSICAL LABOR ON A"
310 PRINT"REGULAR BASIS?":PRINT:GOSUB160
320 IF B$="Y" THEN S=S+3
330 PRINT"DO YOU DO INTENSE EXERCISES (RUNNING, SWIMMING, TENNIS
, ETC.)"
340 PRINT"4 OR MORE TIMES A WEEK FOR AT LEAST 30 MINUTES PER SES
SION?":GOSUB 160
350 IF B$="Y" THEN S=S+2
360 CLS:PRINT"LIFE-STYLE":PRINT:PRINT
370 PRINT"DO YOU LIVE WITH ANOTHER PERSON?":PRINT:GOSUB 160
380 IF B$="Y" THEN S=S+5 ELSE 430
390 PRINT"WHEN PEOPLE ARE LIVING TOGETHER, THEY ARE LESS DEPRESS
ED,"
400 PRINT"EAT BETTER, AND TAKE CARE OF ONE ANOTHER."
410 FOR I=1TO4000:NEXTI
420 GOTO460
430 PRINT"HOW MANY DECADES HAVE YOU LIVED ALONE SINCE AGE 25?"
440 Y$=INKEY$:IF Y$=""THEN440
450 Y=VAL(Y$):S=S-Y
460 CLS:PRINT"PERSONALITY":PRINT
470 PRINT"DO YOU SLEEP MORE THAN 10 HOURS EACH NIGHT?":PRINT:GOS
UB160
480 IF B$="Y" THEN S=S-4 ELSE 510
490 PRINT"EXCESSIVE SLEEP CAN BE A SIGN OF CIRCULATORY DISEASES"
500 PRINT"OR DEPRESSION.":PRINT
510 PRINT"ARE YOU EASILY ANGERED, INTENSE, OR AGGRESSIVE?":PRINT
:GOSUB 160
520 IF B$="Y" THEN S=S-3 ELSE 540
530 GOTO560
540 PRINT"ARE YOU RELAXED OR EASY GOING?":PRINT:GOSUB160
550 IF B$="Y" THEN S=S+3
560 PRINT"ARE YOU A HAPPY PERSON?":PRINT:GOSUB160
570 IF B$="Y" THEN S=S+1 ELSE 590
580 GOTO610
590 PRINT"ARE YOU A SAD PERSON?":PRINT:GOSUB160
```

Program continues

Program continued

```
600 IF B$="Y" THEN S=S-2
610 CLS:PRINT"PERSONALITY":PRINT:PRINT
620 PRINT:PRINT"HAVE YOU GOTTEN A SPEEDING TICKET WITHIN THE PAS
T YEAR?":PRINT:GOSUB160
630 IFB$="N"THEN680
640 PRINT"TRAFFIC ACCIDENTS ARE THE MAJOR CAUSE OF DEATH AMONG"
650 PRINT"YOUNG ADULTS."
660 FOR I=1TO4000:NEXT I
670 S=S-1
680 CLS:PRINT"ACHIEVEMENTS":PRINT:PRINT
690 PRINT"DO YOU EARN MORE THAN $50,000 A YEAR?":PRINT:GOSUB160
700 IF B$="N" THEN 730
710 S=S-2
720 PRINT"HIGH LIVING AND TENSION ARE OFTEN ASSOCIATED WITH WEAL
TH.":PRINT
730 PRINT"DO YOU HAVE A COLLEGE DEGREE?":PRINT:GOSUB160
740 IF B$="N" THEN 790
750 PRINT"EDUCATION IS SUPPOSED TO LEAD TO MODERATION.":PRINT
760 S=S+1
770 PRINT"DO YOU HAVE A GRADUATE OR PROFESSIONAL DEGREE?":PRINT:
GOSUB160
780 IF B$="Y" THEN S=S+2
790 CLS:PRINT"ACHIEVEMENTS":PRINT:PRINT
800 PRINT"ARE YOU 65 YEARS OLD, OR OLDER, AND STILL WORKING?":PR
INT:GOSUB160
810 IFB$="Y" THEN S=S+3 ELSE840
820 PRINT"RETIREMENT SOMETIMES LEADS TO DEATH."
830 FORI=1TO4000:NEXTI
840 CLS:PRINT"HEREDITY":PRINT:PRINT
850 PRINT"DID ANY OF YOUR GRANDPARENTS LIVE TO AGE 80 OR MORE?":
PRINT:GOSUB160
860 IF B$="N" THEN 890
870 PRINT"DID ALL 4 OF YOUR GRANDPARENTS LIVE TO AGE 80 OR MORE?
":PRINT:GOSUB160
880 IF B$="Y" THEN S=S+6 ELSE S=S+2
890 PRINT"DID EITHER OF YOUR PARENTS DIE OF A HEART ATTACK OR ST
ROKE      BEFORE THE AGE OF 50?":PRINT:GOSUB160
900 IF B$="Y" THEN S=S-4
910 PRINT"DID ANY PARENT, BROTHER, OR SISTER UNDER AGE 50 HAVE A
 HEART"
920 PRINT"CONDITION OR CANCER - OR HAS HAD DIABETES SINCE CHILDH
OOD?":GOSUB160
930 IF B$="Y" THEN S=S-3
940 CLS:PRINT"HEALTH":PRINT
950 PRINT"DO YOU SMOKE CIGARETTES?":PRINT:GOSUB160
960 IF B$="N" THEN 1040
970 PRINT"DO YOU SMOKE MORE THAN 2 PACKS A DAY?":GOSUB160
980 IF B$="Y" THEN 1020
990 PRINT"DO YOU SMOKE 1 TO 2 PACKS A DAY?":GOSUB160
1000 IF B$="Y" THEN 1030
1010 S=S-3:GOTO1040
1020 S=S-8:GOTO1040
1030 S=S-6:GOTO1040
1040 PRINT:PRINT"DO YOU DRINK THE EQUIVALENT (OR MORE) OF A HALF
 PINT OF LIQUOR"
1050 PRINT"PER DAY?":PRINT:GOSUB160
1060 IF B$="Y" THEN S=S-1
1070 PRINT"ARE YOU OVERWEIGHT?":PRINT:GOSUB160
1080 IF B$="N" THEN 1180
1090 PRINT"ARE YOU OVERWEIGHT BY 50 LBS. OR MORE?":GOSUB160
1100 IF B$="Y" THEN 1150
1110 PRINT"ARE YOU OVERWEIGHT BY 30 TO 50 LBS.?":GOSUB160
1120 IF B$="Y" THEN 1160
1130 PRINT"ARE YOU OVERWEIGHT BY 10 TO 30 LBS.?":GOSUB160
1140 IF B$="Y" THEN 1170 ELSE 1180
1150 S=S-8:GOTO1180
1160 S=S-4:GOTO1180
1170 S=S-2:GOTO1180
1180 CLS:PRINT"AGE":PRINT:PRINT
1190 A=30
1200 PRINT"ARE YOU LESS THAN 30 YEARS OLD?":GOSUB160
1210 IF B$="Y"GOTO1320
1220 PRINT:PRINT"ARE YOU BETWEEN 30 AND 40 YEARS OLD?":GOSUB160
1230 IF B$="N"GOTO1250
1240 A=39:S=S+2:GOTO1320
1250 PRINT:PRINT"ARE YOU BETWEEN 40 AND 50 YEARS OLD?":GOSUB160
1260 IF B$="N" GOTO 1280
1270 A=49:S=S+3:GOTO1320
1280 PRINT:PRINT"ARE YOU BETWEEN 50 AND 70 YEARS OLD?":GOSUB160
1290 IF B$="N" THEN 1310
1300 A=70:S=S+4:GOTO1320
1310 A=71:PRINT:PRINT"I ASSUME YOU ARE OVER 70 YEARS OLD."
1320 IF G$="M" THEN1360
1330 PRINT:PRINT"DO YOU VISIT A GYNECOLOGIST AT LEAST ONCE A YEA
R?":GOSUB160
1340 IF B$="Y" THEN S=S+2
1350 GOTO1390
1360 IFA<=40THEN1390
1370 PRINT:PRINT"DO YOU HAVE ANNUAL MEDICAL CHECKUPS?":GOSUB160
1380 IF B$="Y" THEN S=S+2
1390 IF (S>=60) AND (S<65) THEN 1470
1400 IF (S>=65) AND (S<70) THEN 1480
1410 IF (S>=70) AND (S<75) THEN 1490
1420 IF (S>=75) AND (S<80) THEN 1500
1430 IF (S>=80) AND (S<85) THEN 1510
1440 IF (S>=85) AND (S<90) THEN 1520
1450 IF (S>=90) AND (S<95) THEN 1530
1460 IF S>=95  THEN 1540
1470 B=26:B1=35:G=15:G1=19:GOTO1550
1480 B=36:B1=47:G=20:G1=29:GOTO1550
1490 B=48:B1=60:G=30:G1=38:GOTO1550
1500 B=61:B1=74:G=39:G1=52:GOTO1550
1510 B=75:B1=86:G=53:G1=69:GOTO1550
1520 B=87:B1=95:G=70:G1=87:GOTO1550
1530 B=96:B1=98:G=88:G1=96:GOTO1550
1540 B=99:B1=99.9:G=97:G1=99.6:GOTO1550
1550 CLS:PRINT:PRINT"YOUR CALCULATED LIFE EXPECTANCY IS";S;"YEAR
S."
1560 IF S<A GOSUB1620
1570 IFS<60THEN1600
1580 PRINT:PRINT"AT THIS AGE, YOU WILL HAVE OUTLIVED ";B;"TO";B1
;"% OF THE MEN"
1590 PRINT"AND ";G;"TO";G1;"% OF THE WOMEN."
1600 PRINT:PRINT:PRINT"DO YOU WANT TO RUN THE PROGRAM AGAI
N (Y/N)?"
1602 GOSUB 160
1604 IF B$<>"Y" THEN END
```

SAVE!

EPSON MX-80

$489.00

THE REAL THING

Yes this is the EPSON MX-80 printer you have read so much about... the one everyone's buying. Better get yours while they are available at this terrific price! One of the most advanced 80 column dot matrix printers available featuring the world's first disposable print head. There are lots of printers to choose from but, once you have compared, you will find it hard to pass up the EPSON MX-80.

FEATURES

- Bidirectional Printing
- Logical seeking of shortest lines
- 80 Characters per second
- 64 Block Graphic Characters
- Forms Handling
- Disposable print head
- Adjustable Tractor-type pin feed
- Paper width 4 to 10 inches
- Inked ribbon cartridge with 3 million character life
- Self test mode
- Full 96 character ASCII with decenders
- Printing Modes: Standard, Double, Emphasized and Double Emphasized
- Correspondence Quality

Add $6.00 shipping (Cont. US). Does not include cable. See opposite page for ordering information.

EXTRAS FOR EPSON

TRS80 Cable (I & III) $29.95
APPLE Cable & Interface $96.00
RS232 Interface $69.95
Ribbon Cartridge, Black $13.95
Interface & Cable (from Mod. I Keyboard) $99.95

ORDER TODAY!!

Save Even More EPSON MX-70

$389.00

NO FRILLS PRINTING AT NO FRILLS PRICE

FEATURES

- ★ Unidirectional printing
- ★ 80 Characters per second
- ★ Bit-plot graphics
- ★ Forms handling
- ★ Disposable print head
- ★ Adjustable tractor-type pin feed
- ★ Paper width 4 to 10 inches
- ★ Inked ribbon cartridge with 3 million character life
- ★ Self test mode
- ★ Full 96 character ASCII
- ★ Printing modes: normal & normal expanded

Add $6.00 shipping (Cont. US). Does not include cable. See opposite page for ordering information.

ALL NEW! MX-80 F/T

All the features of the MX-80 plus friction feed.

$599.00

LIMITED QUANITIES

CABLES, SOFTWARE & OTHER GOODIES

★ DRIVE CABLES

2-Drive $24.95
3-Drive $29.95
4-Drive $34.95
Extender Cables $16.95

★ OPERATING SYSTEMS

DOSPLUS 3.3, 3.3D (spec. Mod I or III) $89.95
LDOS $135.00
NEWDOS/80 .... $135.00
TRSDOS 2.3 DISK & MANUAL $17.95
DOUBLE ZAP II $44.00

Add $1.25 per pound shipping ($1.00 minimum).

★ DISKETTES (5-1/4 in., softsectored, box of 10)

Cert. one side, single density $29.95
Cert. two sides, double density $39.95

★ OTHER SOFTWARE & BOOKS

SCRIPSIT (disk) $89.95
SPECIAL DELIVERY $119.95
XTRA SPECIAL DELIVERY $189.95
SUPER UTILITY $39.95
MICROSOFT BASIC DECODED & OTHER MYSTERIES $25.95
TRS80 DISK & OTHER MYSTERIES .. $19.95
MAKE 80 .... $14.95

SPECIAL WORD PROCESSING PACKAGE

Includes the following:

- ★ EPSON MX-80 PRINTER
- ★ CABLE
- ★ SCRIPTSIT)disk)
- ★ SPECIAL DELIVERY (disk)
- ★ freight and insurance.

SPECIAL

$725.00

(Reg $917.00)

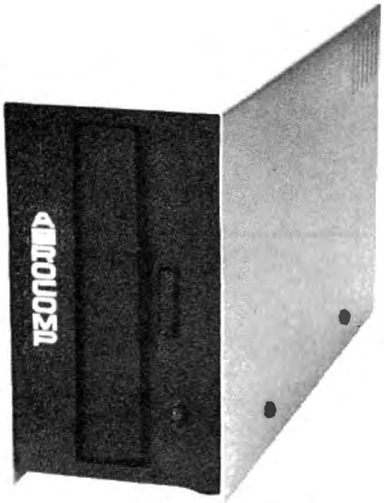
Order today from

AEROCOMP

Redbird Airport, Bldg. 8
P.O. Box 24829
Dallas, Texas 75224 ✓387

See Opposite Page ⇨

AROCOMP5/23



TRS 80*

DISK DRIVES

40 & 80 TRACK

as low as $299.95

ORDER TODAY!

★ 40-Track Drive. $299.95
★ 80-Track Drive. $429.95
★ 40-Track "FLIPPY" Drive. . . . $329.95
★ 80-Track "FLIPPY" Drive. . . . $449.95
★ 40-Track Dual-Head Drive. . $449.95
★ 80-Track Dual-Head Drive. . $579.95

All above drives are complete with silver enclosure, power supply and external drive cable connection 115 VAC 50/60 Hz 230 VAC, 50/60 Hz available on special order

★ 40-Track Bare Drive. $269.95
★ 80-Track Bare Drive. $399.95
★ 40-Track "FLIPPY" Bare Drive $299.95
★ 80-Track "FLIPPY" Bare Drive $419.95

★ SPECIAL PACKAGES ★

STARTER A. $339.95
40-Track Drive, 2-drive cable, TRSDOS 2.3 Disk & Manual, Freight & Ins.

SRARTER B. $369.95
40-Track "FLIPPY" Drive,cable, TRSDOS 2.3 Disk & Manual, Freight & Ins.

COMBO C. $465.00
40-Track "FLIPPY" Drive, 2-drive cable, NEWDOS/80 or LDOS, Freight & Ins.

COMBO D. $585.00
80-Track "FLIPPY" Drive, 2-drive cable, NEWDOS/80 or LDOS, Freight & Ins.

COMBO E. $709.00
Two 40-Track "FLIPPY" Drives, 4-Drive cable, TRSDOS 2.3 Disk & Mannual, Freight & Ins.

COMBO F. $999.00
Two 80-Track "FLIPPY" Drives, 4-drive cable, NEWDOS/80 or LDOS, Freight & Ins.

COMBO G. $909.00
Two 40-Track Dual-Head Drives, cable, TRSDOS 2.3 Disk & Manual, Freight & Ins.

COMBO H. $1299.00
Two 80-Track Dual-Head Drives, cable, NEWDOS/80 orLDOS, Freight & INS. DOSPLUS available with any special package.

Add$5.00 per drive for shipping & handling (Cont. US)

FREE TRIAL OFFER

Order your AEROCOMP Disk Drive and use it with your system for up to 14 days. If you are not satisfied for ANY REASON (except misuse or improper handling), return it, packed in the original shipping container, for a full refund. (Special packages will be perforated) We have complete confidence in our products and we know you will be satisfied! **ORDER TODAY!!**

* TRS80 is TRADEMARK of TANDY CORP

NEW LOW PRICES

Thanks to you our sales volume has allowed us to cut costs and we're passing the savings along. We offer the best disk drive value in the market place. Reliability, features and cost tough to beat. We deliver...and we stand behind our products. AEROCOMP is the only manufacturer to offer FREE TRIAL! Buy Aerocomp drives today. You deserve the best.

BEST FEATURES

★ Fast 5 ms. track-track access time.
★ Operates single or double density.
★ "FLIPPY" feature for lower media cost (40-1 & 80-1).
★ Head load solenoid.
★ Disk ejector.
★ Easy entry door.
★ NEW EXTERNAL DRIVE CABLE CONNECTION. (no longer need to remove the cover to hook up cable)

EPSON MX80 PRINTERS

$489.00

TRS80 Cable (I & III). $29.95
APPLE Cable & Interface. . $96.00
RS232 Interface. $69.95
Ribbon Cartridge, Black. . . $13.95
Print Head (STD). $26.95

Add $6.00 shipping (Cont. U.S.) does not include cable.

MX80 F/T

All the features of MX80 plus Friction Feed. Shipping & cables as above.

$599.00

Add $3.00 for shipping & handling

WARRANTY

We offer you a 120 day unconditional warrenty on parts and labor against any defect in materials and workmanship. In the event service, for any reason, becomes nescessary, our service department is fast, friendly and cooperative.

100% TESTED

AEROCOMP Disk Drives are completely assembled at the factory and ready to plug in when you receive them. Each drive is 100% bench tested prior to shipment. We even enclose a copy of the test checklist, signed by the test technician, with every drive. AEROCOMP MEANS RELIABILITY!!

ORDER NOW!!

To order by mail, specify Model Number(S) of Drive, cable, etc. (above), enclose check, money order, VISA or MASTERCHARGE card number and expiration date, or request C.O.D. shipment. Texas residents add 5% sales tax. Add $5.00 per drive for shipping & handling (Cont. US). Please allow 2 weeks for personal checks to clear our bank. No personal checks will be accepted on C.O.D. shipments-cash, money orders or certified checks only. You will receive a card showing the exact C.O.D. amount before your shipment arrives. Be sure to include your name and shipping address. You will be notified of the scheduled shipping date. Your bank credit card will NOT be charged until the day we ship.!!

DRIVE CABLES

2-DRIVE. $24.95
4-DRIVE. $34.95

WRITE AEROCOMP TODAY FOR MORE VALUES !!!

CALL TOLL FREE FOR FAST SERVICE (800) 824-7888, OPERATOR 24

FOR VISA/MASTERCHARGE/C.O.D. ORDERS

California dial (800) 852-7777, Operator 24. Alaska and Hawaii dial (800) 824-7919, Operator 24.

TOLL FREE LINES WILL ACCEPT ORDERS ONLY!

For Applications and Technical information, call (214) 337-4346 or drop us a card.

Dealers inqiries invited

AEROCOMP

Redbird Airport, Bldg. 8
P.O. Box 24829
Dallas, TX 75224

387

AROCOMP/23
AROCOMP/24

See List of Advertisers on page 354

Get quality graphs of math functions.

Curve Plotter

Steven M. Zimmerman, Ph.D.
Craig Stanley
College of Business
University of South Alabama
Mobile, AL 36688

Your TRS-80 has the ability to draw quality graphics of any mathematical function as good or possibly better than any draftsmen's. Our program will output to CRTs and hard copy printers, and includes scales for both.

The graphic capacity of our TRS-80 interested us not only as visual attention-getter for games, but also as a useful tool in defining and describing equations and other mathematical functions. This program is the latest result in a long line of routines. Our earlier versions graphed somewhat faster than this version, but they lacked a scaling feature which allows for function variations without changing it. This graphing routine can be invaluable for representing business and statistical data.

It also demonstrates the concept of mathematical functions. Any single variable function may be plotted in this program; however, care must be taken to ensure the function exists within the domain and range selec-

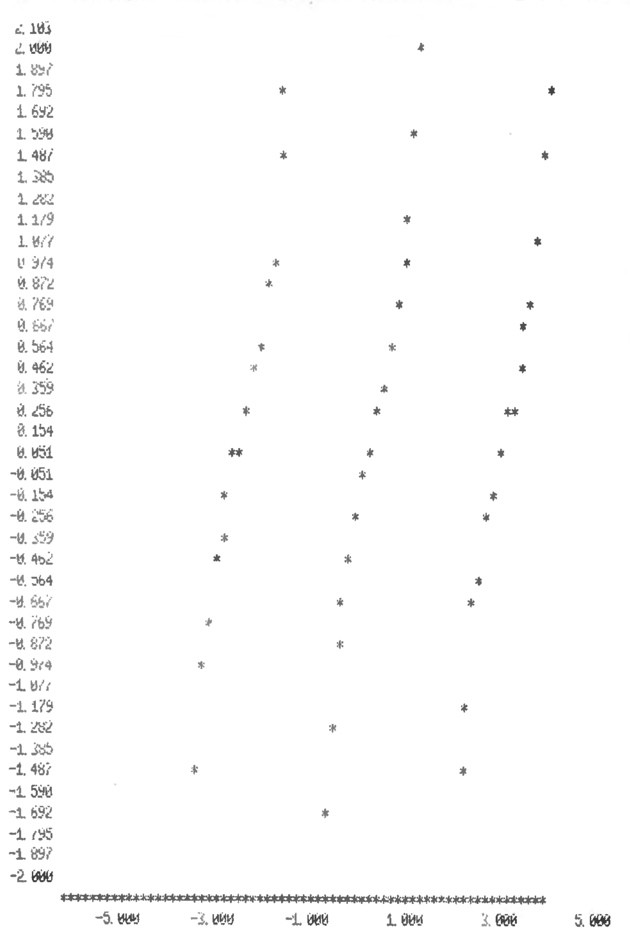


Fig. 1. Y = Tan(x)

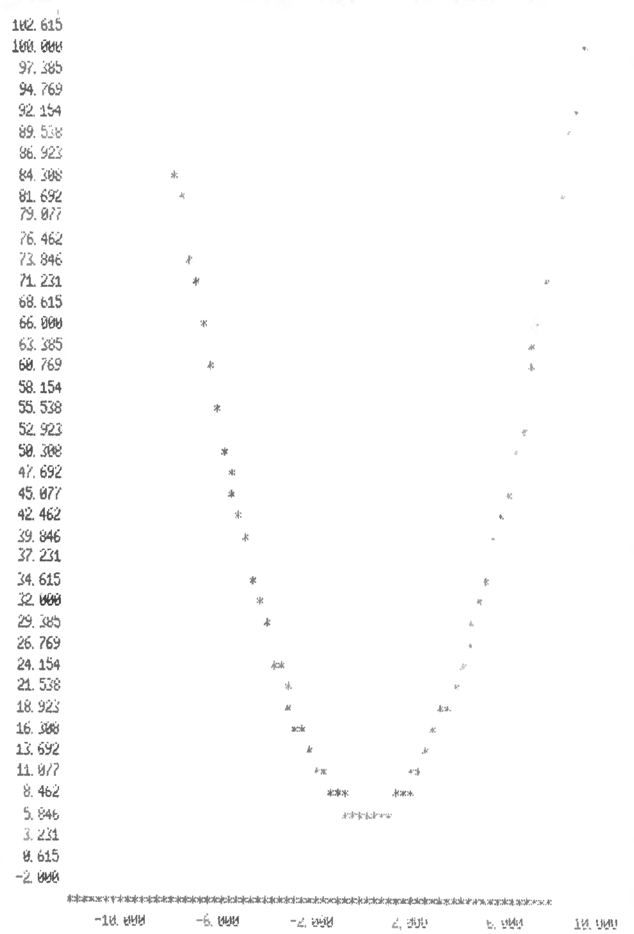


Fig. 2. Y = x↑2 + Y

ted for the plot. The sine, cosine, and other trigonometric functions make delightful graphics when plotted by this program.

Program Output and Input

The results of our curve-plotting program may be outputted to either the CRT or printer. The curve is drawn on the CRT and then is copied on to the printer by a screen-copying routine. Figs. 1 and 2 are sample runs of the program. Fig. 1 represents a tangent function, while Fig. 2 is a second-degree polynomial.

The main input to be aware of is the equation of the line to be plotted. The user must enter the equation in the form: 1000 Y = F(x). For example, to plot the line Y = 2X + 10, you must type: 1000 Y = 2*X + 10, and then restart the program by typing RUN 35. The range of values for X and Y for the function must also be inputted. It is possible to specify where the function does not exist, so care must be taken. If you try to graph your function where it cannot be plotted, three asterisks will appear in the screen's top right corner. When the plotting routine is complete, "Press Enter to continue" will appear at the top. In addition to specifying the domain and range, the user must also note the increment to plotting. You should experiment with this variable to get an idea of what value is needed for any given situation. If your increment value is too small, too few points will be plotted and you will get nothing more than a few dots. On the other hand, if the range is too large, you will get a very dense line. It may take awhile to draw a curve for very large increment values.

The listed program for curve plotting is simple to use and is of enormous value in demonstrating the TRS-80's abilities. It can also be valuable as an add-on to yield a display of the results of other computer operations; we have used curve plotting for a regression analysis program. The program also proved enjoyable as a computer demonstration to an algebra class, and we had no trouble showing the shape of different distributions. ■

Program Listing

```
10 CLEAR500:CLS:REM        "CURVE"
20 CLS:ONERROR GOTO4000:PRINT"CURVE PLOTTER":PRINT"DEVELOPED BY"
:PRINT"STEVEN M. ZIMMERMAN,PH.D. AND CRAIG STANLEY 1980 ":I=0
30 INPUT"DOES LINE 1000 CONTAIN YOUR FUNCTION (Y/N)";Y$:IFY$="N"
END
35 INPUT"PRINTER(Y/N)";P$:INPUT"INCREMENT (100+-)";JJ
40 GOTO2035
1000 Y=1/X
1010 GOTO2125
2035 INPUT"LOWER AND UPPER LIMITS OF X";SX,TX
2036 INPUT"LOWER AND UPPER LIMITS OF F(X)";SY,TY
2040 CLS:FORI=4TO43:SET(18,I):NEXT
2050 FORI=19TO125:SET(I,43):NEXT
2070 I=7:FORJ=64TO832STEP128:I=I-1:PRINT@J,(SY+I*(TY-SY)/6);:NEX
T
2080 I=-1:FORJ=970TO1015STEP9:I=I+1:PRINT@J,(SX+I*(TX-SX)/5);:NE
XT
2090 GOTO2115
2100 PRINT@22,"PRESS ENTER TO CONTINUE";
2110 Q$=INKEY$:IFQ$=""THEN2100ELSE20
2115 S=(TX-SX)/JJ
2120 FORX=SXTOTXSTEPS:GOTO1000
2125 E=23+((X-SX)/(TX-SX))*93:F=40-((Y-SY)/(TY-SY))*37:IFE>5ANDE
<126THENIFF>0ANDF<45 SET(E,F) ELSE PRINT@61,"***";:FORL=1TO10:R=
RND(100):NEXTL:PRINT@61,"   ";
2130 NEXT
2200 IFP$<>"Y"THEN2100
3000 LPRINT" "
3010 E$="###.##":FORI=2TO43:J=42-I:IFI<43LPRINTUSINGE$;(SY+J*(TY
-SY)/39);:
3020 FORX=0TO110STEP1.6
3030 IFPOINT(X,I)THENLPRINT"*";ELSELPRINT" ";
3040 NEXTX:LPRINT" ":NEXTI
3050 FORI=0TO6:G(I+1)=SX+I*(TX-SX)/5:NEXT
3060 E$="          ###,###.###  ###,###.###  ###,###.###  ###,###.
###  ###,###.###  ###,###":LPRINTUSINGE$;G(1),G(2),G(3),G(4),G(5
),G(6):GOTO2100
4000 RESUME 2125
```

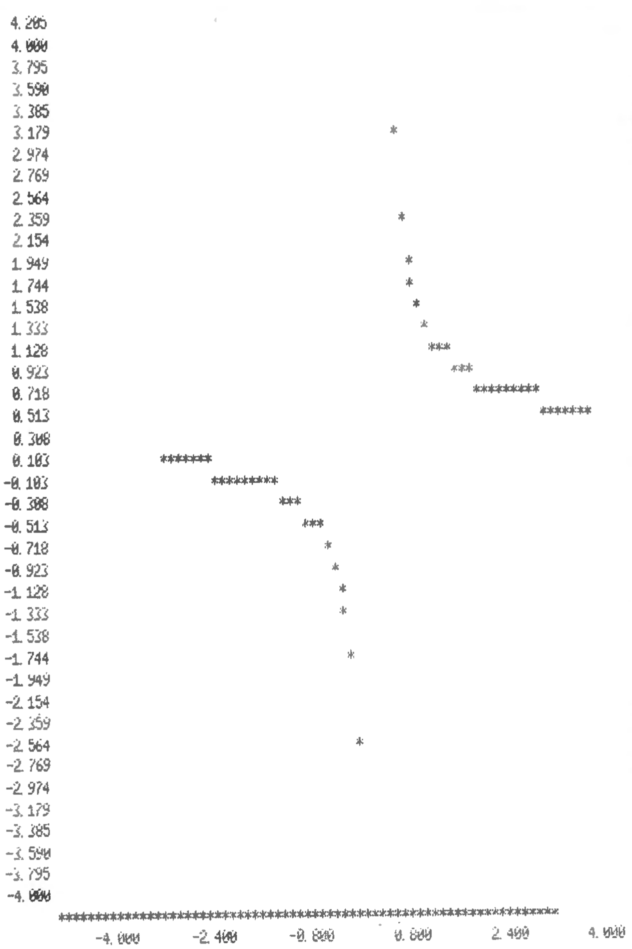


Fig. 3. Y = 1/x

✓See List of Advertisers on page 354

Manipulate the print density on your line printer.

Width Control

Jim Barbarello
RD #1, Box 241 H. Tennent Rd.
Englishtown, NJ 07726

The TRS-80 microcomputer and Radio Shack's serial line printer can team up to create a quality, low-cost business computer system. This system has one drawback, however. The printer (actually Centronics' Model 779 built for Radio Shack) has a character width control which is a continuously variable potentiometer. Once disturbed, readjusting the control to the exact previous setting is almost impossible. Thus, if tab positions must correspond to exact distances on the line (as with forms), the printer is virtually unusable.

What we need is a control to set exactly repeatable character widths (or print density). Also, if a variety of print densities is available, a visually pleasing product can be printed with intermixed character sizes. (Note: The character sizes will be identical in any given line but can vary from line to line.)

Fortunately, the printer design affords easy modification for such a facility. One approach is to replace the current control with a multi-position rotary switch. A 12-position switch will allow selection from 12 different print densities and will require 12 resistors and some wire.

Another approach is more sophisticated, utilizing four CMOS ICs in an external circuit. It does, however, allow software selection from seven print densities. By including the LPRINT CHR$(X) statement in your program (where X can vary from one to seven), the print density will be changed automatically. Any number of LPRINT CHR$(X) statements can be included in the program to continually modify the print density.

The two approaches can be combined in a small cabinet, and a single DPDT switch can select either the hardware (rotary switch) or software (external circuit) control.

The Hardware Approach

The service manual provided with your printer contains a wealth of information. One such tidbit is the printer schematic diagram. A look at this diagram reveals that the print density is a direct function of the frequency of oscillation of ME1, a 555 timer IC. This frequency is varied by adjusting the 50-Kohm potentiometer R25. An alternate method is to replace R25 with a rotary switch that selects discrete resistance values. Since any given switch postion will always provide the same resistance value, the oscillation frequency (and thus print density) will always be the same.

The diagram for this simple

—HARDWARE MOD

S1 = Single pole, 12-position rotary switch (such as Radio Shack 275-1385)
R = 4.7 Kohm, 1/4 watt, ± 10% resistor (12 Required)
Misc: four-ft. wire (approximate), solder, etc.

—SOFTWARE MOD

R1 = 8.2 Kohm, 1/4 watt, ± 10% resistor
R2 = 15 Kohm, "
R3 = 27 Kohm, "
R4 = 10 Kohm, "
D1,D2 = 1N4138 or 1N914 silicon switching diode
IC1 = 4001 quad 2-IN NOR CMOS IC
Ic2,Ic3 = 4013 dual D flip flop CMOS IC
IC4 = 4016 quad analog switch CMOS IC
Misc: 16-ft. wire (approximate), PC or wire wrap board, four IC sockets, solder, etc.

Table 1. List of Materials

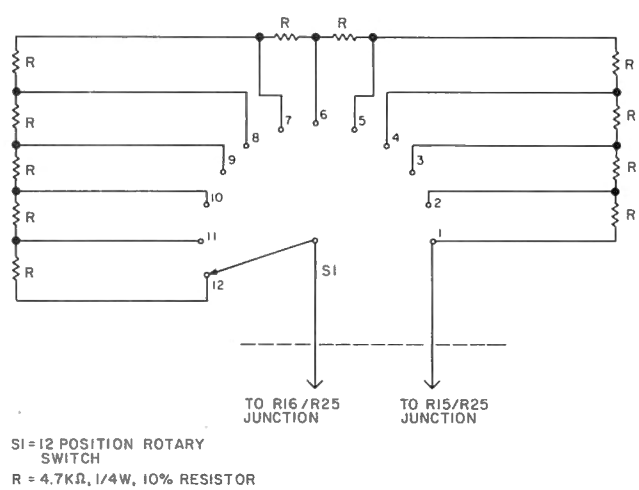


Fig. 1.

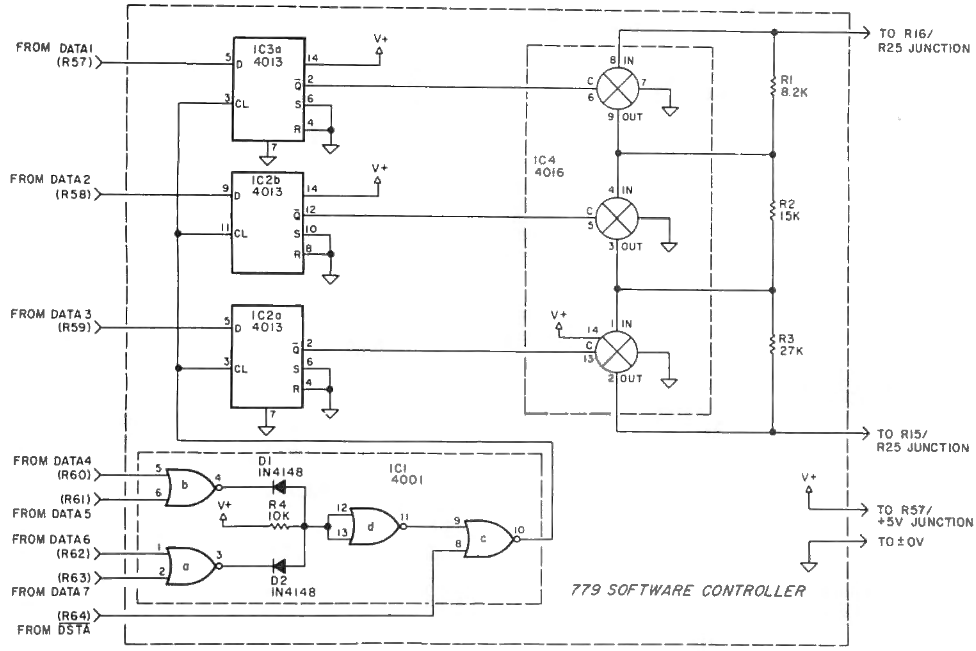


Fig. 2.

modification is shown in Fig. 1. Note that all resistors are 4.7 Kohms, 1/4 watt, ±10 percent. As the switch is rotated from position one to position 12, the total resistance presented to the printer increases in 4.7 Kohm steps until, at position 12, the total of 56.4 Kohms produces the largest print density.

The modification begins by removing the two screws securing the printer's rear cover. Next, remove the two screws securing the PC board and the two screws securing the heat sink attached to the rear of the PC board. (Removal of the PC board is pictured in the service manual.) Finally, remove all connectors to the PC board (they are keyed, so you cannot reconnect them improperly) and remove the PC board from the printer. Unsolder and remove the print density pot R25. Solder a long length (two feet or so) of wire to each of the two R25 PC pads as indicated in Fig. 6. Then reinstall the PC board. Pass the wires through the former access hole for R25 in the rear cover and secure the cover.

Wire a 12-position rotary switch as indicated in Fig. 1. Connect either lead form the PC board to the rotary switch's rotor contact, and the other lead to the switch lug corresponding to position 1. You may fabricate a switch holder from a small piece of aluminum and mark the positions with transfer type lettering. Mounting the switch near the front of the printer makes it easily accessible.

Prior to printing, rotate the switch to the appropriate print density setting. Your program may include a variation of the program line: 5000 CLS:INPUT" PLACE PRINT DENSITY TO 9. PRESS ENTER WHEN READY"; BL$. In this way, you will be alerted to adjust print density before printing begins. An example of the different print densities are provided in Fig. 3.

The Software Approach

It is more desirable to allow the program to directly control print density without your intervention. The circuit shown in Fig. 2 will automatically change the print density when the LPRINT CHR$(X) statement is executed. As indicated in the service manual, a print command causes the computer to place data on lines 1 through 7. After about one µs, the computer pulses the data strobe line low to alert the printer to accept the data.

Since the ASCII codes 1 through 7 have no function in the 779, they can be used for print density control. In this mod, data lines one, two and three are connected to the D inputs of the 4013 CMOS D flip flop's ICs. When the common clock line strobes high, the data is latched into the flip flop. It will remain there until another clock pulse is generated. Data lines four through seven are connected to NOR gates in the 4001 quad 2-In CMOS NOR IC. If any of the data four through data seven lines go high, the output of IC1C is held low. When the data strobe line pulses low, the IC1C output will not change. Thus, when a character code greater than seven is on the data lines, the clock is inhibited. If the data character code is seven or less (data four through data seven equals zero), the output of IC1C will generate a positive clock pulse when the data strobe line pulses low. This will latch in the new information.

The output of the three flip flop latches controls three CMOS analog switches (3/4 of a 4016 CMOS quad analog switch). For a 1 input data, the latch output will be a 0. This 0 opens the switch so that the timing resistor paralleling the switch is con-

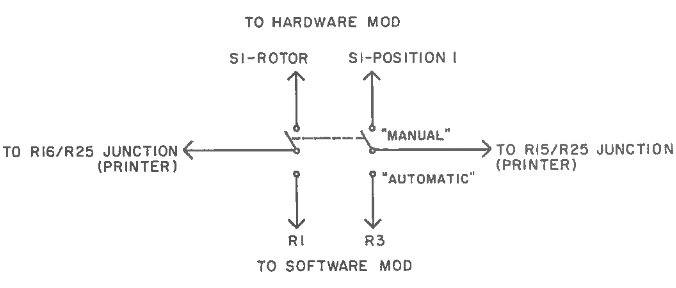


Fig. 2a

```
THIS IS AN EXAMPLE OF PRINT DENSITY # 1
THIS IS AN EXAMPLE OF PRINT DENSITY # 1
THIS IS AN EXAMPLE OF PRINT DENSITY # 2
THIS IS AN EXAMPLE OF PRINT DENSITY # 2
THIS IS AN EXAMPLE OF PRINT DENSITY # 3
THIS IS AN EXAMPLE OF PRINT DENSITY # 3
THIS IS AN EXAMPLE OF PRINT DENSITY # 4
THIS IS AN EXAMPLE OF PRINT DENSITY # 4
THIS IS AN EXAMPLE OF PRINT DENSITY # 5
THIS IS AN EXAMPLE OF PRINT DENSITY # 5
THIS IS AN EXAMPLE OF PRINT DENSITY # 6
THIS IS AN EXAMPLE OF PRINT DENSITY # 6
THIS IS AN EXAMPLE OF PRINT DENSITY # 7
THIS IS AN EXAMPLE OF PRINT DENSITY # 7
THIS IS AN EXAMPLE OF PRINT DENSITY # 8
THIS IS AN EXAMPLE OF PRINT DENSITY # 8
THIS IS AN EXAMPLE OF PRINT DENSITY # 9
THIS IS AN EXAMPLE OF PRINT DENSITY # 9
THIS IS AN EXAMPLE OF PRINT DENSITY # 10
THIS IS AN EXAMPLE OF PRINT DENSITY # 10
THIS IS AN EXAMPLE OF PRINT DENSITY # 11
THIS IS AN EXAMPLE OF PRINT DENSITY # 11
THIS IS AN EXAMPLE OF PRINT DENSITY # 12
THIS IS AN EXAMPLE OF PRINT DENSITY # 12
```

Fig. 3 Example of Manually Selectable Print Densities

```
THIS IS AN EXAMPLE OF PRINT DENSITY # 1
THIS IS AN EXAMPLE OF PRINT DENSITY # 1
THIS IS AN EXAMPLE OF PRINT DENSITY # 2
THIS IS AN EXAMPLE OF PRINT DENSITY # 2
THIS IS AN EXAMPLE OF PRINT DENSITY # 3
THIS IS AN EXAMPLE OF PRINT DENSITY # 3
THIS IS AN EXAMPLE OF PRINT DENSITY # 4
THIS IS AN EXAMPLE OF PRINT DENSITY # 4
THIS IS AN EXAMPLE OF PRINT DENSITY # 5
THIS IS AN EXAMPLE OF PRINT DENSITY # 5
THIS IS AN EXAMPLE OF PRINT DENSITY # 6
THIS IS AN EXAMPLE OF PRINT DENSITY # 6
THIS IS AN EXAMPLE OF PRINT DENSITY # 7
THIS IS AN EXAMPLE OF PRINT DENSITY # 7
```

Fig. 3a Example of Software Selectable Print Densities

```
10 REM***************************************************************
20 REM*******    PROGRAM TO DEMONSTRATE SOFTWARE CONTROL    *******
30 REM*******     OF CENTRONICS 779 SERIAL LINE PRINTER     *******
40 REM*******      USING THE 779 SOFTWARE CONTROLLER        *******
50 REM*******    JAMES J. BARBARELLO    12 OCTOBER 1979     *******
60 REM***************************************************************
70 REM
80 FOR I=1 TO 7
90 FOR J=1 TO 2
100 LPRINT CHR$(I)
110 LPRINT "THIS IS AN EXAMPLE OF PRINT DENSITY #";I
120 NEXT J,I
```

Fig. 4.

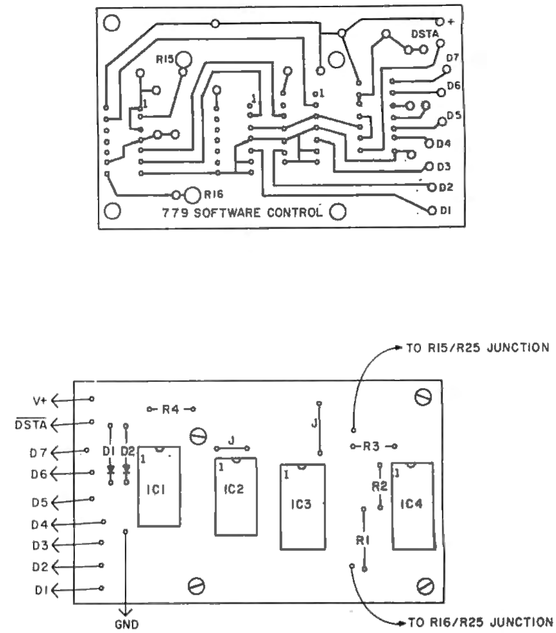


Fig. 5

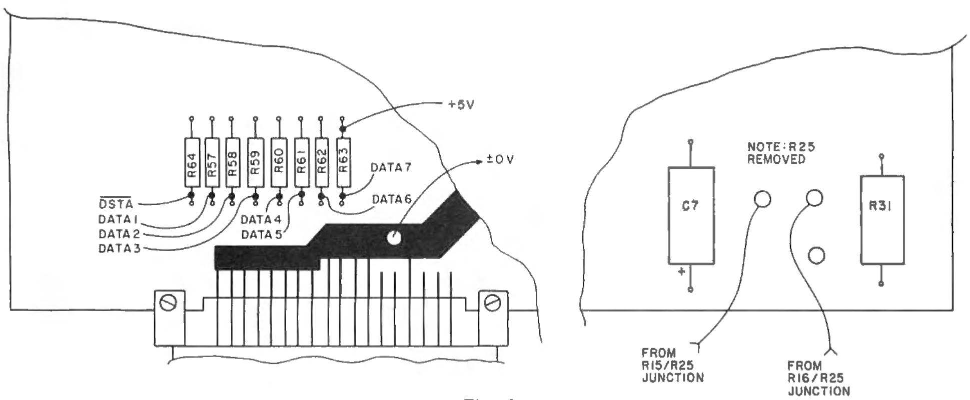


Fig. 6

```
                         80  MICROCOMPUTING
                      PETERBOROUGH, N. H.  03458

TO:
JIM BARBARELLO
R. D. #1 BOX 241H TENNENT ROAD
ENGLISHTOWN N. J.  07726

DEAR JIM:

     WE WOULD LIKE TO INFORM ALL THE HOBBYISTS IN ENGLISHTOWN THAT OUR MAGAZINE
IS OFFERING A SPECIAL DISCOUNT FOR A LIMITED TIME TOACQUAINT YOU WITH AN EXCITING
```

Fig. 7 Example of Form Letter Using Software Print Density Control

```
10 LPRINT " ";CHR$(10):LPRINT CHR$(138)
20 LPRINT CHR$(7):LPRINT TAB(23);"80  MICROCOMPUTING"
30 LPRINT CHR$(3):LPRINT TAB(38);"PETERBOROUGH, N. H.  03458"
50 LPRINT STRING$(3,138)
60 LPRINT CHR$(2):LPRINT "TO:"
70 READ A$,B$,C$
80 LPRINT CHR$(5):LPRINT A$:LPRINT B$:LPRINT C$:LPRINT STRING$(2,138)
90 FOR I=1 TO 12:IF MID$(A$,I,1)=" " THEN I=I-1 ELSE NEXT
95 LPRINT CHR$(7)
100 LPRINT "DEAR ";LEFT$(A$,I);":":LPRINT CHR$(4):LPRINT CHR$(138)
110 FOR I=1 TO 30:IF MID$(C$,I,1)=" " THEN I=I-1 ELSE NEXT
120 LPRINT "     WE WOULD LIKE TO INFORM ALL THE HOBBYISTS IN ";LEFT$(C$,I);" THAT OUR MAGAZINE"
120 LPRINT "IS OFFERING A SPECIAL DISCOUNT FOR A LIMITED TIME TOACQUAINT YOU WITH AN EXCITING"
500 DATA JIM BARBARELLO,R. D. #1 BOX 241H TENNENT ROAD,ENGLISHTOWN N. J.  07726
```

Fig. 8 Program Used to Produce the Form Letter

nected alone in the timing resistor chain R1, R2, R3. Conversely, a zero data input causes the corresponding switch to be turned on, effectively shorting the associated timing resistor. In this way, seven discrete timing resistance values (binary 001 through 111) can be realized. The resistor values indicated provide discrete resistance steps of approximately eight Kohms and maintain the 50-Kohm total resistance formerly provided by R25. Power for the circuit is provided by the printer.

The circuit shown in Fig. 2 can be reproduced on a wire-wrap board, or a PC board can be fabricated using the pattern provided in Fig. 5. In any event, sockets should be used for the ICs. Follow the procedure outlined above in the hardware approach for removing the PC board, R25 and attaching the two new leads in R25's former position (see Fig. 6). Place the printer PC board, component side up, in front of you with the printer cable connector facing you. Tack solder a long length (one foot or so) of wire to each resistor and the PC foil trace as shown in Fig. 6. After identifying each lead, resecure the PC board, route the wires through the printer rear cover and secure the cover. Finally, connect the identified leads to the modified PC board. The modified PC board should be housed in a suitable external case.

The use of the software mod and an example of the different print densities obtainable are provided in Figs. 4, 7 and 8.

Combining the Two

Since both modifications connect to the R15/R25 and R16/R25 junctions, either can be selected with a DPDT switch. The schematic diagrams of such a switching system are shown in Fig. 2a.

Before You Begin

If your printer is less than 90 days old, it carries a warranty. Any modification to the printer could void that warranty. ■

Computerization of the sport of kings.

Equine Equation Evaluation

Raymond J. Herold, CCP
8363 Shady Grove Circle
Manassas, VA 22110

You can go to any racetrack in the country and eavesdrop on the inevitable debates going on among the small groups of horseplayers, and the topic is always the same. George likes horse number three because it has the best clock times. Fred likes the number five because of its last race. Tom thinks a betting coup is in the works involving number seven. John insists that number nine is the class horse in the race and his friends are wasting their time. Once the race is over and the winner determined, the horseplayer with the winning ticket gladly recaps the reasons for his brilliant selection. Yet, no one is listening because there's only 15 minutes before the next race, and another selection must be made. The debate starts anew.

Although not generally known, horse racing is the most widely attended spectator sport in the country. Attendance in 1979 was over 77 million, with baseball a distant second. In the case of football, basketball and other sports, predicting a winner means sitting through three hours of play before knowing the result. With horse racing, the analysis, selection and result is accomplished in rapid-fire succession usually 10 times or more in three hours.

This program will hopefully aid each individual horseplayer in making more informed decisions during his or her excursions to the racetrack.

```
Enter Race Information
Race Number? 8
Number Of Horses In Race? 9
What Is The Distance For This Race? 7F
What Is The Track Record For This Distance? 121
Use Class Ratings In The Analysis—(Y) or (N)? Y
Use Track Variants In The Analysis—(Y) or (N)? Y
Use Subjective Data In The Analysis—(Y) or (N)? Y
Use Workout Times In Analysis—(Y) or (N)? Y
Is The Above Data Correct—(Y) or (N)? Y
—Track Bias Adjustment—
(0) = None
(1) = Inside
(2) = Middle
(3) = Outside
Which One? 1←
```

Fig. 1

```
.....Horse Number 1.....
Number Of Times Raced This
   Year? 14
Number Of Times Finished 1st
   This Year? 2
Number Of Times 2nd Or 3rd
   This Year? 9
Total Earnings For The Year?
   20235
Enter Weight To Be Carried?
   112←
```

Fig. 2

The system is primarily based on the philosophy of speed handicapping. However, provisions have been included to take into account all factors that could influence the outcome of a race. Through the use of optional data entry categories, handicappers who wish to account for criteria such as class, trainer and jockey ratings, track biases and stretch running horses will be able to do so. As with any type of forecasting, the more data available for the analysis, the more accurate the result. The use of different options also provides each handicapper with the ability to tailor use of the program along lines approaching his own handicapping techniques.

The program was written for use on a TRS-80 Level II, with a minimum of 16K. The Microsoft Basic code should be relatively easy to transfer to other machines with a little effort. Data entry is self-explanatory, with examples given in most cases. Editing routines have been incorporated into the program to ensure data integrity whenever possible. The ability to read the charts in the daily racing form is assumed.

Handicapping Strategy

The nucleus of the analysis done by this program is based on speed figures. However, any kind of information given to the program is acted upon. To paraphrase an old saying: "Handicappers do not live by speed alone". Criteria such as class and track bias will have a definite effect on the outcome.

The biggest problem with most computerized handicapping systems is that they tend to be too narrow in approach. In an attempt to speed data entry for a race, some systems only require speed rating and distance. Others only accept data for the last race the horse ran. This kind of system disregards such factors as weather and track conditions, among others. If horse A runs six furlongs in 1:12 on a

SUPER DISCOUNTS ON ALL COMPUTER EQUIPMENT

16K MEMORY....................ONLY $19.95
For **TRS-80** Keyboard or Expansion interface. KEYBOARD requires jumpers: $2.00 Extra. These are 200 ns tested RAM for the **TRS-80, APPLE** or **EXIDY.**

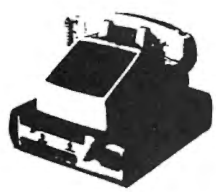
DISK DRIVES for the TRS-80 OR PMC-80:

All of our drives come complete with power supply and chassis. They may be used with existing Radio Shack drives on the same cable! 40 track drives store 102K bytes single density, and 175K double density. 80 track drives have 175K single density and 345K double density! All drives guaranteed 90 days, one year on power supply.

40 track MPI drives $319.95
40 track TEAC drives $315.95
40 track TANDON drives $319.95
80 track MPI drives $449.95
80 track TEAC drives $429.95
2 drive cable .... $ 25.95
4 drive cable .... $ 39.95
NEWDOS 80 VER. II $139.95
NEWDOS 80 VER. I Special sales price.
Limited Quantity $89.95

PERCOM'S DOUBLER II
for double density operation! $169.95
The **DOUBLER** works with the **TRS** or **PMC** expansion interfaces to allow you to use your drives in double density! You may still operate your drives as single density also! Comes with **DBLDOS** operating system which allows you to transfer single density files to double and vice versa! *GREAT BUY!*
DISKETTES: VERBATIM DATALIFE! *BOX OF TEN*
SOFT OR HARD SECTORED 5¼" $32.50
WE HAVE DRIVES AND CONTROLLERS FOR THE MODEL III. CALL FOR PRICES!!!

MODEMS AND TELE-COMMUNICATIONS



LYNX Telecommunication system for the TRS-80 and PMC-80 .... $279.95
Includes terminal software and connections for instant hookup! Can be connected to the **TRS-80** or **PMC-80** with or without an expansion interface!

LYNX now comes standard with auto answer/auto dial & all hardware for Mod I & III! Also includes host program. *GREAT BUY!*

'TARICONNECTION: Modem for 400/800, complete with software! $249.00
APPLECONNECTION: Modem for APPLE II $279.95

COMPLETE SYSTEMS:

PMC-80, 16K LEVEL II COMPUTER....................$739.00
The **PMC-80** is a work alike to the **TRS-80** mod I computer! Comes with Microsoft's **BASIC** in ROM. Built in cassette. 12" video monitor. Expandable to 48K.
Compatible to All **TRS-80 MOD I Programs.**
PMC 80 without monitor $595.00
RF—MOD for PMC to TV hookup $39.95
PMC-80 EXPANDER 100 SYSTEM $499.95
INCLUDES: 32K memory, Parallel printer driver, Disk controller. Fully compatible with **TRSDOS, NEWDOS, VTOS,** and all other **TRS-80 Mod I disk software!**

New New PMC-81 Computer is here..........................CALL

APPLE II COMPUTERS 48K $1299.00
ATARI 400 .... $ 495.95
ATARI 800 COMPUTER .... $ 795.00
ZENITH Z-89 48K, 1 DISK ALL IN ONE COMPUTER $2495.00

VIDEO MONITORS

LEEDEX 100 12' B/W MONITOR $159.95
SANYO 9' B/W MONITOR $199.95

PRINTERS

OKIDATA MICROLINE 80 All OKI'S have TRS-80 Graphics!!! $420.00
Comes with friction and pin feed upper/lower case. Graphics.
EPSON MX-80 PRINTER: Word Processing Quality Printout with Graphics! .... Call
NEC SPINWRITER with Tractor Feed ... $2995.00
DIABLO MODEL 1630 with Tractor $2695.00
UNIVERSAL PRINTER STANDS .... $ 94.50

Other Accessories:

SUP-R-MOD RF Modulator for APPLE ... $29.95
ARCHBOLD SPEED-UP MOD FOR TRS MOD I. Allows up to 300% increase! $45.00
MICROSOFT BASIC Decoded and other mysteries $29.95
TRS-80 DISK and other mysteries $18.95
ZBASIC BASIC COMPILER for MOD I and MOD III TRS-80 and PMC-80 Increase basic program speeds by up to 200 times!!!!!
Tape .... **$79.95** **Disk** **$89.95**

Both For Only $99.95 Specify MOD I or MOD III

WE HAVE HUNDREDS OF PROGRAMS FOR ALL THE POPULAR COMPUTERS! CALL FOR OUR FREE CATALOG

Personal and Business Checks take 3 weeks to clear. For fastest delivery use Certified Checks, Money Orders, Credit Card, or C.O.D

We will try and beat any published price on any system! Call!

SIMUTEK COMPUTER PRODUCTS ™ ✓19

4877 E. Speedway Blvd
Tucson, Arizona 85712
(602) 323-9391 Technical Questions

CALL TOLL FREE TO ORDER
(800) 528-1149
Please mention this magazine when ordering
ARIZONA RESIDENTS ADD 6% SALES TAX.
WE TAKE VISA, MASTER CARD.

✓See List of Advertisers on page 354

fast track, is he really better than horse B who ran six furlongs in 1:13 on a muddy track? Probably not. You may spend a little more time entering data into this program, but you will get better results.

The other major problem with most commercially available handicapping programs is their price tag. It may take several hours to enter this program into your computer, but it's free!

The philosophy behind this program is to account for all pertinent factors. It examines all areas of an individual horse's past performance and develops a composite of that horse. This composite is compared to those of the other horses in the race. In order to make the composite for each horse as meaningful as possible, the following strategy should be followed when entering information for each horse:

- Whenever possible, enter the results of at least three races for each horse. Four or five would be better. The program allows up to 10. This takes longer but it helps to offset special-case conditions.
- Use the track variant option.
- Try to use results from races that were run on fast or good tracks. The track variant will adjust for bad conditions, but races run in the slop aren't indicative of a horse's capabilities.
- *If the track* you attend has a known bias, use the track bias option to adjust.
- Use the class rating option where applicable.
- Remember, you can input too little information but you can't input too much!

Editing

Much of the data entered into the program is edited for valid content. This is done in an attempt to avoid the "garbage in-garbage out" computer tradition. For example, if you wanted to enter a horse's finish position as 10, but the key bounced and you got 100, this could certainly produce some curious results. In a case such as this the program would recognize an obvious error and simply reprint the prompt for that category. You would then re-enter the correct reply. For most categories you will be asked if the data you have just entered is correct. If it isn't, simply reply N to re-enter.

A more subtle error, with devastating effects, can occur when entering the critical time/distance relationship. Take the example of a horse that ran a six-furlong race in 1:13.2. When you put the information into the

Distance	*Enter*
2 furlongs	2F
3 furlongs	3F
3 1/2 furlongs	3.5F
4 furlongs	4F
4 1/2 furlongs	4.5F
5 furlongs	5F
5 1/2 furlongs	5.5F
6 furlongs	6F
6 1/2 furlongs	6.5F
7 furlongs	7F
7 1/2 furlongs	7.5F
7 3/4 furlongs	7.75F
1 mile	1M
1 mile 40 yards	1M 40Y
1 mile 70 yards	1M 70Y
1 1/16 mile	1 1/16M
1 1/8 mile	1 1/8M
1 1/4 mile	1 1/4M
1 1/2 mile	1 1/2M
1 5/8 mile	1 5/8M
1 3/4 mile	1 3/4M
2 miles	2M

Table 1. The program will accept race distance of two furlongs to two miles. The following examples show how to enter the different distances.

program you accidentally enter the time as 1:33.2—a common mistake. The short edit at line 1160 determines the time of the race on a per furlong basis. If the per furlong is outside of an acceptable range, a message telling you that a possible mistake has occurred appears. It will show you the time and distance entered. If the time or distance is an error, responding R to the query "Enter Another Past Performance For Horse Number n" allows you to re-enter the previous past performance.

Using the Program: Race Information

This section (lines 550–588) sets up the basic parameters for each race. Fig. 1 shows the prompts and replies used. The prompts for race number and the number of horses are self-explanatory. The distance for the race to be run is seven furlongs. Table 1 shows how the different distances are to be entered. The track record is used to compute an actual finish time for each horse. The correct method for entering times is shown in Table 2.

The next four lines determine which of the optional categories allowed will be used. In the example, all four are to be used in the analysis. Whether or not a particular track has a bias is generally subjective in nature. For example, if the response to the subjective data prompt was Y, an adjustment for this situation is allowed. Depending on which portion of the track contains the suspected bias, lines 582–586 make the appropriate adjustment.

Set-up for Each Horse

This section allows the entry of basic information for each horse in the race. Fig. 2 shows how this information is entered. As can be seen, this information is obtained from Fig. 3 which charts past performance. This chart also corresponds to the way information is presented in the daily racing form.

Past Performances

You can enter from one to ten past performances for each horse. One is required. Entering four or five will optimize the analysis. The top portion of Fig. 3 shows a sample past performance chart. The bottom portion shows how the information would be entered into the computer. In this sample, the race distance for the past performance being entered was six furlongs. The winner's finishing time was 1:13.2 (Tables 1 and 2). Our sample horse finished second, a head behind the winner. The following shows how lengths should be entered if the horse is not the winner:

Lengths Behind	Enter
2	2
1½	1.5
3¾	3.75
½	.5
¼	.25
neck	.2
head	.1
nose (or winner)	0

In the last line you will be asked to enter the speed rating, and track variant (if used).

Subjective Data

The idea of class is a rather controversial subject to most handicappers. Some insist that the use of class is the only rational way to handicap a race. Others insist just as stubbornly that it is an absolutely worthless idea. The answer probably lies somewhere in between. Because of this, class rating was made an optional category.

The top portion of Fig. 4 shows the range of variation allowed. The movement in class

Time	*Enter*
:37	*37*
:42 2/5	*42.4*
1:09	*109*
1:12 3/5	*112.6*
1:36 1/5	*136.2*
2:10	*210*
2:20 3/5	*220.6*
3:12	*312*
3:40 1/5	*340.2*

Table 2. All race times must be entered in decimal form. Fifths of seconds should be converted to tenths. This table shows the correct method for entering times.

```
                 Ch. f. 4, by Micro Boy-Buffer, by Memory Chip
                      Br.-Micro Stables                    St. 1st 2nd 3rd  Amt.
                       Tr.-W. Green        112       1979  14   2   4   5  20,235

11Nov79-7Blr fst 6f  :23  :47 1/5  1:13 1/5  Allowance  7 7 5 4 4  2hd Herold R H b 112  7.10  74,33
 2Nov79-6Lrl fst 7f  :23  :46 3/5  1:25 2/5  Allowance  2 9 8 8 6  3nk Herold R H b 114  4.50  81,16

                -- Sample Distance Chart --      00:08:52

              3F     5F     7F      1M 40Y   1 3/8M
              3.5F   5.5F   7.5F    1M 70Y   1 1/2M
              4F     6F     7.75F   1 1/8M   1 3/4M
              4.5F   6.5F   1M      1 1/4M   2M

= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

<<<<< Past    Performance — Horse    Number    2 — Race    #    1 >>>>>
Race Distance  — (Use Table Above)...........................................? 6F
Winner's Finish Time  — (EX: 112.4)..........................................? 113.2
Horses' Finish Position (EX: 4)..............................................? 2
Length's Behind Winner  — (EX: 4.25  —  0 IF NONE)...........................? .1
Speed Rating and Track Variant  — (EX: 76,12)................................? 74,33←
```

Fig. 3. Sample Distance Chart

```
                    -- Class Table --
          < >  A Lot          .
          < >  Moderate       . . ------>> Up In Class
          < >  Slight         .
          < >  Equal Class
          < >  Slight         .
          < >  Moderate       . . ------>> Down In Class
          < >  A Lot          .
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
Using the Table Above, Indicate the Horse's Class Rating as
Compared to the Other Horses in the Race.
? 0←

                    -- Subjective Data --
Is Jockey in Top 5 Ratings  — <Y> OR <N>? N
Is Trainer in Top 5 Ratings  — <Y> OR <N>? N
Is Horse Showing Consistent Improvement  — <Y> OR <N>? N
Does Horse Close Consistently In Stretch Run  — <Y> OR <N>? Y
Has Horse Been Idle for More Than 30 Days  — <Y> OR <N>? N←
```

Fig. 4

is not as difficult to determine as the degree of that movement. Whether a horse is changing class slightly, moderately or a lot is left to the experience of the individual handicapper using this program. The bottom portion of Fig. 4 shows the other subjective data classification. It should be pointed out that only a Y response has any effect on the analysis. A response of N leaves the ratings unchanged.

Interpreting The Results

Once all the data for an upcoming race has been entered, the program goes into the analysis phase of its operation (lines 5000-5999). The program compares each horse's relative standing to the other horses in the race for each of the primary categories (5020-5096). Points

H#	COMP. TIME	AVE. FIN.	%WPS	EARN/ START	SR/TV AVE.	COMP.	POINTS	ODDS
1	127.1	4.8	13.3	396	88.8	45.0	5.0	20-1
2	125.7	3.0	78.6	1445	102.0	5.3	32.8	5-2 ***
3	126.5	2.5	42.9	1894	100.5	7.7	22.6	4-1
4	127.0	1.8	66.7	1414	102.0	7.5	23.2	4-1 *
5	128.6	4.3	52.9	1134	89.0	15.2	11.5	7-1
6	126.9	4.0	47.1	1162	98.0	10.3	17.0	5-1
7	126.7	4.0	44.4	1061	100.3	9.8	17.8	5-1
8	126.3	2.0	42.9	1067	102.7	6.8	25.9	7-2 **
9	126.9	2.3	36.4	970	102.7	9.1	19.3	9-2

Wagering

$5.00 On # 2 Win & Place
$2.00 On # 8 Show

Exactas
2—8 2—4

Fig. 5. Comparative analysis for Race #8

Program Listing 1

```
10 ' THOROUGHBRED HANDICAPPING
20 ' COPYRIGHT 1979
30 ' RAY HEROLD
40 ' 8363 SHADY GROVE CIR.
50 ' MANASSAS VA.   22110
60 '
100 CLEAR200:DIM HD(20,11),WC(20),WN(20),OD$(20)
200 CLS:PRINT@400,"THOROUGHBRED HANDICAPPING":FORX=1TO1000:NEXTX

500 CLS:PRINT:PRINT"ENTER RACE INFORMATION":PRINT
510 INPUT"RACE NUMBER";RN
515 INPUT"NUMBER OF HORSES IN RACE";NH:IFNH<5ORNH>20THEN515
520 INPUT"WHAT IS THE DISTANCE FOR THIS RACE";WD$:GOSUB1300:DR=W
D:IFDR=0THEN520
525 INPUT"WHAT IS THE TRACK RECORD FOR THIS DISTANCE";TR:IFTR<15
ORTR>400THEN525
530 RS=0
535 IFTR>99THENRS=RS+60:TR=TR-100:GOTO535
540 RS=RS+TR
545 S1=0:S2=0:S3=0:S4=0:S5=0
550 O$="":INPUT"USE CLASS RATINGS IN THE ANALYSIS - <Y> OR <N>";
O$:IFO$="Y"THENS2=1:ELSEIFO$<>"N"THEN550
560 O$="":INPUT"USE TRACK VARIANTS IN THE ANALYSIS - <Y> OR <N>"
;O$:IFO$="Y"THENS3=1:ELSEIFO$<>"N"THEN560
570 O$="":INPUT"USE SUBJECTIVE DATA IN THE ANALYSIS - <Y> OR <N>
";O$:IFO$="Y"THENS4=1:ELSEIFO$<>"N"THEN570
572 O$="":INPUT"USE WORKOUT TIMES IN ANALYSIS - <Y> OR <N>";O$:I
FO$="Y"THENS5=1:ELSEIFO$<>"N"THEN572
573 O$="":PRINT:INPUT"IS THE ABOVE DATA CORRECT - <Y> OR <N>";O$
:IFO$="N"THEN500ELSEIFO$<>"Y"THEN573
575 IFS4<>1THEN600
580 CLS:O=0:PRINT:PRINTTAB(5)"-- TRACK BIAS ADJUSTMENT --":PRINT
"<0> = NONE":PRINT"<1> = INSIDE":PRINT"<2> = MIDDLE":PRINT"<3> =
 OUTSIDE":INPUT"WHICH ONE";O
582 IFO=1THENFORX=1TO4:HD(X,10)=HD(X,10)+((NH/2)-X):NEXTX
584 IFO=2THENFORX=5TO8:HD(X,10)=HD(X,10)+2:NEXTX
586 IFO=3ANDNH>=9THENFORX=9TONH:HD(X,10)=HD(X,10)+(NH*.25):NEXTX

588 IFO<0ORO>3THEN580
600 FORCH=1TONH
605   TT=0:RC=0:TP=0:RR=0:CT=0:WO=0:BT=999
610   GOSUB1600:GOSUB1000
615   IFS5=1THENGOSUB1700
616   IFS2=1THENGOSUB1500
620   IFS4=1THENGOSUB2000
690 NEXTCH
900 GOSUB5000:GOSUB4000:GOSUB6000:CLS:PRINT@400,"NEXT RACE...":F
ORX=1TO20:FORY=1TO11:HD(X,Y)=0:NEXTY:NEXTX:GOTO500
```

Program continues

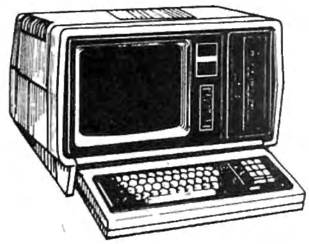
DISCOUNT TRS-80®

BUY DIRECT

COMPUTER SPECIALISTS

CALL US. . . SAVE MONEY

MODEL II



26-4002
64K 1 Drive
$3399.00

26-4160 1 Drive Exp.	$1034.00
26-4161 2 Drive Exp.	1574.00
26-4162 3 Drive Exp.	2114.00
26-4530 Scripsit II.	265.00
26-4512 Profile II.	162.00
26-4511 Visicalc II.	265.00
26-4501 Gen Ledger.	180.00
26-4506 Mail List.	72.00

PRINTERS

CENTRONICS

BEST PRICES

Text Quality Centronics
739-1 Printer. **$729.00**

We carry the full line of TRS-80 Computers. All other software, furniture, and accessories at discount from catalog price. We stock most items to assure you fast delivery and save you money.

26-1140 Expansion Interface.	$249.00
26-1141 16K Exp. Interface.	339.00
26-1142 32K Exp. Interface.	429.00
26-1145 RS232C Board.	84.00
26-1160/1 Mini Disk Drive.	419.00
26-1563 Scripsit-Disk.	79.00
26-1566 Visicalc.	83.00

* * * * * **PRINTERS** * * * * *

26-1155 Quick Printer.	187.00
26-1167 Line Printer VII.	360.00
26-1166 Line Printer VI.	1080.00
26-1158 Daisy Wheel II.	1799.00
26-1168 Line Printer VIII.	720.00

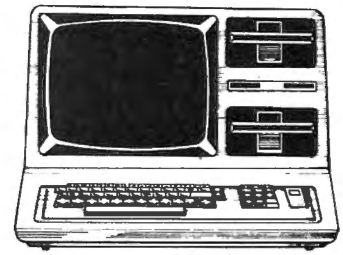
Pocket Computer

26-3501 1.9K P.C.	$188.00
26-3503 Cassette IF.	45.00
14-812 Recorder.	72.00

WRITE US FOR A
FREE CATALOG

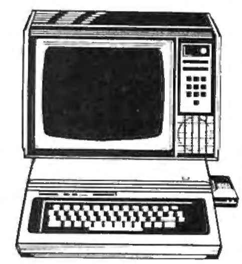
THOUSANDS OF SATISFIED CUSTOMERS

MODEL III



26-1061 4K I.	$609.00
26-1062 16K III.	849.00
26-1066 48K III W/2 Drives, RS232.	2149.00

COLOR



26-3001 4K.	$318.00
26-3002 16K Ext. Basic.	488.00
26-3008 Joysticks.	22.50
26-3010 Color Video.	353.00
26-1206 Recorder.	54.00

commodore

AUTHORIZED DEALER
CALL FOR PRICES

ALL POCKET AND COLOR
COMPUTER SOFTWARE
SOLD AT DISCOUNT

1-800-841-0860 Toll Free Order Entry

MICRO MANAGEMENT SYSTEMS, INC. ✓423

No Taxes on Out Of State Shipments

Immediate Shipment From Stock on Most Items

DOWNTOWN PLAZA SHOPPING CENTER
DEPT. #4
115 C. SECOND AVE. S.W.
CAIRO, GEORGIA 31728
(912) 377-7120 Ga. Phone No. & Export

TRS-80 is a registered trademark of the Tandy Corp.

A copy of the manufacturer's warranty can be obtained free upon specific written request to the Electronic's Department of our Cairo, Georgia Retail Store.

✓See List of Advertisers on page 354

Program continued

```
1000 FORCR=1TO10
1010   CLS:GOSUB11500:PRINT"<<<<< PAST PERFORMANCE  -  HORSE NUM
BER";CH;" -  RACE #";CR;">>>>>"
1110   PRINT@640,"RACE DISTANCE  -  (USE TABLE ABOVE)...........
........";:INPUTWD$
1115   GOSUB1300:RD=WD:IFRD=0THEN1110
1117   FT=-1:PRINT@704,"WINNERS'S FINISH TIME  -  (EX: 112.4)...
................";:INPUTFT
1118   IFFT<=15ORFT>400THEN1117
1120   FP=-1:PRINT@768,"HORSES' FINISH POSITION (EX: 4).........
...............";:INPUTFP
1121   IFFP<0ORFP>25THEN1120
1122   TP=TP+FP
1124   LB=-1:PRINT@832,"LENGTH'S BEHIND WINNER  -  (EX: 4.25  -
 0 IF NONE)...";:INPUTLB
1126   IFLB<0ORLB>50THEN1124
1130   SR=-1:TV=-1:IFS3=1THENPRINT@896,"SPEED RATING AND TRACK V
ARIANT  -  (EX: 76,12)........";:INPUTSR,TV:RR=RR+SR+TV:ELSEPRIN
T@896,"SPEED RATING  -  (EX: 82).............................";
:INPUTSR
1131   IFSR<20ORSR>120THENRR=RR-(SR+TV):GOTO1130
1132   IFS3=1AND(TV<0ORTV>75)THENRR=RR-(ST+TV):GOTO1130
1150   TS=0:HT=FT:GOSUB1200
1160   IFUT/RD<10ORUT/RD>13.5THENCLS:PRINT:PRINT"A POSSIBLE INPU
T ERROR HAS OCCURED.":PRINT:PRINT"THE FOLLOWING WAS ENTERED:":PR
INT"RACE DISTANCE.... ";HD$:PRINT"RACE TIME........";HT:PRINT:IN
PUT"PRESS ENTER";A$
1175   CLS:O$="":PRINT:PRINT"ENTER ANOTHER PAST PERFORMANCE FOR
HORSE NUMBER";CH:INPUT"<Y> OR <N>   -  <R> TO RE-ENTER PREVIOUS
RACE";O$:IFO$="N"THENCR=11ELSEIFO$="R"THENCR=CR-1:RC=RC-1:RR=RR-
(SR+TV):CT=CT-CC:TP=TP-FP:TT=TT-AT
1176   IFO$<>"N"ANDO$<>"Y"ANDO$<>"R"THEN1175
1179 NEXTCR
1180 IFRC>0THENHD(CH,1)=TT/(RC+WO):HD(CH,1)=HD(CH,1)-((HD(CH,1)-
BT)*.25):HD(CH,2)=TP/RC:HD(CH,9)=RR/RC:ELSECLS:PRINT:PRINT"REQUI
RED RACE INFO MISSING FOR HORSE NUMBER";CH:PRINT"RETRY DATA ENTR
Y.":INPUT"PRESS ENTER";A$:CH=CH-1:RETURN
1190 IFHD(CH,2)<3ANDNR<4THENHD(CH,10)=HD(CH,10)-(NH*.2)
1199 RETURN
1200 IFFT>99THENTS=TS+60:FT=FT-100:GOTO1200:ELSETS=TS+FT
1210 UT=TS+(LB/5):IFDR<>RDTHENAT=UTELSETR=UT-((100-SR)/5):AT=UT+
(RS-TR)
1215 IFDR<>RDTHENAT=((AT/RD)*DR):GOSUB1400
1220 TT=TT+AT:RC=RC+1
1230 IFAT<BTTHENBT=AT
1290 RETURN
1300 WD=0:HD$=WD$
1310 IFRIGHT$(WD$,1)="F"THENWD$=LEFT$(WD$,LEN(WD$)-1):WD=VAL(WD$
):IFWD<2ORWD>7.9THENWD=0:RETURN:ELSERETURN
1320 IFRIGHT$(WD$,1)<>"M"ANDRIGHT$(WD$,1)<>"Y"THENRETURN
1325 IFWD$="1M"THENWD=8
1330 IFWD$="1 1/16M"THENWD=8.5
1335 IFWD$="1 1/8M"THENWD=9
1340 IFWD$="1 3/16M"THENWD=9.5
1345 IFWD$="1 1/4M"THENWD=10
1350 IFWD$="1 5/16M"THENWD=10.5
1355 IFWD$="1 3/8M"THENWD=11
1360 IFWD$="1 7/16M"THENWD=11.5
1365 IFWD$="1 1/2M"THENWD=12
1367 IFWD$="1 9/16M"THENWD=12.5
1370 IFWD$="1 5/8M"THENWD=13
1372 IFWD$="1 11/16M"THENWD=13.5
1375 IFWD$="1 3/4M"THENWD=14
1377 IFWD$="1 13/16M"THENWD=14.5
1380 IFWD$="1 7/8M"THENWD=15
1382 IFWD$="1 15/16M"THENWD=15.5
1384 IFWD$="2M"THENWD=16
1386 IFWD$="1M 40Y"THENWD=8.1818
1388 IFWD$="1M 70Y"THENWD=8.3182
1390 RETURN
1400 X1=0:X2=1.3:X3=DR-RD
1410 IFX3<0THEN1450
1420 IFX3>=1THENX1=X1+X2:X2=X2*1.35:X3=X3-1:GOTO1420
1430 IFX3>0THENX1=X1+((X2*X3)+(X3/2))
1440 GOTO1490
1450 X1=1.1*X3
1490 AT=AT+X1
1499 RETURN
1500 CLS:GOSUB11000
1510 CC=0:PRINT"USING THE TABLE ABOVE, INDICATE THE HORSE'S CLAS
S RATING AS":PRINT"COMPARED TO THE OTHER HORSES IN THE RACE."
1520 INPUTCC:IFCC<-3ORCC>3THEN1500
1530 IFCC>0THENHD(CH,10)=HD(CH,10)-(CC+1)
1540 IFCC<0THENHD(CH,10)=HD(CH,10)+ABS(CC-1)
1590 RETURN
1600 CLS:PRINT:PRINT".....HORSE NUMBER";CH;".....":PRINT
1605 EY#=0:NR=0
1610 INPUT"NUMBER OF TIMES RACED THIS YEAR";NR:IFNR<0ORNR>60THEN
1610
1620 INPUT"NUMBER OF TIMES FINISHED 1ST THIS YEAR";HD(CH,3):IFHD
(CH,3)<0ORHD(CH,3)>50ORHD(CH,3)>NRTHEN1620
1630 INPUT"NUMBER OF TIMES 2ND OR 3RD THIS YEAR";HD(CH,4):IFHD(C
H,4)<0ORHD(CH,4)>50ORHD(CH,4)>NRTHEN1630
1640 INPUT"TOTAL EARNINGS FOR THE YEAR";EY#:IFEY#<0OREY#>2000000
THEN1640
1650 IFNR>0THENHD(CH,6)=EY#/NR:HD(CH,5)=((HD(CH,3)+HD(CH,4))/NR)
*100
1660 IFHD(CH,5)>55ANDNR>5ANDHD(CH,3)=0THENHD(CH,10)=HD(CH,10)-(N
H*.4)
1670 IFHD(CH,5)>=60ANDNR<6THENHD(CH,10)=HD(CH,10)-(NH*.15)
1675 IFHD(CH,5)>=50ANDNR<3THENHD(CH,10)=HD(CH,10)-(NH*.15)
1680 INPUT"ENTER WEIGHT TO BE CARRIED";HD(CH,7):IFHD(CH,7)<80ORH
D(CH,7)>160THEN1680
1690 RETURN
1700 CLS:PRINT:PRINT"ENTER RECENT WORKOUT TIME FOR HORSE NUMBER"
;CH:O$="":INPUT"<Y> OR <N>";O$
1710 IFO$="N"THENRETURN
1720 IFO$<>"Y"THEN1700
1730 WT=0:WD$="":PRINT
1740 INPUT"TIME........";WT
```

Program continues

are allocated on this basis, with each category being weighed according to its importance on the outcome of the race. The result of this analysis is displayed on the screen as a row of statistics for each horse (Fig. 5). The information listed includes horse number, speed rating in minutes, seconds and tenths, average finish position, earnings per start, total point accumulation, a numerical comparison ratio, and computed odds. In addition, the horse with the highest rating will have *** next to his odds. The second will have **, and the third *. If you route the results to a line printer, the above information will be listed along with the average speed rating/track variant total for each horse.

The speed rating is the projected time for each horse to cover the distance. It should be emphasized however, that the horse with the best speed rating time is not always the projected winner. In fact, he may not even be in the top three. This is because all aspects of a horse's past performance are taken into account. The horse may have earned a good speed rating on the basis of one, never to be repeated, exceptional performance. The keyword here is consistency!

The columns for average finish, percent in the money, and earnings per start are self-explanatory. The column for points is the horse's accumulated total for all areas of performance. The highest point total represents the projected winner. It is important to understand that each horse's point total is useful only for the current race. He would probably earn a completely different total when matched against a different field of horses.

The column titled Comp is used as a comparison ratio in the wagering and odds calculation routines. It is simply the total points for each horse divided into the total points allocated for the race. The computed odds represent what the computer thinks the chance of each horse winning actually is. They may or may not be similar to the odds at race time.

The wagering suggestions are only meant to be used as a guide. What the suggestions tell you is how strongly the computer feels about its selected winner. If, for example, the computer suggests betting $2 to place and show on its projected winner, it is telling you that the selection is somewhat shaky. On the other hand, if the computer suggests betting $5 to win and place on the horse, it is telling you that it has a great deal of confidence in the selection. For those of you who like to see a visual representation of data, a bar graph routine has been included in the program. The bars will show each horse's point total.

Horse Racing Tips

First, ignore most tips! People who insist they have inside information abound at the track like bees around honey. A little common sense is in order here. If I were someone who had true inside information, I'd be damned if I would tell anyone.

Don't bet more money than you can realistically afford to lose. This seems so obvious, yet is ignored by more people who should know better. Along the same line, don't go to the track with the idea of making a killing. You'll be eating hotdogs and beans for the rest of the week. Go to have a good time.

Try to avoid being influenced by the odds. There is a great tendency (experience speaking here) to bet on a horse that has attractive odds. If you like a particular horse, stick with him. Even if his odds go to 2-1, it's better to collect $6 than zilch. It's amazing how large the size of the discarded ticket pile at your feet will get by betting on longshots.

Lastly, if you see ads for handicapping programs promising 80 and 90 percent winners, avoid them like the plague. If I had a program that picked 80 per cent winners, I'd be on my yacht heading for Bermuda. This program, if used correctly, will pick roughly 40 per cent winners. That won't buy you a Cadillac, but it may pay for a couple of those programs you've been wanting.■

Program continues

```
1741 IFWT<15ORWT>400THEN1740
1745 INPUT"DISTANCE....";WD$:GOSUB1300:D=WD:IFD=0THEN1745
1746 WS=0
1750 IFWT>99THENWS=WS+60:WT=WT-100:GOTO1750:ELSEWS=WS+WT
1760 IFDR<>DTHENWS=((WS/D)*DR):RD=D:AT=WS:GOSUB1400:WS=AT
1770 TT=TT+WS:WO=WO+1
1790 RETURN
2000 CLS:PRINT:PRINT"    -- SUBJECTIVE DATA --"
2010 O$="":PRINT:INPUT"IS JOCKEY IN TOP 5 RATINGS  -  <Y> OR <N>
";O$:IFO$="Y"THENHD(CH,10)=HD(CH,10)+(NH*.2):ELSEIFO$<>"N"THEN20
10
2020 O$="":INPUT"IS TRAINER IN TOP 5 RATINGS  -  <Y> OR <N>";O$:
IFO$="Y"THENHD(CH,10)=HD(CH,10)+(NH*.3):ELSEIFO$<>"N"THEN2020
2030 O$="":INPUT"IS HORSE SHOWING CONSISTENT IMPROVEMENT  -  <Y>
 OR <N>";O$:IFO$="Y"THENHD(CH,10)=HD(CH,10)+(NH*.2):ELSEIFO$<>"N
"THEN2030
2040 O$="":INPUT"DOES HORSE CLOSE CONSISTENTLY IN STRETCH RUN -
<Y> OR <N>";O$:IFO$="Y"THENHD(CH,10)=HD(CH,10)+(NH*.3):ELSEIFO$<
>"N"THEN2040
2050 O$="":INPUT"HAS HORSE BEEN IDLE FOR MORE THAN 30 DAYS - <Y>
 OR <N>";O$:IFO$="Y"THENHD(CH,10)=HD(CH,10)-(NH*.3):ELSEIFO$<>"N
"THEN2050
2090 RETURN
4000 B1$=" WIN PLACE & SHOW":B2$=" WIN & PLACE":B3$=" WIN":B4$="
 PLACE & SHOW":B5$=" PLACE":B6$=" SHOW"
4005 FORX=1TO4:C$(X)="":NEXTX
4010 B7$="$10.00 ON # ":B8$="$5.00 ON # ":B9$="$2.00 ON # "
4020 GOSUB4100:GOSUB4200:GOSUB4300:GOSUB4400
4090 RETURN
4100 IFHD(R1,11)<=NH*.6ANDHD(R1,10)>HD(R2,10)+(NH*.9)ANDHD(R1,3)
>1THENC$(1)=B7$+STR$(R1)+B3$:RETURN
4110 IFHD(R1,11)<=NH*.6ANDHD(R1,10)>HD(R2,10)+(NH*.7)ANDHD(R1,3)
>0THENC$(1)=B8$+STR$(R1)+B2$:RETURN
4120 IFHD(R1,11)<=NH*.63ANDHD(R1,10)>HD(R2,10)+(NH*.5)ANDHD(R1,3
)>0THENC$(1)=B9$+STR$(R1)+B1$:RETURN
4130 IFHD(R1,11)<=NH*.66ANDHD(R1,10)>HD(R2,10)+(NH*.4)THENC$(1)=
B9$+STR$(R1)+B1$:RETURN
4140 IFHD(R1,11)<=NH*.68ANDHD(R1,10)>HD(R2,10)+(NH*.25)THENC$(1)
=B9$+STR$(R1)+B3$
4150 IFHD(R1,11)<=NH*.7ANDHD(R1,10)>HD(R2,10)+(NH*.2)THENC$(1)=B
9$+STR$(R1)+B4$:RETURN
4160 IFHD(R1,10)>HD(R2,10)+(NH*.4)THENC$=B9$+STR$(R1)+B2$:RETURN
:ELSEC$(1)=B9$+STR$(R1)+B4$:RETURN
4200 IFHD(R1,10)>HD(R2,10)+(NH*.25)ANDHD(R2,10)>=HD(R3,10)+(NH*.
5)THENC$(2)=B9$+STR$(R2)+B4$:RETURN
4210 IFHD(R2,10)>HD(R3,10)+(NH*.4)THENC$(2)=B9$+STR$(R2)+B5$:RET
URN:ELSEC$(2)=B9$+STR$(R2)+B6$:RETURN
4300 IFHD(R3,10)>HD(R4,10)+(NH*.3)ANDHD(R3,11)<=NH*.71THENC$(3)=
B9$+STR$(R3)+B6$
4310 RETURN
4400 IFHD(R1,11)<NH*.6ANDHD(R1,10)>HD(R2,10)+(NH*.9)THENC$(4)=ST
R$(R1)+" -"+STR$(R2)+"  "+STR$(R1)+" -"+STR$(R3)+"  "+STR$(R1)+"
 -"+STR$(R4)+"  "+STR$(R1)+" -"+STR$(R5):RETURN
4410 IFHD(R1,11)+HD(R2,11)+HD(R3,11)<NH*2ANDHD(R3,10)>HD(R4,10)+
(NH*.75)THENC$(4)="BOX NUMBER "+STR$(R1)+" -"+STR$(R2)+" -"+STR$
(R3):RETURN
4420 IFHD(R1,11)+HD(R2,11)<NH*1.3ANDHD(R2,10)>HD(R3,10)+(NH*.75)
THENC$(4)=STR$(R1)+" -"+STR$(R2)+"   "+STR$(R2)+" -"+STR$(R1):RE
TURN
4430 IFHD(R1,11)+HD(R2,11)<NH*1.3ANDHD(R2,10)>HD(R3,10)+(NH*.5)T
HENC$(4)=STR$(R1)+" -"+STR$(R2)+"  "+STR$(R2)+" -"+STR$(R1)+"  "
+STR$(R1)+" -"+STR$(R3)+"  "+STR$(R2)+" -"+STR$(R3):RETURN
4440 IFHD(R1,11)<NH*.62ANDHD(R1,10)>HD(R2,10)+(NH*.4)ANDHD(R2,10
)>HD(R3,10)+(NH*.3)ANDHD(R3,10)>HD(R4,10)+(NH*.5)THENC$(4)=STR$(
R1)+" -"+STR$(R2)+"  "+STR$(R1)+" -"+STR$(R3)+"  "+STR$(R2)+" -"
+STR$(R3):RETURN
4450 IFHD(R1,11)<NH*.63ANDHD(R1,10)>HD(R2,10)+(NH*.6)ANDHD(R2,10
)<=HD(R3,10)+(NH*.3)THENC$(4)=STR$(R1)+" -"+STR$(R2)+"  "+STR$(R
1)+" -"+STR$(R3):RETURN
4460 C$(4)="-- NONE --":RETURN
5000 CLS:PRINT@400,"COMPARATIVE ANALYSIS IN PROGRESS..."
5010 FORX=1TONH
5020   FORY=1TONH
5030      IF(HD(X,1)<=HD(Y,1))ANDS3=1THENHD(X,10)=HD(X,10)+1.1
5040      IF(HD(X,1)<=HD(Y,1))ANDS3<>1THENHD(X,10)=HD(X,10)+2.1
5050      IF(HD(X,9)>=HD(Y,9))ANDS3=1THENHD(X,10)=HD(X,10)+1
5060      IFHD(X,2)<=HD(Y,2)THENHD(X,10)=HD(X,10)+.6
5070      IFHD(X,5)>=HD(Y,5)THENHD(X,10)=HD(X,10)+.6
5080      IFHD(X,6)>=HD(Y,6)THENHD(X,10)=HD(X,10)+.6
5090      IFHD(X,7)>HD(Y,7)+3THENHD(X,10)=HD(X,10)-.3
5095   NEXTY
5096 NEXTX
5097 FORX=1TONH
5098   IFHD(X,10)<=0THENHD(X,10)=1
5099 NEXTX
5100 TP=0:FORX=1TONH:TP=TP+HD(X,10):NEXTX
5200 FORX=1TONH:HD(X,11)=TP/HD(X,10):NEXTX
5300 FORX=1TONH:WC(X)=HD(X,11):WN(X)=X:NEXTX
5305 SS=0:FORX=1TONH-1
5310   IFWC(X)<=WC(X+1)THEN5360
5320   H1=WC(X):H2=WN(X)
5330   WC(X)=WC(X+1):WN(X)=WN(X+1)
5340   WC(X+1)=H1:WN(X+1)=H2
5350   SS=1
5360 NEXTX
5390 IFSS=1THEN5305
5400 R1=WN(1):R2=WN(2):R3=WN(3):R4=WN(4):R5=WN(5)
5500 FORX=1TONH
5505   TI=0
5510   IFHD(X,1)>60THENTI=TI+100:HD(X,1)=HD(X,1)-60:GOTO5510
5520   TI=TI+HD(X,1):HD(X,1)=TI
5530 NEXTX
5600 FORX=1TONH
5610   IFNH>9THENOD=HD(X,11)*.9ELSEOD=HD(X,11)
5612   IFNH>14THENOD=OD*.9
5616   IFOD<=3.5THENOD$(X)="3-2"
5620   IFOD>3.5ANDOD<=4.5THENOD$(X)="2-1"
5625   IFOD>4.5ANDOD<=5.5THENOD$(X)="5-2"
5630   IFOD>5.5ANDOD<=6.5THENOD$(X)="3-1"
5635   IFOD>6.5ANDOD<=7.5THENOD$(X)="7-2"
5640   IFOD>7.5ANDOD<=8.5THENOD$(X)="4-1"
5645   IFOD>8.5ANDOD<=9.5THENOD$(X)="9-2"
```

Program continued

✓See List of Advertisers on page 354

Program continues

```
5650   IFOD>9.5ANDOD<=11.5THENOD$(X)="5-1"
5655   IFOD>11.5ANDOD<=13.5THENOD$(X)="6-1"
5660   IFOD>13.5ANDOD<=15.5THENOD$(X)="7-1"
5665   IFOD>15.5ANDOD<=17.5THENOD$(X)="8-1"
5670   IFOD>17.5ANDOD<=19.5THENOD$(X)="9-1"
5675   IFOD>19.5ANDOD<=21.5THENOD$(X)="10-1"
5680   IFOD>21.5ANDOD<=23.5THENOD$(X)="11-1"
5683   IFOD>23.5ANDOD<=25.5THENOD$(X)="12-1"
5685   IFOD>25.5ANDOD<=27.5THENOD$(X)="13-1"
5687   IFOD>27.5ANDOD<=29.5THENOD$(X)="14-1"
5690   IFOD>29.5ANDOD<=31.5THENOD$(X)="15-1"
5692   IFOD>31.5THENOD$(X)="20-1"
5695   IFX=WN(1)THENOD$(X)=OD$(X)+" ***"
5696   IFX=WN(2)THENOD$(X)=OD$(X)+" **"
5697   IFX=WN(3)THENOD$(X)=OD$(X)+" *"
5699 NEXTX
5999 RETURN
6000 CLS
6010 P1$="     COMP.     AVE.             EARN/          "
6020 P2$="H#   TIME      FIN.     %WPS    START     COMP.  POINTS
 ODDS"
6030 MS$="##   ###.#     ##.#    ###.#    ######    ##.#   ##.#
 %     %"
6040 IFNH<13THENPL=NHELSEPL=12
6050 I=1
6100 PRINTP1$:PRINTP2$:PRINT
6105 FORX=ITOPL
6110   PRINT USING MS$;X,HD(X,1),HD(X,2),HD(X,5),HD(X,6),HD(X,11
),HD(X,10),OD$(X)
6120 NEXTX
6125 IFPL>12THEN6140
6130 IFNH>12THENPRINT@960,"PRESS ENTER FOR MORE";:INPUTA$:I=13:P
L=NH:CLS:GOTO6100
6140 PRINT@960,"1-WAGERING    2-GRAPH    3-PRINT    4-NEXT RACE"
;:IFNH>12THENPRINT"    5-RELIST";
6150 O=0:INPUTO
6160 IFO=1GOSUB7000
6170 IFO=2GOSUB8000
6180 IFO=3GOSUB9000
6190 IFO=4THENRETURN
6199 GOTO6000
7000 CLS:PRINT:PRINT"   ** WAGERING **":PRINT
7010 PRINTC$(1):PRINTC$(2):PRINTC$(3)
7020 PRINT:PRINT"  EXACTAS"
7030 PRINTC$(4)
7080 PRINT:PRINT:INPUT"PRESS ENTER";A$
7090 RETURN
8000 CLS:PRINT@836,STRING$(59,CHR$(140));
8010 PRINTSTRING$(31," ");:FORX=1TO10:PRINT1;:NEXTX:PRINT" 2";:P
RINT:PRINT"H# ";:FORX=1TO9:PRINTX;:NEXTX
8020 FORX=0TO9:PRINTX;:NEXTX:PRINT0;
8030 PRINT@0,"78   ";STRING$(59,"-")
8031 PRINT@64,"72  ";STRING$(59,"-")
8032 PRINT@128,"66  ";STRING$(59,"-")
8035 PRINT@192,"60  ";STRING$(59,"-")
8036 PRINT@256,"54  ";STRING$(59,"-")
8037 PRINT@320,"48  ";STRING$(59,"-")
8038 PRINT@384,"42  ";STRING$(59,"-")
8040 PRINT@448,"36  ";STRING$(59,"-")
8050 PRINT@512,"30  ";STRING$(59,"-")
8060 PRINT@576,"24  ";STRING$(59,"-")
8063 PRINT@640,"18  ";STRING$(59,"-")
8065 PRINT@704,"12  ";STRING$(59,"-")
8070 PRINT@768," 6  ";STRING$(59,"-");
8100 FORX=1TONH
8105   IFHD(X,10)<=80THENHP=HD(X,10)ELSEHP=80
8110   FORY=40TO(40-(HP/2))STEP-1
8120     SET((X*6)+2,Y)
8130   NEXTY
8140 NEXTX
8150 A$="":IFINKEY$=""THEN8150
8999 RETURN
9000 CLS:IFPEEK(14312)>127THENPRINT:PRINT"PRINTER NOT READY - PR
ESS ENTER";:INPUTA$:RETURNELSEPRINT@410,"PRINTING..."
9005 PH$="COMPARATIVE ANALYSIS FOR RACE NUMBER"
9010 P1$="      COMP.    AVE.             EARN/          SR/TV"
9020 P2$="H#    TIME     FIN.    %WPS     START           AVE.   COM
P.  POINTS   ODDS"
9030 PL$="##    ###.#    ##.#    ###.#    ######          ###.#   ##.
#   ###.#    %        %"
9100 LPRINT:LPRINTTAB(20)PH$;RN
9110 LPRINT:LPRINTP1$:LPRINTP2$:LPRINT
9200 FORX=1TONH
9210   LPRINT USING PL$;X,HD(X,1),HD(X,2),HD(X,5),HD(X,6),HD(X,9
),HD(X,11),HD(X,10),OD$(X)
9220 NEXTX
9230 LPRINT:LPRINT:LPRINT
9240 LPRINT"  ** WAGERING **":LPRINT:LPRINTC$(1):LPRINTC$(2):LPR
INTC$(3):LPRINT
9250 LPRINT"  EXACTAS":LPRINTC$(4):LPRINT:LPRINT:LPRINT
9290 RETURN
11000 CLS:PRINTTAB(15)"-- CLASS TABLE --":PRINT
11010 PRINT"<3>  A LOT                   ";CHR$(149)
11020 PRINT"<2>  MODERATE                ";CHR$(149);CHR$(149);"-----
->> UP IN CLASS"
11030 PRINT"<1>  SLIGHT                  ";CHR$(149)
11040 PRINT"<0>  EQUAL CLASS"
11050 PRINT"<-1> SLIGHT                  ";CHR$(149)
11060 PRINT"<-2> MODERATE                ";CHR$(149);CHR$(149);"-----
->> DOWN IN CLASS"
11070 PRINT"<-3> A LOT                   ";CHR$(149)
11097 PRINTSTRING$(63,"=")
11099 RETURN
11500 PRINTTAB(15)"-- SAMPLE DISTANCE CHART --":PRINT
11510 PRINTTAB(5)"3F          5F          7F          1M 40Y      1
 3/8M"
11520 PRINTTAB(5)"3.5F        5.5F        7.5F        1M 70Y      1
 1/2M"
11530 PRINTTAB(5)"4F          6F          7.75F       1 1/8M      1
 3/4M"
11540 PRINTTAB(5)"4.5F        6.5F        1M          1 1/4M      2
M"
11550 PRINT:PRINTSTRING$(63,"="):PRINT:RETURN
```

THE PATCH

WE'VE MOVED!

Now we can offer faster and more reliable service -- Including UPS upon request!

DUALCASE*

UPPER/lowercase, full time from power-up; **NO** software; Standard typewriter keyboard operation (shift to UPPERCASE); Control characters can be displayed; 128 Total character set plus full graphics.

SWITCHABLE*

No switches. No driver. Operates from keyboard. Totally disables "PATCH" functions.

KEYBOARD DEBOUNCE*

Extra keyboard debounce, full time from power-up; **NO** software. If dirty keys are a problem, this is for you.

BLOCK CURSOR*

Replaces the underline style cursor directly. Easier to locate on a full screen. No distracting blinking.

SHORT CASSETTE LEADER*

For tape based systems. Does **NOT** change baud rate. Only shortens recorded leader. Saves four seconds of waiting time. Great for data files! No conflict with high baud rate cassette systems.

ELECTRONIC SHIFT-LOCK*

No extra keys or switches. Simply tap either shift key, UPPERCASE lock, normal shift unlocks.

"THE PATCH" is not only a lowercase modification, but a complete system offering many of the enhancements you would like available on your TRS-80. How is this possible? "THE PATCH" is a "smart chip" which monitors the same address lines as the ROM in your TRS-80 Model I Level II. "THE PATCH" recognizes certain addresses, turns off the LII ROM, then substitutes its' own byte of data. This "Patching" is what allows ALL these enhancements with **NO** software overhead, thus **NO** software conflict. In fact "THE PATCH" is the ONLY lowercase modification that is compatible with both Basic and machine language programs AND the ONLY all hardware lowercase modification.

You install "THE PATCH", no need to send your TRS-80 away so there's no hidden installation cost. Detailed instructions guide even the most inexperienced owner to complete installation in about 30 minutes. Works with the two or three chip ROM set. "THE PATCH" unit plugs into the ROM sockets (does not replace existing ROM). Lowercase does require installation of the extra video RAM (supplied with "THE PATCH").

"THE PATCH" is compatible with any word processor, any DOS and also other languages which use ROM sub-routines. "THE PATCH" is never obsolete. As new features become available they can be installed in your "PATCH".

"THE PATCH" is covered with a one year limited warranty on materials and workmanship. (Does void Radio Shack's 90 day warranty). Ask for details . . . we can install for a small fee.

✓62

CALL NOW
(208) 772-9571

CECDAT, INC.
P. O. Box 497
Hayden Lake, ID 83835

ORDER INFORMATION

Name ....
Street ....
City .... State .... ZIP ....
Signature ....
☐ Check, Money Order, Bank Draft
☐ VISA, MASTERCHARGE, Purchase Orders (add 3%)
Card/PO No. ....
Expiration Date .... Today's Date ....
You must check one:
☐ "MEM SIZE" ☐ "MEMORY SIZE"
Price valid through December 31, 1981

"THE PATCH 2" Switchable @ $94.97 ea. ....

OPTIONS: (please check)

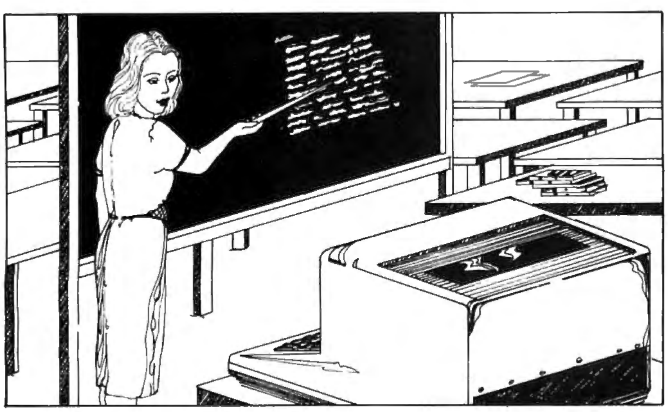
☐ "Electronic Shift-Lock" Add $30.00 ....
☐ "Short Cassette Leader" Add $10.00 ....
☐ "Block Cursor" No Charge
☐ "Debounce" No Charge

Ship. & Hand. @ 2.50 ea. ....
COD ADD 5% . . . .
ID Sales Tax 3% (Id Res) ....
TOTAL ORDER . . . .

✓See List of Advertisers on page 354

Modify ISI's teacher program for network operation.

Teacher Mod



Madeleine Fish
Grant Union High School
1400 Grand Ave.
Sacramento, CA 95838

In this article I present a very simple modification to Instant Software's Teacher program. The changes enable a teacher using a network system to format a lesson using Teacher and then send all the questions at once to a classroom of computers via the network. The program as sold cannot be used on the network. The original format requires each student to individually load questions from tape.

Once you make these changes, save the two programs (original and modified) on tape or disk. The questions must be stored on tape using this program.

The Teacher requires a two-step load with the first program being the logo. Save the second program on a tape or disk. It is this program that we will modify. We are going to create a program to load our questions through the network by making the following changes: Delete 40,50,90-560,580-1630,2160-3420. Add these lines:

```
60 CLS:INPUT"DO YOU WISH TO INPUT
DATA FROM A CASSETTE (ANSWER NO
IF WISH TO SEND ALREADY LOADED
QUESTIONS TO THE NETWORK)";BB
85 IF BB = "NO" THEN 569
570 INPUT"PRESS ENTER TO SEND
YOUR QUESTIONS OUT TO THE NET-
WORK";ZZ: GOTO 1640
1640 PRINT#-1,BT
1642 PRINT#-1,N,BN
1650 FOR Q = 1 TO N
1660 PRINT#-1,A(Q),B(Q):PRINT @ 576,
"QUESTION";Q
1670 NEXT Q
1680 GOTO 60
2140 CLS:GOTO 570
```

The procedure for using our new system is as follows:

- Create a question tape following exactly the directions supplied with the software.
- Load into your control computer the original unmodified program that was saved (program as sold without the initial logo program).
- Send this program via the network to the students' computers.
- The students' display will ask if they want to load from tape. They (or the teacher) should respond with yes and follow directions as if a tape load was to follow.
- Load the shortened modified program into the control computer. Follow directions and load your question tape.
- The computer will then instruct you to depress the Enter key when you are ready send out the questions via the network.

There are some disk-based authorware systems available for the TRS-80. They will also present a problem as far as the network is concerned since we are transmitting information through the cassette port and the speed of our information transfer is necessarily slow.

This is by no means an elegant or complete authorware system, but for an investment of $10, teachers can begin to customize lessons for their students and specific educational program.■

Are my TRS-80* business systems really that good? Michael Tannenbaum thinks so.

Here's what he said about my Model II systems in the August "80 Microcomputing."

On the General Ledger's ability to file and report:

"...the best I have yet seen in a microcomputer system."

On the Accounts Receivable system:

"...clear and simple to follow."

"...the first time I have seen this feature in a microcomputer system."

(referring to its ability to calculate a salesman's commission)

On our policy of customizing our systems to each buyer's needs:

"...Taranto does his customers an enormous service."

And on our systems in general:

"...a comprehensive and well-documented system, an impressive product at a very reasonable price."

Thank you, Michael.

Taranto
& ASSOCIATES, INC. ✓45

The Total System Store™

121 Paul Drive, San Rafael CA 94903.
Outside California, toll free (800) 227-2868. In California, (415) 472-2670.
Authorized dealers throughout America.

✓See List of Advertisers on page 354

Though not probable, it's possible.

Serial Printing With EDTASM-Plus

Howard H. Cohen
1204 Branch Lane
Glen Burnie, MD 21061

After purchasing a Microsoft Editor/Assembler-Plus I wanted a printer for hard copy. After months of waiting I bought a used Selectric terminal and Small System's TRS232. However, to my dismay the Microsoft EDTASM Plus does not support serial printers and there is no company policy to integrate the two products. Therefore, I had to integrate them myself.

Using Small System's RSM Monitor I searched, moved, and changed some locations successfully. These changes assume you have the following software and at least 32K of memory: RSM Small Systems Monitor, Microsoft's EDTASM Plus (version 1.06) and the TRS232 Formatter software. If you do not have the Formatter, get it. It's well worth $14.95.

Run EDTASM Plus and key in the Formatter program from the supplied listing. While you are keying in Formatter, you must change the parameters that you would normally change using the Basic load version (e.g., BAUD, NULLCT, etc. I have the newer Model I TRS-80, therefore I don't use the keyboard debounce program in the Formatter. With a bit more work, integration of the debouncer could be accomplished.) Add the code in Table 1 in front of the start label. Add the following End card at the end of the Formatter:

END PRNTDR.

Then assemble the new Formatter with: A NEWFOR /MO/WE/ and record it on cassette tape.

Next, reload EDTASM Plus but do not Enter. Load in the RSM Monitor at AC00H and execute it. Enter the following commands:

E 4561	Change the 3B to 81
E 4562	Change the 00 to 72
E 4459	Change DD to F7
E 4460	Change the 71 to 73

Now load the tape you made (system *? NEWFOR). Go back to RSM (SYSTEM *? /44032) which was loaded AC00H. Issue the following command to create your composite program tape: P 4380 73F7 4380 EDTASM. (The last step can be repeated to record two copies.)

Next, load your new program. Turn on your serial printer and print away.

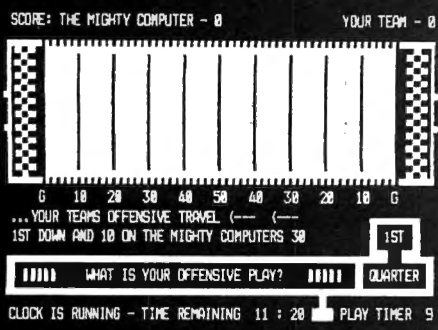
Note the two key addresses (above) in EDTASM Plus, 4561H and 4459H. Address 4561H is where the printer driver is called and 4459H is the address that sets up the start of the EDTASM Plus text buffer. The address at 4561H was changed to call the new printer routine. The address at 4459H was changed to allow the Formatter to become part of EDTASM Plus.

It is important to note that once the EDTASM Plus and the TRS232 Formatter are merged, the Quash function in EDTASM Plus should not be used. If it is, it will cause EDTASM Plus to make an error. The code will call the Formatter that was quashed along with Z-Bug or the Assembler. ■

```
         ORG    7281H   ; ORG THE FORMATTER BEHIND
                          THE EDTASM +
PRNTDR   PUSH   AF      ; SAVE ALL EDTASM + 'S REGISTER
         PUSH   BC
         PUSH   DE
         PUSH   HL
         PUSH   IX
         PUSH   IY
         LD     C,A     ; MOVE THE CHAR TO C-REG
                          WHERE TRS232 IS EXPECTING IT
         CALL   START   ; CALL TRS232
         POP    IY      ; RESTORE ALL REGISTERS
         POP    IX
         POP    HL
         POP    DE
         POP    BC
         POP    AF
         RET            ; RETURN TO EDTASM +
```

Table 1

SUPER MICRO-PRO-CESSOR FOOTBALL



Animated Graphics — Game Time Clock (48K-32K) Available for TRS80* LII M1 or M3 48K DISC $22.95 — 32K DISC $19.95. (48K has more of everything!!!)

16K DISC OR TAPE $12.95

Many more super programs. Please write for free listing and program details.

Attention: Software Authors—We want good programs—Top royalty plan. Dealer inquiries welcomed.

* TRS80 is a Reg. Trademark of Tandy Corp.

MICRO PRO SYSTEMS
Route #2 Box 533
Cumming, Georgia 30130

495

$$ DISCOUNT PRICES $$

SCOTCH DISKETTES

Soft Sector - Single Density - 5¼ inch
32% less abrasive than the industry average.

Soft box of 10	$27.95
Storage box of 10	$28.95
3M Disk Drive Cleaning Kit (5¼ inch)	$22.95
Shipping & insurance for each box/kit	$1.00

TRS-80 SOFTWARE

Designed/tested with TRSDOS & NEWDOS+
Requires 16 K and one disk drive unless noted.
Software prices includes shipping & insurance.

Directory Librarian with diskette order $9.00*
Reads directories of up to 99 diskettes. Maintains a master library of names and free grans. Fast sorts. Supports printer. Req's 48K.

Typing Teacher with diskette order $4.00*
Teaches touch typing. Letter, word, and paragraph exercises. Monitors typing speed.

Math Drills with diskette order $2.00*
Prints personalized practice worksheets for 5 types of math problems. Ages 5-9. Req's MX80.

Game Pack #1 with diskette order $2.00*
Exciting arcade games - Duel, Race & Maze

All four programs with diskette order $14.00*

*without diskette order - add $5.00 for 1 or more

EPSON MX80 PRINTER

80 chars/sec - 9x9 matrix - letter quality
Epson provides 90-day warranty for MX80

- Regular tractor model w/TRS80 cable	$519.00
- Friction/Tractor model w/TRS80 cable	$619.00
- Shipping & insurance	$10.00
- Friction Feed Adaptor (incl. shipping)	$54.95

PAPER & LABELS

Special proposal quality paper - pin fed
Pure white 20 lb. bond with watermark

- Box of 2300 sheets - 9½ x 11 inch	$37.00
- Shipping & insurance for each box	$9.00

Pressure Sensitive Labels

- Box of 5000 labels - 3½ x $^{15}/_{16}$ inch	$19.00
- Shipping & insurance for each box	$4.00

SUPER SPECIAL PACKAGE
Everything in this advertisement!

Diskettes (1 box), cleaning kit, MX80 (w/cable-add $100 for MX80FT), all software, paper & labels (1 box ea). Includes shipping. $649.00

Visa and Mastercard accepted.
Michigan residents add sales tax.

THE COMPUTER ANSWER 471
P.O. Box 573 T
Birmingham, Michigan 48009
(313) 642-4338 (Order Phone • 24 hours/day)

CANADIANS
NOW IN STOCK
ACORN SOFTWARE

PINBALL
BASKETBALL
INVADERS FROM SPACE

Super action with SOUND
& Machine Language SPEED
Each avail for Mod 1 on
CASS $19.95 DISK $24.95

SYSTEM SAVERS
Will copy SYSTEM TAPES
onto DISK or Cassette
Mod 1 DISK SYSTEM $19.95

MEMOREX 5¼" DISKETTES
$5.50 ea 10 for $50.00

MORE TO COME (& Mod 3)
SEND FOR FREE INFORMATION
We accept VISA

CMD MICRO 459
PO BOX 1212
EDMONTON, ALBERTA
T5J 2M4

STOCKCHART - I

MORE THAN JUST A COMPUTER PROGRAM
IT'S AN INVESTMENT TOOL

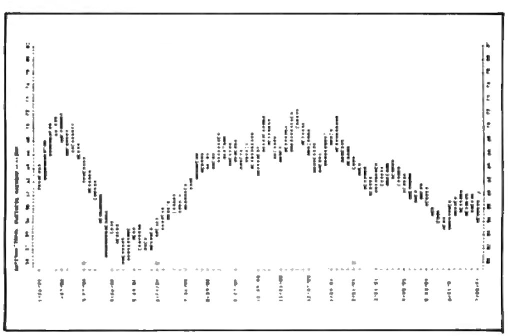
- written for TRS-80* model-I
- requires 16K RAM for cassette, 32K RAM & 1-drive for disk version
- printer option
- disk version $50.00
- cassette version $30.00

to order, send check, money order, or COD to:

Micro-Investment Software
9621 Bowie Way
Stockton, CA. 95209 46

APPLE II* & ATARI-800* version available soon, write for more information

TM of Tandy Corp., Apple Computer, Atari respectively



- generate price-chart from HIGH, LOW, & CLOSE PRICES
- price-trend analysis assist investors in BUY/SELL decision
- auto scaling for price-chart generation
- accept fractions, allow users to enter prices as appear in newspapers
- SPLIT routine to adjust prices for stock split
- EDIT capability for users to change price data
- STATUS mode to display price-trend signals
- PURGE utility to delete unwanted data from disk (disk version only)

GET THE PRICE - TREND PICTURE
WITH
STOCKCHART - I

DEALERS INQUIRIES INVITED

An improved operating system for the Model II.

Snapp's XBasic

Rick Lederman
3375 Airport Highway
Toledo, OH 43609

The Snapp Inc. advertisement promised "the best of Apparat's features in implementing NEWDOS Basic". After one phone call, one plastic cash number and three days, my disk arrived.

Snapp's installation instructions are clear and it takes only a few minutes to modify a disk. To my amazement the software worked as good as or better than the ad claimed.

The first of XBasic's modules are designed to reduce the number of keystrokes that are required during programming or software modification.

There are the six single keystroke commands: The left arrow lists the first line of a program; the right arrow lists the last line of a program; the up arrow lists the previous line; the down arrow lists the next line; the lists the current line; and the comma edits the current line.

There are also 10 abbreviations for commands:

A Auto	L List
C CLS	M Merge
D Delete	N New
E Edit	P Llist
K Kill	S System

Listing and editing your program becomes a piece of cake if you need only type the left arrow (list first line) then type the right arrow (list next line) while holding the Repeat key. This causes the program to list until you release the Repeat key. Then type a comma and you are in the edit mode for the current line. That is much quicker than typing LLIST (enter) then (Break) and edit (line number).

The Model II DOS 1.2 also requires that you enter System"I" immediately before swapping disks whereas DOS 2.0 requires that you enter System"I" after swapping disks. The abbreviation S"I" is appreciated.

The next and, for me, the most valuable routine is XREF, Snapp's cross reference facility. This utility is similar to Apparat's in that you may list references to the screen and printer and list references to line numbers or integer constants and variables. But XREF also places an asterisk in front of any line number not an integer constant. It also places the asterisk before line numbers during the variable listing if the variable is modified in that line.

The enhanced renumbering facility corrects some problems with Radio Shack's Renumber and allows you to specify an upper limit on a block of lines that are to be re-

```
  1   100 150
  3   80
  5   80
 10   100
 12   60 150
 80   *90
100   *80
  A   60($ *110($ 160($
  B   *130/$ 140/$
  J   50
  L   50
 PI   *70 *80/3 110
  X   *100/% 110/%2 120/% *150/% 160/%2 170/%
```

Fig. 1.

```
1 DEFINT J-L:DIMA$(12):PI=3.1415927
2 IFPI>3THEN4ELSEPI=PI[5
3 GOTO2
4 FORX%=1TO10:A$(X%)=STR$(PI*X%):NEXTX%:B$="NOW PRINT THE ARAY":
PRINTB$:FORX%=1TO12:PRINTA$(X%);X%:NEXTX%:END
5 REM        ***** XBTEST *****
10 REM  A PROGRAM TO DEMONSTRATE SNAPP, INC. XBASIC
20 REM  FOR THE TRS-80 MODEL II
30 REM  THIS PROGRAM IS DESIGNED TO DO NOTHING
40 REM
50 DEFINTJ-L
60 DIMA$(12)
70 PI=3.1415927
80 IF PI>3 THEN GOTO 100 ELSE PI=PI[5
90 GOTO 80
100 FOR X%=1 TO 10
110 A$(X%)=STR$(PI*X%)
120 NEXT X%
130 B$="NOW PRINT THE ARRAY"
140 PRINT B$
150 FOR X%=1TO12
160 PRINT A$(X%);X%
170 NEXTX%
180 END
```

Program Listing 1

DOSPLUS

FEATURES:

1) Radio Shack compatibility
2) Error free variable length records
3) Full lower case detection and support
4) Repeating keyboard with NO keybounce EVER
5) Shift [0] typewriter keyboard option
6) Execute only protection feature for BASIC programs
7) Automatic track support for 35 through 80 track drives (mixed)
8) Device I/O handling with FORCE command
9) Supports high speed clock modification (up to 4.0mhz)
10) Supports mixed mode (single & double density) automatically
11) Allows disable-enable of break key
12) Allows user to define step rate per drive and re-configure system disk
13) Allows for efficient use of double-headed drives
14) Built in screen printer (shift [CLEAR]) with [BREAK] key abort
15) Multiple command chaining with "DO"
16) Built in memory test with CLEAR command
17) New printer driver which allows complete forms control and paging
18) Automatic serial printer driver with optional auto linefeed
19) Execute any DOS command from BASIC and return to BASIC
20) Free space map of diskette with optional output to printer
21) Copy with variable length files
22) Complete RS232 control from keyboard with status check
23) Create and pre-allocate files from DOS
24) Display current date and time from DOS
25) More information from Directory with optional printer output
26) Enter DEBUG with shift [BREAK] to allow use of [BREAK] from BASIC
27) New DISKDUMP/CMD sector display/modify program (works with filespecs)
28) New DISKZAP/CMD single/double density disk editor
29) New BACKUP (more reliable, no more pack ID check)
30) New FORMAT (more reliable, no need to bulk erase disk first)
31) New MAP utility (maps out disk, showing where files are located)

New DOSPLUS Z80 Extended Disk BASIC

1) Faster loads and saves
2) BASIC Reference utility (lines, variables, keywords, printer option)
3) BASIC Renumber utility (renumber section of text, block text move)
4) Shorthand features for almost ANY direct command (LOAD, SAVE, etc.)
5) Shorthand features for editing (listing and editing with single key)
6) CMD"M" instantly displays currently set variables
7) Global search and replace in BASIC text
8) Line printer TAB to 255
9) OPEN"E" to end of sequential file (for output)
10) DI (delete and insert text line)
11) DU (duplicate text line)
12) ",R" & ",V" options after LOAD and RUN (files open & save variables)
13) OPEN"D" allowed (Model II compatible) equal to OPEN"R"
14) DOS commands from BASIC
15) Automatic, error-free variable length records
16) Single step execution with TRON (fabulous for debugging)
17) CRUNCH (BASIC program compressor)
18) New TBASIC (tiny BASIC) offers full BASIC commands
19) TBASIC and DOSPLUS together only use 8K of RAM (40K left in 48K TRS-80)

***** 7 MORE UTILITIES *****

1) Single drive copy
2) Restore (dead files)
3) Purge (unwanted files)
4) Clearfile (destroys data by writing zeros to file)
5) Transfer (moves all user files from one disk to another)
6) Spooler (allows printing of text while freeing up the CPU)
7) Crunch (Basic program compressor)

***** ALSO *****

- New I/O package 30% faster
- No BREAK key death from DOS
- No closing killed files and ruining diskettes

DOSPLUS gives you more of what you buy an operating system for. Speed and reliability without sacrificing simplicity and power. If you need extra power without extra wait, then you need DOSPLUS!

Single or double density systems available for Model I. Model III DOSPLUS ready for immediate delivery.

Perhaps the best investment you can make for your TRS-80! Listen to what others have had to say about DOSPLUS.

"Overall, DOSPLUS is the fastest operating system I have seen..."

Pete Carr in 80-US Journal.

"DOSPLUS...the better mousetrap."

Stewart Fason in 80-Microcomputing

"On a scale of 1 to 10, I give DOSPLUS a solid 9."
Reese Fowler in 80-Microcomputing
(Model III DOSPLUS review)

For the BASIC programmer, our features are unmatched. For the average businessman, our speed and simplicity cannot be beat.

So, join the satisfied users who have joined DOSPLUS. Experience excellence! Experience DOSPLUS!

DOSPLUS comes complete with full utilities, PLUS a FREE patch to enable Model I Scripsit/Super Script to run on Model III, UNLIMITED Backups!

Model I DOSPLUS — $99.95
Model III DOSPLUS — $99.95
Model I double density upgrade — $175.00
Master Directory 1.2 (double density) — $29.95

STEP ON THOSE DOS BUGS!! ORDER TODAY!!

VISA

master charge

MICRO SYSTEMS SOFTWARE INC — ✓384
Specializing in the Tandy Line

(305) 983-3390

5846 Funston Street
Hollywood, FL 33023

CALL TOLL FREE FOR FAST SERVICE
(800) 824-7888, OPERATOR 193
FOR VISA/MASTERCHARGE/C.O.D. ORDERS
California dial (800) 852-7777, Operator 193
Alaska and Hawaii dial (800) 824-7919, Operator 193
TOLL FREE LINES WILL ACCEPT ORDERS ONLY!
For Applications and Technical information, call (305) 983-3390 or drop us a card.

Dealers inquiries invited

SINCE YOU COULDN'T WAIT...... WE DIDN'T.....!

IMB

INTRODUCES THE FIRST IN A SERIES OF ACTION PACKED, HIGH RESOLUTION COMPUTER SIMULATIONS.

METEOR STORM © 1981

CONTAINS HIGH-RESOLUTION, SOUND ENHANCED LASER BLASTS; COCKPIT PERSPECTIVE; VARIABLE SKILL LEVEL; AUTOMATIC MULTI-GAME SCORE TALLY AND JOYSTICK OPERATION. (JOYSTICK NOT INCLUDED.)

EXCLUSIVELY DESIGNED FOR

TRS-80* COLOR EXTENDED COMPUTER

Send check or money order $12.95
IMB: ILLUSTRATED MEMORY BANKS
P.O. BOX 289
WILLIAMSTOWN, MA 01267 ✓467

* A TRADEMARK OF TANDY CORPORATION

HAVE YOU *GOT YOURS?*

DISK & OTHER MYSTERIES: how to recover disk files, the layout of information on disks, memory maps, problem solutions, and more -- the fixit book for disk users! $22.50, 130+ pages.

MICROSOFT BASIC DECODED: a detailed, commented guide to the Level II ROM. $29.95, 300+ pages.

CUSTOM TRS-80: brand new & full of super info on customizing your 80. Included subjects are high res graphics, upper and lower case, reverse video, high speed clocks, and more. Software section, too! $29.95, 300+ pages.

THE ALTERNATE SOURCE

SPECIAL OFFER

If you clip this ad, **or** mention it in your order, you can subtract **$3.00** off the cost of **each** book that you order! Each order must be accompanied with $2.00 to cover shipping expenses. Mail to: TAS, 1806 Ada Street, Lansing, MI 48910, or phone (517) 487-3358 or 485-0344. (Add $1.50 for COD, 4% for charge cards).

✓138

TRS-80* MODEL III DISK DRIVE KITS

Process Control Technology's Disk Drive Controller offers more performance for less Dollars!

Jumper program nearly any combination of internal and external Disk Drives. (All four drives can be external, or up to two drives can be internal.)

Hardware compatable with 40, 80, or 77 track (8 inch) drives!

Hardware compatable with double sided drives!

TRS-80* MODEL III
With Two Internal Disk Drives, 32K RAM $1899
TRS-80* MODEL III, MODEL III BASIC, 16K RAM $899

INTERNAL DRIVE KITS

DISK DRIVE 1 .... $679
DISK DRIVE 2 .... $279

EXTERNAL DRIVES

DISK DRIVE 3 (Includes Cable) $359
DISK DRIVE 4 .... $329

Fully Warranted for 90 Days.
*TRS-80 is a trademark of Radio Shack.

PROCESS CONTROL TECHNOLOGY ✓408
P.O. BOX 8825
STOCKTON, CA 95208
(209) 952-6576

Mayday!™

THE UNINTERRUPTIBLE POWER SUPPLY

. . . that keeps your computer on when the power fails . . .

Mayday protects the computer and disk system from power surges, sudden drops in voltage, or a complete disruption of power.

Most problems operators or programmers have are caused by voltage or current interruptions of the computer.

There are 12 different Mayday UPS power supplies, one of which is specifically designed to handle your computer system from 50 watts up to over 600 watts.

Protect your time and investment. One power failure could well cost you the price of the Mayday. Prices start at $240 for the 150 watt system. (Line surge voltage regulator protection is available at additional cost).

For more information, please call or write.

from••• *Sun Research, Inc.*
Box 210 New Durham, NH 03855
(603) 859-7110 TWX 510-297-4444

✓151

```
A $ (0) " " (1) " 3.14159" (2) " 6.28319" (3) " 9.42478" (4) " 12.5664"
   (5) " 15.708" (6) " 18.8496" (7) " 21.9912" (8) " 2 .1327" (9)
   " 28.2743" (10) " 31.4159" (11) " " (12) " " B $ "NOW PRINT THE ARRAY"
PI ! 3.14159
X % 13
```

Fig. 2

numbered. It also allows you to scan the text for renumbering errors such as undefined line numbers or other errors in the text, after which the text numbering remains unchanged.

XRENUM also allows a block of text to be duplicated in another area of the program without destroying the original text or its renumbering. When renumbering a very large program XRENUM is as much as 30 times faster than Radio Shack's Renumber.

Beyond NEWDOS utilities for the Model I, Snapp offers the Model II user XDUMP, a dynamic variable-print facility. This facility prints the current value of any or all variables on the screen or printer.

Fig. 2 demonstrates the results of invoking XDump after the program is run. By entering the command Z, the computer prints the current value of each variable. In the example, PI is in single precision form and equals 3.14159 while X is an integer and equals 13.

XBasic also includes the string/key word cross reference facility called XFind. This facility displays the line numbers where key words appear. This is indispensable for converting from one Basic to another.

'	110
-	50
=	70 80 100 110 130 150
>	80
DEFINT	50
DIM	60
ELSE	80
END	180
FOR	100 150
GOTO	80 90
IF	80
NEXT	120 170
PRINT	140 160
REM	5 10 20 30 40
STR$	110
THEN	80
TO	100 150
↑	80

Fig. 3.

XFind also displays all line numbers that reference any string or portion of a string which you select. This is very handy for reducing space required for quoted strings that are used more than once.

Fig. 3 demonstrates the results of entering F, which instructs the computer to display the line number locations for all key words in the program. The key word Print appears in lines 140 and 160. The string search function is invoked by entering F," " which requests all string references beginning with a null and continued through zzzzz be output to the printer. The example program contains only one fixed string reference which is in line 130.

Finally, XBasic offers XCompress which reduces your program size to an absolute minimum. This is accomplished in several phases, most of which may be optionally selected. These phases are:

- removal of remarks;
- removal of irrelevant blanks;
- removal of irrelevant tab characters;
- removal of extraneous colons;
- removal of the Let key word;
- removal of quotes at the end of a line,
- removal of GOTO following a Then;
- removal of non-significant characters from long variable names;
- removal of non-executable code;
- removal of variable typing characters (!#$%) when given a DEFxxx;
- merging of multiple statements into single lines;
- renumbering the program on a one-by-one basis.

The XCompress facility is invoked by entering H followed by any special parameters you may require. I usually request that the program not be renumbered, which allows me to keep track of differences between program versions more easily.

DOSFIX

The previously discussed utilities are well worth the price, but Snapp Inc. chooses to add at no charge their DOSFIX, a set of repairs to the Model II disk operating system. These include both official Fort Worth and Snapp originals.

The first 16 remove unnecessary files from the operating system, some which don't work anyway, plus junk like SYSL/SYS which only prints the logo on the screen during the power-up sequence. Another patch moves the Break key to CTRL 6. This patch has appeared in *80 Microcomputing* before and has often saved me from entering CONT (enter) which lets the program continue from where it stopped after accidentally hitting the Break key while trying to use the backspace key.

The last DOS fix has stopped my cussing upon power up or reset when the Forms command initializes the printer and the computer issues two line feeds, destroying my top-of-form alignment.

Thanks to Snapp XBasic and their DOSFIX, my Model II no longer seems to fight my every move. ■

```
1 DEFINT J-L:DIMA$(12):PI=3.1415927
2 IFPI>3THEN4ELSEPI=PI[5
3 GOTO2
4 FORX%=1TO10:A$(X%)=STR$(PI*X%):NEXTX%:B$="NOW PRINT THE ARAY":
PRINTB$:FORX%=1TO12:PRINTA$(X%);X%:NEXTX%:END
```

Program Listing 2

✓See List of Advertisers on page 354

Using Model I Scripsit on the Model III.

Patching Across

Richard Koch
2740 Washington St.
Eugene, OR 97405

Recently Radio Shack announced that the Model III version of Scripsit can be copied a limited number of times. Owners can duplicate the program twice, after which backup skips over the Scripsit program; the duplicated programs cannot be copied at all.

If Scripsit cannot be duplicated, things I do with the Model I are impossible on the Model III. I keep a copy of Scripsit on each disk to document computing tasks as soon as they are completed. Apparently that is not possible on the Model III. Moreover, I use a serial printer which prints at 1200 baud, provided it can shake hands with the output program. I have modified Scripsit to provide this handshake, but a program that cannot be duplicated cannot easily be modified.

I predict that Scripsit's limited backup ability will be dropped when Radio Shack decides to emphasize the word processing ability of the Model III. But in the meantime, many of us have a problem. Luckily, it turns out that the old Model I Scripsit package can be easily modified to work on the Model III.

Making the Modification

This modification requires the disk version of Scripsit. Recall that the disk comes with two programs, Scripsit/UC and Scripsit/LC. Naturally we will use Scripsit/LC, because Model III displays lowercase.

"If Scripsit cannot be duplicated, things I do with the Model I are impossible on the Model III."

Begin by placing Scripsit on a Model III disk using Convert. You might like to try this unmodified Scripsit; in large measure, it still works. The right shift key fails, and nothing can be output to the printer. The cassette recorder doesn't work either.

Scripsit starts at 5200H, overlapping the end of Debug. First lift it higher in memory so Debug can inspect the code. The trick runs like this:

Use Debug to insert the following hex code starting at address 9AB0. F3 21 00 72 11 00 52 01 B0 28 ED B0 FB C3 3F 52 F3 21 AF 7A 11 AF 9A 01 B0 28 ED B8 FB C3 2D 40. Quit Debug and enter the command Dump Move/CMD (Start = 9AB0, End = 9AD0, TRA = 9AC0). Then run the unmodified Scripsit/LC. As soon as it begins, enter the command End to return to DOS. Next enter the command Move. The DOS prompt will return. Enter Dump SCR/CMD (Start = 7200, End = 9AC0, TRA = 9AB0). If everything works, you now have a program on disk called SCR. When this program runs, it provides the unmodified Scripsit. But SCR can also be loaded for inspection.

The modifications we must make now are rather minor. Our first problem is that Scripsit doesn't print anything. In the Model I, a character is printed by placing it in the accumulator and then performing LD (37E8H),A. For the Model III, the analogous command is Out (0F8H),A.

To make Scripsit print, Load SCR/CMD and then turn on Debug. Use the M command to change the code 32 E8 37 to D3 F8 00; this change must be made five times, at locations 7244, 865E, 8722, 9A97 and 9A9E.

Our second problem is that the right shift key does not work. The computer reads the status of the shift key by examining memory location 3880. In the Model I, this location contains a zero when no shift key is pressed and a one otherwise.

✓See List of Advertisers on page 354

But in the Model III, the location contains either zero, one, two or three depending on the combination of left and right keys pressed. The code which reads the status of the shift key is LD A,(3880H). We will change this code to Call 78A2H.

Use Debug to change the code 3A 80 38 to CD A2 78. This change must be made twice, at locations 80CD and 80F9.

Of course we need code to process this call. The new routine will read LD A,(3880H); OR A; RET Z; LD A,1; RET. Room will be made for this code by shortening the message "RS232 Interface Not Ready" to "RS232 Not Ready."

Use Debug to change the code starting at 9891 from 1A 52 53 2D 32 33 32 20 49 4E 54 45 52 46 41 43 45 20 4E 4F 54 20 52 45 41 to 10 52 53 2D 32 33 32 20 4E 4F 54 20 52 45 41 44 59 3A 80 38 B7 C8 3E 01 C9.

The cassette recorder must be repaired. We will use ROM routines; five are involved.

Use Debug to change CD F0 63 to C3 87 02 starting at 83CB, to change CD F0 63 to C3 96 02 starting at 83DA, to change D9 F5 0E to C3 64 02 starting at 83AF, to change D9 0E 08 to C3 35 02 starting at 8381, and to change E5 21 00 to C3 F8 01 starting at 83F9.

Finally, the serial printer routine must be changed. If you have an ordinary parallel printer, you might want to skip this one.

> *"The RS232 serial interface... converts data...into a stream of pulses that can be sent to a printer."*

Fixing the Serial Printer

The RS232 serial interface is a device that converts data used by the computer into a stream of pulses that can be sent over a wire to a printer. There is no unique way to do the conversion; Radio Shack's board can do it several different ways. Before data can be output, therefore, the board must be told how it will convert data into pulses. In technical language, the board must be initialized.

In the Model I, this initialization is done in a strange way. The RS232 board contains eight very small switches, called DIP switches. The particular method used in serial output is programmed by setting these switches. But, mysteriously, the switches are not connected to the RS232 board in any serious way. Instead, the computer must tell the board which method to use. Whenever Scripsit is run, the program asks the RS232 board how the switches have been set. It then initializes the board as the switches command.

The Model III does not have DIP switches. We must tell Scripsit from the beginning how the board should be initialized. To do that, we replace the code starting at 661A by: LD A, (Baud Rate); *Out* (0E9H),A; LD A, (Parity and Stop Bits); Out (0EAH),A; JR 15H.

Using Debug, find the following code at location 861A: DB E9 E6 F8 F6 D3 EA DB E9. Replace this code with 3E ** D3 E9 3E ## D3 EA 18 15. The values chosen for ** and ## will depend on your printer. The value ** must be computed using the information "Transmit/Receive Baud Rate Code" on pages 8/5 and 8/6 of the *Model III Operation and Basic Language Reference Manual*. The value ## must be computed using the information "Parity/Word Length/Stop-Bit Code" on pages 8/6 and 8/7 of the same manual.

Finishing the Task

After the above changes have been made, turn off Debug and issue the command Dump Scripsit/CMD (Start = 7200, End = 9AC0, TRA = 9AB0). Test the resulting program. It should work exactly as it did on the Model I.■

★★★

4MHZ, DOUBLE DENSITY, COLOR & B/W GRAPHICS . . THE LNW80 COMPUTER

COMPARE THE FEATURES AND PERFORMANCE

FEATURES	LNW80	PMC-80**	TRS-80* MODEL III
PROCESSOR	4.0 MHZ	1,8 MHZ	2.0 MHZ
LEVEL II BASIC INTERP.	YES	YES	LEVEL III BASIC
TRS80 MODEL 1 LEVEL II COMPATIBLE	YES	YES	NO
48K BYTES RAM	YES	YES	YES
CASSETTE BAUD RATE	500/1000	500	500/1500
FLOPPY DISK CONTROLLER	SINGLE/ DOUBLE	SINGLE	SINGLE/ DOUBLE
SERIAL RS232 PORT	YES	YES	YES
PRINTER PORT	YES	YES	YES
REAL TIME CLOCK	YES	YES	YES
24 X 80 CHARACTERS	YES	NO	NO
VIDEO MONITOR	YES	YES	YES
UPPER AND LOWER CASE	YES	OPTIONAL	YES
REVERSE VIDEO	YES	NO	NO
KEYBOARD	63 KEY	53 KEY	53 KEY
NUMERIC KEY PAD	YES	NO	YES
B/W GRAPHICS, 128 X 48	YES	YES	YES
HI-RESOLUTION B/W GRAPHICS, 480 X 192	YES	NO	NO
HI-RESOLUTION COLOR GRAPHICS (NTSC), 128 X 192 IN 8 COLORS	YES	NO	NO
HI-RESOLUTION COLOR GRAPHICS (RGB), 384 X 192 IN 8 COLORS	OPTIONAL	NO	NO
WARRANTY	6 MONTHS	90 DAYS	90 DAYS
TOTAL SYSTEM PRICE	$1,914.00	$1,840.00	$2,187.00
LESS MONITOR AND DISK DRIVE	$1,450.00	$1,375.00	---

When you've compared the features of an LNW80 Computer, you'll quickly understand why the LNW80 is the ultimate TRS80 software compatible system. LNW RESEARCH offers the most complete microcomputer system at an outstanding low price.
We back up our product with an unconventional 6 month warranty and a 10 days full refund policy, less shipping charges.

LNW80 Computer $1,450.00
LNW80 Computer w/B&W Monitor & one 5" Drive $1,914.00
All orders must be prepaid, CA residents please include 6% sales tax.
Contact us for shipping charges

* TRS80 Product of Tandy Corporation.
** PMC Product of Personal Microcomputer, Inc.

LNW80

- BARE PRINTED CIRCUIT BOARD & MANUAL $89.95

The LNW80 - A high-speed color computer totally compatible with the TRS-80*. The LNW80 gives you the edge in satisfying your computation needs in business, scientific and personal computation. With performance of 4 MHz, Z80A CPU, you'll achieve performance of over twice the processing speed of a TRS-80*. This means you'll get the performance that is comparable to the most expensive microcomputer with the compatibility to the world's most popular computer (TRS-80*) resulting in the widest software base.

FEATURES:
- TRS-80 Model 1 Level II Software Compatible
- High Resolution Graphics
  - RGB Output - 384 x 192 in 8 Colors
  - NTSC Video or RF MOD - 128 x 192 in 8 Colors
  - Black and White - 480 x 192
- 4 MHz CPU
- 500/1000 Baud Cassette
- Upper and Lower Case
- 16K Bytes RAM, 12K Bytes ROM
- Solder Masked and Silkscreened

LNW SYSTEM EXPANSION

- BARE PRINTED CIRCUIT BOARD AND MANUAL $69.95
WITH GOLD CONNECTORS $84.95

The System Expansion will allow you to expand your LNW80, TRS-80*, or PMC-80** to a complete computer system that is still totally software compatible with the TRS-80* Model 1 Level II.

FEATURES:
- 32K Bytes Memory
- 5" Floppy Controller
- Serial RS232 20ma I/O
- Parallel Printer
- Real Time Clock
- Screen Printer Bus
- On Board Power Supply
- Solder Masked and Silkscreened

LNDoubler & DOS PLUS 3.3D

- Assembled and Tested W/DOS PLUS 3.3D. $175.00

Double-density disk storage for the LNW Research's "System Expansion" or the Tandy's "Expansion Interface". The LNDoubler™ is totally software compatible with any double density software generated for the Percom's Doubler***. The LNDoubler™ provides the following outstanding features.

- Store up to 350K bytes on a single 5" disk
- Single and double density data separation
- Precision write precompensation circuit
- Software switch between single and double density
- Easy plug in installation requiring no etch cuts, jumpers or soldering
- 35, 40, 77, 80 track 5" disk operation
- 120 day parts and labor Warranty

*** Doubler is a product of Percom Data Company, Inc.

DOS PLUS 3.3D

Micro Systems software's double density disk operating system. This operating system contains all the outstanding features of a well developed DOS, with ease in useability.

KEYBOARD

LNW80 KEYBOARD KIT $84.95

The Keyboard Kit contains a 63 key plus a 10 key, P.C. board, and remaining components.

CASE

LNW80 CASE $84.95

The streamline design of this metal case will house the LNW80, LWN System Expansion, LNW80 Keyboard, power supply and fan, LNDoubler™, or LNW Data Separator. This kit includes all the hardware to mount all of the above. Add $12.00 for shipping

PARTS AVAILABLE FROM LNW RESEARCH
- 4116 - 200ns RAM
  - 6 chip set $26.00
  - 8 chip set $33.50
  - 16 chip set $64.00
  - 24 chip set $94.00
  - 32 chip set $124.00
- LNW80 "Start up parts set" LNW80-1 . . . . $82.00
- LNW80 "Video parts set" LNW80-2 . . . . $31.00
- LNW80 Transformer LNW80-3 . . . . $18.00
- LNW80 Keyboard cable LNW80-4 . . . . $16.00
- 40 Pin computer to expansion cable $15.00
- System Expansion Transformer $19.00
- Floppy Controller (FD1771) and UART (TR1602) . . . $30.00

LNW RESEARCH
C O R P O R A T I O N

2620 WALNUT
TUSTIN CA. 92680 ✓53

ORDERS & INFO. NO. 714-544-5744
SERVICE NO. 714-641-8850

VISA MASTER CHARGE ACCEPTED

UNLESS NOTED ADD $3 FOR SHIPPING

Stand up and be computed!

The Secret Ballot

David D. Busch
515 E Highland Avenue
Ravenna, OH 44266

Secret Ballot allows any reasonably small group to hold an election using the TRS-80 Model I computer as an impartial, incorruptible voting machine. Users' groups, computer clubs, social groups—any organization that elects officers can use the computer to conduct a secret ballot.

Only voters who have registered ahead of time can vote; the program asks for the name of the person voting, and then strikes that name off its list. It is possible, of course, to vote in the name of someone else—but if that person later tries to vote, he or she will be barred, possibly invalidating the election.

Secret Ballot does not keep track of who voted for whom; as long as other members of the group cannot watch the screen, balloting is 100 percent secret. Write-in votes are not allowed.

As written, Secret Ballot accommodates 100 group members; this can be enlarged to the limits of memory by making a single change in the DIM statement on line 20. Eligible voters may be input ahead of time, and stored in a disk file named Voters. Up to 20 candidates for each of 20 different offices may be input and stored in two additional disk files, Jobs, which stores the names of the offices, and Candid, which keeps track of the candidates.

Input Voters' Names

This module is contained in lines 220–460. If an existing voter file is being updated, it is first loaded from disk in a routine at lines 480–540. The current number of voters in the file, NV, is input first, and then a For. . . Next loop of one to NV retrieves the names and loads them into a string array, VOTERS$(n).

Line 360 adds new names. As each name is input, NV is incremented by one, so the latest name is deposited in the array one element beyond the most recent entry. NV begins at zero if a new file is being created. Caution: If you create a new Voters disk file, it erases any existing file by that name on that disk. Be sure you want to initialize a new file before beginning. The user inputs X instead of a name when finished adding voters to the file.

There is no way to delete a voter's name from the file because the voter list is used only as a check to see that someone is voting just once. A ballot-box stuffer could conceivably vote in the name of a departed member, if he or she understood the program. If your group has a number of despicable members who would attempt to cheat, you could probably write an enhancement to Secret Ballot to delete names, or rig a trap within the program.

When all the voters have been entered, the file is saved to disk (lines 400–460), and control returns to the menu.

Input Candidates Names

Unlike the Voters disk file, the candidates/offices files can only be created from scratch, not updated. Usually, a slate of candidates will be finalized just before an election, and changed for the next one. The eligible voters will essentially remain the same, requiring only minor updates.

To put forth a new slate, the election officer first has to enter a list of the names of the positions to be filled (lines 710–780). The program will handle up to 20 jobs. After the offices have been saved to disk (lines 800–850), the candidates may be entered. The title of the office is listed, and the candidates entered in a For . . . Next loop contained at lines 870–980. The candidate data is stored in a two-dimensional array, CA$(row, col.). Each row accounts for a particular candidate, while the columns of the array point to the specific job. That is, if president were the first office entered into JOB$(n), then the first three candidates would be assigned CA$(1,1), CA$(2,1),CA$(3,1), etc. The first vice presidential candidate would be CA$(1,2), and so on. This string array is saved to disk in lines 1000–1070.

Inspect List of Candidates and Voters

You may wish to check the lists before balloting begins. This option loads all the data files, and lists them to the screen in lines 1430–1690. Because X is used to exit the data entry process, that character will always be the last in each candidate or voter file. The program looks for the X, and sets CFLAG to tell the program whether or not to print an array element to the screen. You could direct the program to exit the display loop when it encountered the X, but the loop is so short in most cases (five or six offices), that little time is saved. Take note when you exit a For . . . Next loop prematurely, because of the error that might be generated the next time you reuse that loop counter in the same, or similar, For. . . Next loop.

Voting

To conduct the actual polling,

Color computer owners,

32K PLUS DISKS*

$298.00

Yes, that's right - for as little as $298.00 you can add 32K of dynamic RAM, and a disk interface, to your TRS-80 Color Computer! If you just want the extra memory it's only $199.00, and you can add the disk interface later for $99.00.

Just plug the *Color Computer Interface (CCI)*, from Exatron, into your expansion socket and "Hey Presto!" - an extra 32K of memory. No modifications are needed to your computer, so you don't void your Radio Shack warranty, and Exatron give both a 30 day money-back guarantee and full 1 year repair warranty on their interface.

The *CCI* also contains a 2K machine-language monitor, with which you can examine (and change) memory, set break-points, set memory to a constant and block-move memory.

So what about the *CCI Disk Card*? Well as we said it's only an extra $99.00, but you'll probably want Exatron's *CCDOS* which is only $29.95 – unless you want to write your own operating system. The *CCI Disk Card* uses normal TRS-80 Model I type disk drives, and *CCDOS* will even load Model I TRSDOS disks into your color computer – so you can adapt existing TRS-80 BASIC programs.

As a further plus, with the optional *ROM Backup* adaptor, you can dump game cartridges to cassette or disk. Once the ROM cartridge is on cassette, or disk, you can reload, examine and modify the software. The *ROM Backup* adaptor is only $19.95.

For more information, or to place an order, phone Exatron on their Hot Line *800-538 8559* (inside California 408-737 7111), or clip the coupon.

excellence in electronics

exatron

DEALER ENQUIRIES INVITED

Exatron, ✓500
181 Commercial Street,
Sunnyvale, CA 94086

☐ Please send a 32K Color Computer Interface for $199.00

☐ Please send a CCI Disk Card for $99.00

☐ Please include CCDOS and manual for $29.95

☐ Also include a ROM Backup adaptor for $19.95

Please add $5.00 for shipping to all orders, and 6 percent sales tax in California.

Name ....

Address ....

City ....

State .... Zip ....

Charge my:

☐ MasterCard Interbank Code ....

☐ Visa Expiration Date ....

Card ....

☐ Check enclosed for ....

☐ Ship COD ($2.00 extra)

Signature ....

*32K RAM plus Disk Interface   TRS-80 is a trademark of Tandy the computer should be set up in a location where curious onlookers cannot see who a member is voting for. The first step is to enter the member's name as it appears on the membership roster. This entry must match exactly the entry in VOTER$(n). Using the full legal name (with middle initial or name) when compiling Voter$(n) provides an extra measure of security. Anyone trying to vote for someone else must know the full name of the person they are trying to impersonate.

```
1 ' ********************************************************
    *                                                      *
2 ' *                    SECRET BALLOT                     *
    *                                                      *
    *                    DAVID D. BUSCH                    *
3 ' *                    515 E. HIGHLAND AVE.              *
    *                    RAVENNA, OHIO 44266               *
    *                                                      *
4 ' ********************************************************

10 CLEAR 2000
20 DIM VOTER$(100),CA$(20,20),BAL(20,20),JOB$(20)
30 ' ****** MENU ******

40 CLS:PRINT
50 PRINT " ::::::   SECRET BALLOT ::::::"
60 PRINT
70 PRINT "DO YOU WANT TO :"
80 PRINT"        1. INPUT VOTERS' NAMES"
90 PRINT"        2. INPUT CANDIDATES' NAMES"
100 PRINT"        3. VOTE"
110 PRINT"        4. INSPECT LIST OF CANDIDATES AND VOTERS"
120 PRINT"        5. SEE RESULTS OF ELECTION"
130 PRINT
140 PRINT "   ENTER CHOICE : ";
150 A$=INKEY$:IF A$="" GOTO 150
160 A=VAL(A$)
170 IF A<1 OR A>6 GOTO 150
180 PRINT A
190 ON A GOSUB 220,710,1090,1430,1780
200 GOTO 40
210 ' ****** ENTER NAMES OF VOTERS ******

220 CLS:PRINT:PRINT
230 PRINT "ARE YOU :"
240 PRINT "              1.) CREATING A NEW VOTER FILE"
250 PRINT "              2.) UPDATING EXISTING VOTER FILE"
260 PRINT
270 INPUT "     ENTER CHOICE:";B$
280 B=VAL(B$)
290 IF B<1 OR B>2 GOTO 220
300 ON B GOTO 320,310
310 GOSUB 480
320 CLS:PRINT:PRINT
330 NV=NV+1
340 PRINT "ENTER VOTER'S NAME";
350 PRINT " ENTER ' X ' WHEN FINISHED "
360 INPUT VOTERS$(NV)
370 IF VOTER$(NV)="X" THEN NV=NV-1: GOTO 400
380 GOTO 320
390 '****** SAVE VOTER FILE TO DISK ******

400 OPEN "O",1,"VOTERS"
410 PRINT #1,NV
420 :  FOR N2=1 TO NV
430 :    PRINT #1,VOTER$(N2);",";
440 :  NEXT N2
450 CLOSE 1
460 RETURN
470 ' ****** LOAD VOTER FILE FROM DISK ******

480 OPEN "I",1,"VOTERS"
490 INPUT #1,NV
500 :  FOR N2=1 TO NV
510 :    INPUT #1,VOTERS$(N2)
520 :  NEXT N2
530 CLOSE 1
540 RETURN
550 '****** LOAD CANDIDATES FROM DISK ******

560 OPEN "I",1,"JOBS"
570 INPUT #1,NJ
580 :  FOR N2=1 TO NJ
590 :    INPUT #1,JOB$(N2)
600 :  NEXT N2
610 CLOSE 1
620 OPEN "I",1,"CANDID"
630 :  FOR ROW=1 TO 20
640 :    FOR COL=1 TO NJ
650 :      INPUT #1,CA$(ROW,COL)
660 :    NEXT COL
670 :  NEXT ROW
680 CLOSE 1
690 RETURN
700 ' ****** ENTER NAMES OF OFFICES ******

710 CLS:PRINT:PRINT
720 INPUT "HOW MANY OFFICES WILL BE ELECTED";AN$
730 NJ=VAL(AN$)
740 :  FOR AN=1 TO NJ
750 :    CLS:PRINT:PRINT
760 :    PRINT"ENTER NAME OF OFFICE #";AN;
770 :    INPUT JOB$(AN)
780 :  NEXT AN
790 ' ****** SAVE NAMES OF OFFICES TO DISK ******

800 OPEN "O",1,"JOBS"
810 PRINT #1,NJ
820 :  FOR N2=1 TO NJ
830 :    PRINT #1,JOB$(N2);",";
840 :  NEXT N2
850 CLOSE 1
860 ' ****** ENTER CANDIDATES' NAMES ******

870 :  FOR N2=1 TO NJ
880 :    CLS:PRINT:PRINT
890 :    C=C+1
900 :    PRINT JOB$(N2)
910 :    PRINT
920 :    PRINT "ENTER CANDIDATES' NAMES"
930 :    PRINT "ENTER ' X ' WHEN FINISHED"
940 :    INPUT CA$(C,N2)
950 :    IF CA$(C,N2)="X" GOTO 970
960 :    GOTO 880
970 :    C=0
980 :  NEXT N2
990 '****** SAVE CANDIDATE FILE TO DISK ******

1000 OPEN "O",1,"CANDID"
1010 :  FOR ROW=1 TO 20
1020 :    FOR COL=1 TO NJ
1030 :      PRINT #1,CA$(ROW,COL);",";
1040 :    NEXT COL
1050 :  NEXT ROW
1060 CLOSE 1
1070 RETURN
1080 ' ****** VOTE ******

1090 GOSUB 480
1100 GOSUB 560
1110 '****** QUALIFY VOTER ******

1120 CLS:PRINT:PRINT
1130 INPUT "PLEASE ENTER YOUR NAME :";VT$
1140 IF VT$="END" RETURN
1150 VFLAG=0
1160 :  FOR N6=1 TO NV
1170 :    IF VOTER$(N6)=VT$ THEN VFLAG=1:VOTER$(N6)=""
1180 :  NEXT N6
1190 IF VFLAG=1 GOTO 1270
1200 PRINT " SORRY, ";VT$;" BUT YOUR NAME DOES NOT"
1210 PRINT " APPEAR IN OUR VOTER LIST.  EITHER YOU HAVE"
1220 PRINT " ALREADY VOTED, ARE INELIGIBLE, OR SPELLED YOUR"
1230 PRINT " NAME DIFFERENTLY THAN IT APPEARS IN OUR LIST "
1240 PRINT
1250 :  FOR N7=1 TO 5000:NEXT N7 : GOTO 1120
1260 ' ***** CONDUCT POLLING ******

1270 :  FOR NW=1 TO NJ
1280 :    CLS:PRINT
1290 :    PRINT JOB$(NW)
1300 :    PRINT
1310 :      FOR N8=1 TO 20
1320 :        IF CA$(N8,NW)="X" THEN CFLAG=1
1330 :        IF CFLAG<>1 PRINT N8;".) ";CA$(N8,NW)
1340 :      NEXT N8
1350 :    PRINT
1360 :    INPUT "PLEASE VOTE:";VOTE$
1370 :    VOTE=VAL(VOTE$)
1380 :    BAL(VOTE,NW)=BAL(VOTE,NW)+1
1390 :    CFLAG=0
1400 :  NEXT NW
1410 GOTO 1120
1420 '****** INSPECT LIST ******

1430 GOSUB 480
1440 GOSUB 560
1450 CLS:PRINT
1460 :  FOR N2=1 TO NJ
1470 :    PRINT JOB$(N2)
1480 :    PRINT
1490 :    PRINT "CANDIDATES:  "
1500 :    PRINT
1510 :      FOR N4=1 TO 20
1520 :        IF CA$(N4,N2)="X" THEN CFLAG=1
1530 :        IF CFLAG<>1 THEN PRINT CA$(N4,N2)
1540 :      NEXT N4
1550 :    GOSUB 1710
1560 :    CFLAG=0
1570 :  NEXT N2
1580 PRINT
1590 CLS:PRINT
1600 PRINT "VOTERS :":PRINT
1610 T=0
1620 :  FOR N5=1 TO NV
1630 :    T=T+1
1640 :    IF T/10=INT(T/10) GOSUB 1710
1650 :    PRINT VOTER$(N5)
1660 :  NEXT N5
1670 PRINT
1680 GOSUB 1710
1690 RETURN
1700 ' ****** 'PAGE' SUBROUTINE ******

1710 PRINT
1720 PRINT "HIT ENTER TO SEE REST OF LIST";
1730 INPUT A$
1740 CLS:PRINT
1750 RETURN
1760 ' ****** PRINT OUT RESULTS ******

1770 CFLAG=0
1780 :  FOR NQ=1 TO NJ
1790 :    CLS:PRINT
1800 :    PRINT JOB$(NQ)
1810 :    PRINT
1820 :    PRINT "CANDIDATE","NO. VOTES"
1830 :      FOR N9=1 TO 20
1840 :        IF CA$(N9,NQ)="X" THEN CFLAG=1
1850 :        IF CFLAG<>1 PRINT CA$(N9,NQ),BAL(N9,NQ)
1860 :      NEXT N9
1870 :    GOSUB 1710
1880 :    CFLAG=0
1890 :  NEXT NQ
1900 GOSUB 1710
1910 RETURN
```

Program Listing

When the name is entered in line 1130, a For...Next loop looks at every name in VOTER$(n), and, if the name is located, a flag, VFLAG, is set to one, which empowers the person to vote. The name is also removed from the array (line 1170), so that person cannot vote twice in that session. The original disk file, of course, remains unchanged, so it can be loaded and reused for repeated ballotings. If no name match is found in VOTER$(n), a scolding message appears on the screen, and control goes back to the name input stage.

Two nested loops at lines 1270–1410 present each office and its candidates, and allow the voter to cast a ballot. Candidates' totals are stored in an array, BAL(row, col.), in which the rows and columns correspond with the positions of the candidates names in CA$(row, col.). Each element of BAL(row, col.) is incremented each time the appropriate candidate receives a vote. Nothing prevents a member from throwing away a vote by entering a number (or character) that does not have a corresponding candidate.

Balloting is ended by entering the word End, instead of a name. At this point, control branches back to the menu, where officials can choose to have the results printed out to the screen.

Print Out Results

This short subroutine at lines 1770–1910 uses two nested For ...Next loops to print out all the elements of CA$(row, col.), and their respective vote totals in BAL(row, col.).

Secret Ballot provides a simple way to conduct a computer club's election. Obviously, a users' group is a bad place to depend on the security of a TRS-80 to protect the integrity of an election. You may want to add an appropriate POKE or two to disable the Break key. Or, you could compile this program using Microsoft's Basic compiler as a way of discouraging tampering.■

✓See List of Advertisers on page 354

Exploit the 737's features with enhanced software.

Enhanced Word Processing

John F. Rogers
600 Seventh St.
Morgan City, LA 70380

Flextext
Apparat, Inc.
Denver, CO
$29.95
Subscript 5.0
Prosoft
No. Hollywood, CA $79.95

Radio Shack introduced its Line Printer IV (Centronics 737-1) as having fonts of word processing quality. However, the July, 1980 newsletter announcement and the advertisement both carefully minimized the fact that the Scripsit word processor could *not* right-justify the proportional font, which is the high quality one, *nor* use any of the other special features, such as underlining, sub and superscripting, elongated characters, or forward and reverse linefeeds.

Happily for TRS-80 users who have the Centronics 737 printer, there are several software packages available that will allow you to take advantage of most or all of that printer's capabilities. Some are enhancements of Scripsit, such as Apparat's Flextext, Terry Fiveash's Scriptmod, or Acorn's Superscript. Others are complete programs, such as Prosoft's Newscript and Subscript 5.0 or SSM's Lazy Writer. Computex's PEN737 enhances Electric Pencil to utilize the 737 fully.

Flextext

Apparat, the Denver people who upgraded TRSDOS so well with NEWDOS, show off their enhancement of Scripsit with a handsome full-page ad totally prepared on the Centronics 737-1 printer. Flextext modifies Scripsit on disk so that printer function commands embedded in text use almost every ability of the printer—proportional font right-justification (mixed with elongated and compressed characters), underlining, superscripts and subscripts—all anywhere within a document.

A special file called Modify/SCP is entered in DOS to incorporate Flextext comments in Scripsit. The modification is stored as a separate file named SCP/LC or SCP/UC without any alteration of the original Scripsit file. (LC and UC stand for lowercase and uppercase.) SCP/xx has all of Scripsit's features. Simply enter SCP/LC or SCP/UC instead of Scripsit/xx.

The embedded-in-text printer commands always begin with the ESCape character (the less-than symbol-<) followed by a primary action code (either an S or an E, for Start or End) followed by a secondary action code that identifies the 737 printer function to be activated (or halted). For example, <SU would begin the underlining function.

One excellent feature actually overcomes a 737 shortcoming—stopping elongated printing at the end of a print line. Flextext continues double-width printing for line after line if desired. Special print commands activate Flextext when it is time to print a document. They are P,FLX for continuous forms and P,FLX,P for single pages.

With Apparat's Flextext, Radio Shack's Scripsit combined with the Centronics 737 printer becomes an outstanding word processor.

Subscript

Subscript 5.0 combines programs named Edit, Script, Prop, and Mininit into a Basic/machine-code package that is line-oriented rather than character-oriented like Scripsit. The user composes in Edit mode, inserting print format commands as he goes along. Since this is done in Basic, one must hit Enter after each 255 characters, or preferably, at the end of each display line (64 characters).

When the document is ready for printing, Script is called up to read the printer function commands and, combined with machine-code Prop, format and print the document using all of the 737's abilities.

Mininit is an upper/lowercase driver that supports Radio Shack's modification. It also detects Basic's string compression mode, which can appear like a CPU hangup, since Basic is reallocating string space; a graphic C is displayed in the upper right corner of the display. Mininit also has a screen-print routine—shift-zero transfers the display contents to the printer. Keyboard debounce is built in.

Even though Subscript is a Basic-centered package, it has some features not found in Scripsit or Electric Pencil. One is a darkness control code, which causes the printing to be overstruck for darker type. When this code is operating, the 737 prints a line, does a carriage return, then prints the same line again on top of the first printing.

Another is a table-of-contents code which builds up a separate file of strings which will be printed as the table of contents pages, automatically adding the number of the page where each string is located. Still another is you to number paragraphs.

A fourth extra feature is the alter code that can print characters that are not on the keyboard, such as square brackets or braces, a vertical line or backward slash, or any other special character that the 737 can print. These characters are conveniently listed.

However, Flextext wins over Prop in right-justification ability. Prop can right-justify single and double-width proportional styles mixed on a line, but not a mixture of proportional and compressed fonts, while Flex-text handles all combinations (excluding 10 characters per inch).

NEWSCRIPT

Prosoft has further upgraded Subscript 5.0. Newscript combines prop and Mininit with new machine language routines into a single command: NS/CMD, which automatically invokes Basic and brings up the menu program. The menu is written in Basic, allowing you to easily change it to suit your particular needs. While Subscript 5.0 can run in 32K TRS-80's, Newscript requires 48K.

Some important changes and additions make Newscript even more convenient and versatile than subscript. It is a full-screen editor rather than the usual line-oriented editor. The cursor can be placed anywhere in the text to make changes or additions. It is no longer necessary to hit Enter after every 255 characters, but only when memory is full. When Enter is hit, Edit processes everything on the screen and then generates a blank line so the user may continue typing.

The alter function has been dropped in favor of direct entry of special characters while in Shift-Clear control mode; single-key entry of square brackets, braces, etc., is much quicker.

The Directory command displays a disk's directory (NEWDOS/80 or Model III). An index is generated by inserting the proper control word near the words or phrases to be indexed. The "Line Manipulation Area" (LIMA) commands, displayed down the left side of the screen, let you insert, delete, duplicate, and mark lines for various purposes.

The Read control word allows creation of form letters with different names and addresses taken from a disk file. Newscript and Subscript also support some of the special functions of the Epson MX-80, IBM Selectric, and Diablo 1620 printers.

The documentation for both packages is absolutely first rate —probably the best I've ever seen! The Newscript manual even has a How To section.

Newscript (and Subscript 5.0) comes on disk from which the user should create a DOS disk containing all the necessary files. With two disk drives, creating a System disk with Newscript on it is simplicity itself: In DOS mode, enter INSTALL:1 and the installation is completed without operator intervention—very thoughtful of Prosoft.

Summary

The main difference between Scripsit/Flextext and Newscript/Subscript is in printing speed. Because Script is in Basic, it reads controls more slowly than Flextext does, and there are noticeable pauses during printing. This is not, however, a major shortcoming; a few seconds here and there really aren't inconvenient. ■

✓See List of Advertisers on page 354

And on the 7th day at this Sunday school, they computed.

The Walls of Jericho

Chuck and Jo McNichols
2052 Granada Drive
Dayton, OH 45431

The TRS-80 can capture student interest and help with teaching chores in a Sunday school, since motivation is a major problem when kids feel they've already done their time in class during the week. This article presents a 16K Level II TRS-80 program that combines an attention-getting welcome sequence with a game that makes memorizing Bible facts and terminology enjoyable.

The open classroom approach to Sunday school, which features learning centers and gives students flexibility in choosing activities, makes it easy to work a computer center into the teaching program. Many Protestant churches now use the *Christian Education: Shared Approaches* (CESA) materials that support teaching with learning centers. We've found it relatively easy to get our computer center started by adapting some CESA workbook ideas, but other educational materials work as well.

The Computer Center

Our fourth and fifth-grade learning center is just a large table that holds the computer and two mounted posters which identify the center and provide instructions for operating the computer. It shows the location of the TRS-80 space bar and says any further instructions will appear on the video display. We've chosen to control programs primarily with space bar input because the location of specific keys is difficult for non-touch typists.

As children arrive in the classroom, they are greeted by a welcoming message that displays their first names randomly on the screen. This is followed by the drawing of a happy face. The welcoming sequence repeats until a shift A is input to begin the game.

Playing the Walls of Jericho

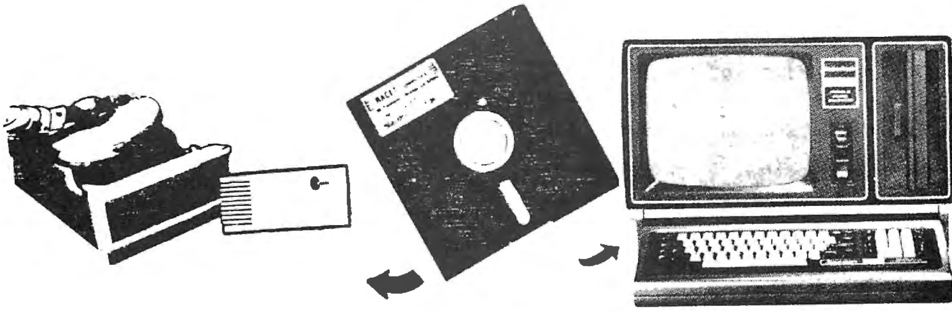
This game was designed for use with a unit on the battle of Jericho, but the questions can be easily altered for other topics. The object is to knock down the wall of Jericho by correctly answering a sequence of multiple choice questions.

Following an introduction describing the game's rules, a wall is drawn. Then, a term taken from the lesson flashes on and off at a randomly selected block in the wall while multiple choice responses are displayed at the bottom of the screen. An arrow cycles among the responses, pausing for a few seconds at each one. The player presses the space bar when the phrase believed to match the flashing word is indicated. A correct response causes the wall above the flashing word to crumble; an incorrect response leaves the wall intact. The player gets 20 tries to knock down the entire wall. Each player's score is the number of chances remaining when the wall is gone. At the end of each game a "next player" message is displayed and the program is set up for the next contestant. The best score achieved so far is retained and displayed while successive games are in progress.

Program Instructions

To alter the number of names displayed in the welcoming segment, the value of NX in line 180 should be changed. New nine-character string constants can then be set up representing students' first names. Mixing a blank or two (like NM$(21)) in the list adds to the visual impact of the display. The happy-face routine (lines 510–850) relies on a circle-drawing technique used for many years with vector-graphics computers.

To customize the Walls of Jericho game, it's necessary to change the data statements at lines 1230–1400, the response subroutines that begin at line 3460, and the computed GOSUB on line 2360. The data statement at line 1240 gives the number of terms available for display at random locations in the wall, and is the value of NQ. Responses for each question are displayed as a result of the computed GOSUB on line 2360. The index for this GOSUB is obtained from the array entry Q(I, 1) where I is the index of the term currently being displayed. The GOSUB index values are



FIELD PROVEN!!

10 MEGABYTES and MORE for the TRS-80* Model II plus SHARED ACCESS to HARD DISK DRIVE

Hard/Soft Disk System (HSDS) Software allows access as single drive. You can have that 10 Megabyte continuous file - that 50,000 name maillist or inventory! Or a directory with 1000 entries! All completely compatible with TRSDOS 2.0 BASIC. You can mix floppy and hard disk drives. Includes special utilities including HPURGE, DCS Directory Catalog System, HZAP Hard Disk Superzap, and many special formatting options. Three to eight times faster than floppy! RACET quality

HARD DISK DRIVE & CONTROLLER $5995. Second User $595. HSDS Software $400. (Note: HSDS now also available for CORVUS drives!!)

KFS-80 (1-drive 32K Min - Mod II 64K) **Mod I and III $100, Mod II $175**
Assembly language ISAM facility! Interfaces with BASIC. Very fast access to records in large files. Access/insert times 1-4 seconds. B-Tree directory structure eliminates record search. No file reorganization required. Sector buffering reduces media and drive wear while increasing performance. Can be used for primary and secondary file database applications. Records are kept in sorted order on disk by alpha key. Deferred write option. Comprehensive file maintenance and debugging utilities are included. **Attention applications programmers** - Save time in implementing your custom database application software.

MAILLIST (1-drive 32K Min - Mod II 64K) **Mod I and III $75, Mod II $150**
This ISAM-based maillist minimizes disk access times. Four keys — no separate sorting. Supports 9-digit zip code and 3-digit state code. Up to 30 attributes. Mask and query selection. Record access/update time 1-4 seconds!!

DISCAT (32K 1-drive Min) **Mod I and III $50**
This comprehensive Diskette Cataloguing/Indexing utility allows the user to keep track of thousands of programs in a categorized library. Machine language program works with all TRSDOS and NEWDOS versions. Files include program names and extensions, program length, diskette numbers, front and back, and diskette free space. RS232 drivers and other features.

LPSPOOL (32K 1-drive Min) **Mod I $75**
LPSPOOL — Add multi-tasking to permit concurrent printing while running your application program. The spooler and despooler obtain print jobs from queues maintained by the system as print files are generated. LPSPOOL supports both parallel and serial printers.

BASIC LINK FACILITY 'BLINK' (Mod I Min 32K 1-disk) **Mod I $25; Mod II $50; Mod III $30**
Link from one BASIC program to another saving all variables! The new program can be smaller or larger than the original program in memory. The chained program may either replace the original program, or can be **merged** by statement number. The statement number where the chained program execution is to begin may be specified!

INFINITE BASIC (Mod I & Mod III Tape or Disk) **Mod I $50; Mod III $60**
Extends Level II BASIC with complete MATRIX functions and 50 more string functions. Includes RACET machine language sorts! Sort 1000 elements in 9 seconds!! Select only functions you want to optimize memory usage.

INFINITE BUSINESS (Requires Infinite BASIC) **Mod I & III $30**
Complete printer pagination controls — auto headers, footers, page numbers. Packed decimal arithmetic — 127 digit accuracy +, −, *, /. Binary search of sorted and unsorted arrays. Hash codes.

COMPROC (Mod I & Mod III — Disk only) **Mod I $20; Mod III $30**
Command Processor. Auto your disk to perform any sequence of instructions that you can give from the keyboard. DIR, FREE, pause, wait for user input, BASIC, No. of FILES and MEM SIZE, RUN program, respond to input statements, BREAK, return to DOS, etc. Includes lowercase driver software, debounce and screenprint!

GSF (Mod I & III Tape or Disk - Specify Memory Size) **Mod I $25; Mod II $50; Mod III $30**
Generalized Subroutine Facilities. The STANDARD against which all other sorts are compared! And then compare prices! Machine language — fast and powerful! Multi-key multi-variable and multi-key character string. Zero and move arrays. Mod II includes USR PEEKS and POKES. Includes sample programs.

DSM (Mod I Min 32K 2-drive system. Mod II 64K 1-drive. Mod III Min 32K 1-drive) **Mod I $75; Mod II $150; Mod III $90**
Disk Sort/Merge for RANDOM files. All machine language stand-alone package for sorting speed. Establish sort specification in simple BASIC command File. Execute from DOS. Only operator action to sort is to change diskettes when requested! Handles multiple diskette files! Super fast sort times — improved disk I/O times make this the fastest Disk Sort/Merge available on the TRS.

UTILITY PACKAGE (Mod II 64K) **$150**
Important enhancements to the Mod II. The file recovery capabilities alone will pay for the package in even one application! Fully documented in 124 page manual! XHIT, XGAT, XCOPY and SUPERZAP are used to reconstruct or recover data from bad diskettes! XCOPY provides multi-file copies, 'wild-card' mask select, absolute sector mode and other features. SUPERZAP allows examine/change any sector on diskette including track-0, and absolute disk backup/copy with I/O recovery. DCS builds consolidated directories from multiple diskettes into a single display or listing sorted by disk name or file name plus more. Change Disk ID with DISKID. XCREATE preallocates files and sets 'LOF' to end to speed disk accesses. DEBUG!! adds single step, trace, subroutine calling, program looping, dynamic disassembly and more!!

BASIC CROSS REFERENCE UTILITY (Mod II 64K) **$50**
SEEK and FIND functions for Variables, Line Numbers, Strings, Keywords. 'All' options available for line numbers and variables. Load from BASIC — Call with 'CTRL'R. Output to screen or printer!

DEVELOPMENT PACKAGE (Mod II 64K) **$125**
Includes RACET machine language SUPERZAP, Apparat Disassembler, and Model II interface to the Microsoft 'Editor Assembler Plus' software package including uploading services and patches for Disk I/O. Purchase price includes complete copy of Editor Assembler + and documentation for Mod I. Assemble directly into memory, MACRO facility, save all or portions of source to disk, dynamic debug facility (ZBUG), extended editor commands.

DEALER inquiries invited. System Houses - check for special pricing on hard drives and software.

CIRCLE READER REQUEST FOR FREE 24-PAGE CATALOG
*TRS-80 IS A TRADEMARK OF TANDY CORPORATION

CHECK, VISA, M/C, C.O.D., PURCHASE ORDER
TELEPHONE ORDERS ACCEPTED (714) 997-4950

RACET COMPUTES
1330 N. GLASSELL, SUITE M, ORANGE, CA 92667

loaded by the same data statements that load the terms (lines 1280–1400).

The W(,,) matrix controls the overall graphic display. The screen is divided into a 12 by 6-block matrix, each represented by a string variable and drawn by one of the subroutines beginning at line 3180. Drawing or removing the wall is controlled by calling these subroutines using computed GOSUBs at lines 2090 and 2640.

The main loop for play of the game begins at line 2160 and ends at line 2900. A wall location in the lower nine rows of the 12 by 6 block grid that has not been disintegrated yet is randomly chosen for display of the term (lines 2190–2220). Seventy percent of the time one of the bottom four rows is chosen with the current code. Then a term is randomly selected from those not yet correctly matched with a response. When all terms have been correctly identified, the entire set is again made available. Thus, there can be less than 20 terms in a game with 20 chances for the player.

The possible responses are then displayed in a routine that also flashes the term and moves the response arrow (beginning at line 2320). Routines for reacting to an incorrect or correct response begin at lines 2540 and 2660 respectively. Every time part of the wall is removed, the W(,,) matrix is checked to see if anything is left (lines 2830–2890). If not, we have a winner, and an appropriate message at line 3120 precedes the recycle of the game for the next player. If the wall survives 20 tries by the player to match terms, the code at line 2910 is used to restart the game.

Our Experience With the TRS-80

The biggest problem we've encountered is needing another machine for the class, which numbers over 20 students. Today's fourth and fifth graders seem very comfortable working and playing in a computer environment. We plan to continue our experimentation with this learning medium, and would enjoy sharing experiences with other Sunday school teachers who have found similar use for their microcomputers. ■

Welcoming Segment

NX	Number of names for welcoming message.
NM$(30)	Array holding names as nine-character strings.
LN(10)	Screen locations for display of names.
W	Controls length of time-delay subroutine.
FL	If FL = 1, circle drawing subroutine draws full circle. Otherwise, a semicircle is drawn.
N	Number of increments defining the circle.
R	Radius of the circle.
S	Starting angle for circle.
E	Ending angle for circle.

Walls of Jericho Game

W(12, 6, 2)	Describes the 12 row by 6 column grid that defines the wall.
Q(20, 3)	Question control array. The three values provided for each question represent the index of the computed GOSUB that displays the responses, the number of the correct response (one if A is correct, two for B, three for C), and the number of responses (up to three). Q(I,0) is set to zero when a term has been correctly identified.
Q$(20)	Nine-character terms that represent the questions.
NB	Holds best score across multiple players.
P,R,S	Define screen locations for displaying responses.
S$(6)	Displayed sequentially to disintegrate the wall.
BF$, BR$, etc.	Nine-character graphics strings which represent the various wall components.

Table 1. Major Variables and Arrays

Program Listing

```
100 CLEAR 1000
110 DIM W(12,6,2),Q(20,3),Q$(20),NM$(30),LN(10)
120 '
130 'WELCOME MESSAGE AND HAPPY FACE
140 '
150 CLS
160 PRINT CHR$(23);
170 'NX IS NUMBER OF NAME ENTRIES, UP TO 30
180 NX=24
190 NM$(1)="ANN      ":NM$(2)="LAURIE   ":NM$(3)="DAVID    "
200 NM$(4)="ROBERT   ":NM$(5)="LISA     ":NM$(6)="BARBARA  "
210 NM$(7)="POLK     ":NM$(8)="KATIE    ":NM$(9)="ANNE     "
220 NM$(10)="ELLEN    ":NM$(11)="LAURA    ":NM$(12)="LEXY     "
230 NM$(13)="ASHLEY   ":NM$(14)="JENNIFER ":NM$(15)="JOHN     "
240 NM$(16)="BILLY    ":NM$(17)="ANNA     ":NM$(18)="EVAN     "
250 NM$(19)="KATIE    ":NM$(20)="STEVEN   ":NM$(21)="         "
260 NM$(22)="SCOTT    ":NM$(23)="JASON    ":NM$(24)="MICHAEL  "
270 'SETUP SCREEN POSITIONS FOR RANDOM NAME DISPLAY
280 LN(1)=200:LN(2)=236:LN(3)=340:LN(4)=360:LN(5)=450
290 LN(6)=480:LN(7)=590:LN(8)=620:LN(9)=704:LN(10)=746
300 PRINT@0,"WELCOME TO THE WESTMINSTER";@64,"LEARNING CENTER...
";
310 'INCREASE W IN NEXT LINE TO WAIT LONGER AFTER MESSAGE
320 W=400:GOSUB 860
330 'K LIMIT TELLS HOW MANY TIMES TO REPEAT EACH NAME
340 FOR K=1 TO 3
350  FOR I=1 TO NX
360  'LC IS RANDOMLY SELECTED SCREEN POSITION
370   LC=RND(10)
380   PRINT@LN(LC),NM$(I);
390  'INCREASE W IN NEXT LINE TO HOLD NAME LONGER
400   W=50:GOSUB 860
410  'CHECK FOR EXIT REQUEST
420   GOSUB 670
430  NEXT I
440 NEXT K
450 FOR J=1 TO 10
460  PRINT@LN(J),"         ";
470 NEXT J
480 PRINT@960,"   WE'RE GLAD YOU'RE HERE !!";
490 W=1000:GOSUB 860
500 GOSUB 670
510 'DRAW HAPPY FACE
520 PRINT CHR$(28);
530 FL=1:N=225:R=35:E=2*ATN(1):S=E/N:E=E+E/N
540 GOSUB 700
550 GOSUB 670
560 FL=0:N=181:R=28:S=7*E/N:E=E+2*E/N
570 GOSUB 700
580 GOSUB 670
590 'SETUP NOSE,N$, AND EYES,E$
600 E$=CHR$(141)+CHR$(140)+CHR$(140)+CHR$(140)+CHR$(142)
610 N$=CHR$(136)+CHR$(176)+CHR$(176)+CHR$(152)
620 PRINT@470,E$;:PRINT@485,E$;:PRINT@606,N$;
630 W=2000:GOSUB 860
```

Program continued

WE WILL NOT BE UNDERSOLD

DISK DRIVES

FOR TRS-80* Model I

CCI-100	5¼", 40 Track (102K)	$299

ADD-ON DRIVES FOR ZENITH Z-89

CCI-189	5¼", 40 Track (102K)	$394
Z-87	Dual 5¼" system	$995

External card edge and power supply included. 90 day warranty/one year on power supply.

RAW DRIVES
5¼" TEAC or TANDON **$ CALL** POWER SUPPLIES $ CALL

DISKETTES — Box of 10

5¼"	Maxell	$40	BASF/Verbatim	$26.95
8"	Maxell	$45	BASF/Verbatim	$36.00

PLASTIC FILE BOX-Holds 50 5¼" diskettes		$19.00
PLASTIC LIBRARY CASE	5¼" $3.00	8" $ 4.00
HEAD CLEANING DISKETTE		$25.00
FLOPPY SAVER	$11.95	RINGS $ 6.95

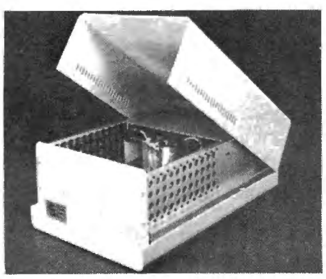
16K RAM KITS
2 for $37 **$19**
200ns for TRS-80,* Apple II, (specify): **Jumpers $2.50**

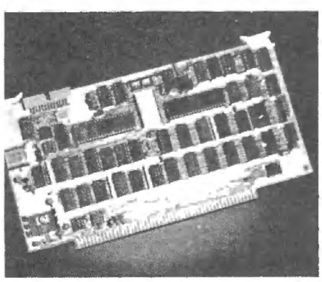
SYSTEM SPECIAL
Apple II Plus 48K w/drive and controller. Epson MX-80 printer and interface. SUP-R Mod RF Modulator: List $2965 You Pay **$2299**

COMPUTERS/TERMINALS

ARCHIVES	64K, 2-Drives, 77 Track	$CALL
ALTOS	ACS8000 Series	$ CALL
ZENITH	48K, all-in-one computer	$2200
ZENITH	Z-19	$ 725
TELEVIDEO	910 $559 920C $ 729 950	$1039
IBM	3101 Display Terminal	$1189
ATARI	400 $ 429 800	$ 795
MATTEL	INTELLIVISION	$ 259
APPLE PERIPHERALS		$ CALL

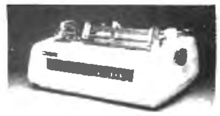
S-100 CALIFORNIA COMPUTER SYSTEMS





MAINFRAME	Model 2200A	$349
Z80 CPU	Model 2810	$259
MOTHER BOARD	Model 2501	$106
16K STATIC RAM, 200ns	Model 2116C	$309
32K STATIC RAM, 200ns	Model 2032C	$619
64K DYNAMIC RAM	Model 2065C	$580
FLOPPY DISC CONTROLLER	Model 2422A	$345
EXTENDER BOARD	Model 2520K	$ 52
2P + 2S I/O	Model 2718A	$309

PRINTERS



NEC SPINWRITER
Letter Quality High Speed Printer

R.O.	$2395
R.O. with tractor feed	$2555
KSR with tractor feed	$2795

NEC SPINWRITER	3500 Series, 5 new models	$CALL
EPSON	MX-70 MX-80 MX-80FT MX100	$CALL
PAPER TIGER		
IDS 445	Graphics & 2K buffer	$ 695
IDS 460	Graphics & 2k buffer	$ 799
IDS 560	Graphics	$ 999
ANADEX	DP-8000 $849 DP-9500/01	$1295
OKIDATA		
Microline 80	Friction & pin feed	$ 415
Microline 80	Friction, and pin & tractor feed	$ 500
Microline 82	Friction & pin feed feed	$ 615
Microline 83	120 cps, uses up to 15" paper	$ 849
CENTRONICS	739, new model with graphics	$ 749
C. ITOH		
Starwriter I	25 cps, parallel interface	$1449
Starwriter I	25 cps, serial interface	$1529
Starwriter II	45 cps, parallel interface	$1829
Starwriter II	45 cps, serial interface	$1949

PRINTER SPECIAL
SEIKOSHA GP-80M List $399 You Pay **$319**

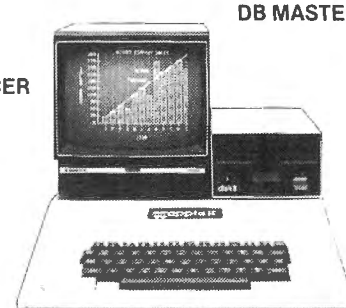
MONITORS

BELL & HOWELL	9" B & W BHD911			$195
LEEDEX	12" B & W	$129	12" Green Screen	$159
	13" Color	$329		
SANYO	9" B & W	$149	12" Green Screen	$238
	12" B & W	$219	13" Color	$406
ZENITH	13" Color	$349	12" Green Screen	$129

TELECOMMUNICATIONS

LIVERMORE STAR MODEM 2-year guarantee	$125
UNIVERSAL DATA SYSTEMS UDS-103	$179
D-CAT HARD WIRED DIRECT MODEM	$189
AUTO-CAT Auto Answer, Direct Connect Modem	$249
APPLE-CAT II	$349
D.C. HAYES MICRO-MODEM	$295
D.C. HAYES SMART-MODEM	$235
CCI Telnet Communications Package	$135

APPLE ACCESSORIES AND SOFTWARE



VISICALC $155.00	DB MASTER $159.00
Z-80 SOFTCARD	$259.00
VIDEX BOARD	$259.00
KEYBOARD ENHANCER	$110.00
16K CARD	$169.00
APPLE JOYSTICK	$ 49.00
SUP-R MOD	$ 25.00
CCS CARDS	$ CALL
GALAXIAN	$ 22.95
SPACE ALBUM	$ 35.00
ASTEROIDS	$ 17.95
FLIGHT SIMULATOR	$ 29.00
WIZARD & PRINCESS	$ 28.00
SARGON 2	$ 29.00
MYSTERY HOUSE $ 24.00	HI-RES FOOTBALL $ 35.00

For fast delivery, send certified checks, money orders or call to arrange direct bank wire transfers. Personal or company checks require two to three weeks to clear. All prices are **mail order only** and are subject to change without notice. **Call for shipping charges.**

DEALER (NATIONAL/INTERNATIONAL) INQUIRIES INVITED **Send for FREE Catalogue**

The CPU SHOP
✓298

TO ORDER CALL TOLL FREE 1-800-343-6522
TWX: 710-348-1796 Massachusetts Residents call 617/242-3361

5 Dexter Row, Dept. MC09M
Charlestown, Massachusetts 02129
Hours 10AM-6PM (EST) Mon.-Fri. (Sat. till 5)

Technical Information call 617/242-3361
Massachusetts Residents add 5% Sales Tax
Tandy Corporation Trademark/‡ Digital Research

master charge THE INTERBANK CARD
VISA

✓See List of Advertisers on page 354

Program continues

```
640 GOSUB 670
650 GOTO 150
660 'SUBROUTINE TO CHECK FOR SHIFT-A WHICH STARTS GAME
670 IF INKEY$=CHR$(97) THEN 900
680 RETURN
690 'SUBROUTINE TO DRAW A CIRCLE OR FRACTION OF A CIRCLE
700 'ENTRY VALUES N,S,E,R
710 PN=8*ATN(1)/N
720 SP=SIN(S):CP=COS(S)
730 SD=SIN(PN):CD=COS(PN)
740 K=(E-S)/PN
750 FOR I=1 TO K
760  SET(64+R*CP,24+.4*R*SP)
770  SET(64-R*CP,24+.4*R*SP)
780  IF FL<>1 THEN 810
790  SET(64+R*CP,24-.4*R*SP)
800  SET(64-R*CP,24-.4*R*SP)
810  S=SP*CD+CP*SD
820  CP=-SP*SD+CP*CD
830  SP=S
840 NEXT I
850 RETURN
860 'WAIT ROUTINE
870 FOR WI=1 TO W:NEXT:RETURN
880 END
890 '
900 'WALLS OF JERICHO GAME
910 '
920 'BEST SCORE SO FAR
930 NB=0
940 P=845:R=64:S=128
950 'LOAD QUESTIONS,RESPONSES
960 RESTORE
970 CLS
980 PRINTCHR$(23)
990 PRINT@4,STRING$(24,CHR$(140));
1000 PRINT@70,"WALLS OF JERICHO GAME";
1010 PRINT@132,STRING$(24,CHR$(131))
1020 PRINT
1030 PRINT"PRESS  THE SPACE  BAR  WHEN"
1040 PRINT
1050 PRINT"===>   POINTS TO THE ANSWER"
1060 PRINT
1070 PRINT"THAT GOES WITH THE FLASHING"
1080 PRINT
1090 PRINT"WORD.  YOU HAVE 20 TRIES TO"
1100 PRINT
1110 PRINT"MAKE THE WALLS COME"
1120 PRINT"                 TUMBLING"
1130 PRINT"                       DOWN"
1140 READ NQ 'NUMBER OF QUESTIONS
1150 FOR I=1 TO NQ
1160  READ Q$(I)
1170  Q(I,0)=1
1180  FOR J=1 TO 3
1190   READ Q(I,J)
1200  NEXT J
1210 NEXT I
1220 NC=20 'NUMBER OF CHANCES TO KNOCK DOWN WALL
1230 'NUMBER OF QUESTIONS
1240 DATA 13
1250 'EACH DATA STATEMENT GIVES 9 CHARACTER QUESTION
1260 'FOLLOWED BY QUESTION INDEX, CORRECT ANSWER,
1270 'NUMBER OF ANSWERS
1280 DATA " THE ARK ",1,1,3
1290 DATA " JOSHUA  ",2,2,3
1300 DATA " JERICHO ",3,2,3
1310 DATA "  NOMAD  ",4,2,3
1320 DATA " PROPHET ",4,3,3
1330 DATA "  EGYPT  ",5,1,3
1340 DATA "  SEVEN  ",6,3,3
1350 DATA " JORDAN  ",7,1,3
1360 DATA "  MOSES  ",8,3,3
1370 DATA "MEMORIAL ",9,2,3
1380 DATA " SYMBOL  ",9,1,3
1390 DATA "TRUMPETS ",10,2,3
1400 DATA " RED SEA ",11,3,3
1410 'STRINGS FOR PICTURE ELEMENTS
1420 'BLOCK IN WALL
1430 BF$=STRING$(8,CHR$(143))+CHR$(133)
1440 BR$=STRING$(9,CHR$(143))
1450 H$="         "
1460 'SEQUENCE TO MAKE WALL CRUMBLE
1470 S$(1)=CHR$(153)+CHR$(155)+CHR$(174)+CHR$(161)+CHR$(138)+CHR
$(159)+CHR$(175)+CHR$(182)+CHR$(158)
1480 S$(2)=CHR$(146)+CHR$(173)+CHR$(132)+CHR$(134)+CHR$(153)+CHR
$(145)+CHR$(147)+CHR$(150)+CHR$(146)
1490 S$(3)=CHR$(152)+CHR$(180)+CHR$(188)+CHR$(164)+CHR$(140)+CHR
$(168)+CHR$(172)+CHR$(180)+CHR$(156)
1500 S$(4)=CHR$(136)+CHR$(160)+CHR$(164)+CHR$(136)+CHR$(148)+CHR
$(132)+CHR$(160)+CHR$(176)+CHR$(160)
1510 S$(5)=CHR$(144)+CHR$(176)+CHR$(136)+CHR$(160)+CHR$(144)+CHR
$(176)+CHR$(144)+CHR$(160)+CHR$(160)
1520 S$(6)="         "
1530 'TOWER
1540 T1$=CHR$(191)+CHR$(176)+CHR$(191)+CHR$(176)+CHR$(191)+CHR$(
176)+CHR$(191)+CHR$(176)+CHR$(191)
1550 T2$=STRING$(9,CHR$(191))
1560 T3$=CHR$(128)+STRING$(7,CHR$(191))+CHR$(128)
1570 'GATE
1580 G$=CHR$(131)+CHR$(131)+CHR$(191)+CHR$(131)+CHR$(131)+CHR$(1
31)+CHR$(191)+CHR$(131)
1590 GL$=G$+CHR$(131)
1600 GR$=G$+CHR$(129)
1610 'NAME OVER GATE
1620 N1$=" JERICHO-"
1630 N2$="KEEP OUT "
1640 'SETUP PICTURE MATRIX
1650 'W(12,6,3) DEFINES THE WALL, 6 BLOCKS WIDE AND 12 HIGH
1660 'THE FIRST ELEMENT IS 1 IF A BLOCK IS AT THE LOC'N
1670 'THE SECOND ELEMENT IS THE SCREEN POSITION
1680 'THE THIRD ELEMENT IS THE COMPUTED GOTO INDEX WHICH
1690 'CHOOSES THE BLOCK TO DRAW
```

Program continued

Program continues

```
1700 FOR I=1 TO 9
1710  FOR J=1 TO 6
1720   W(I,J,0)=1
1730   W(I,J,1)=708-(I-1)*64+(J-1)*9
1740   W(I,J,2)=1
1750  NEXT J
1760  W(I,6,2)=6
1770 NEXT I
1780 FOR I=10 TO 12
1790  W(I,1,0)=1:W(I,6,0)=1
1800  W(I,1,1)=131-(I-10)*64:W(I,6,1)=178-(I-10)*64
1810  W(I,1,2)=3+(I-10):W(I,6,2)=3+(I-10)
1820 NEXT I
1830 FOR I=10 TO 12
1840  FOR J=2 TO 5
1850   W(I,J,0)=0
1860  NEXT J
1870 NEXT I
1880 'GATE
1890 FOR I=1 TO 4
1900  W(I,3,2)=2
1910  W(I,4,2)=7
1920 NEXT I
1930 'NAME OVER GATE
1940 W(5,3,2)=8
1950 W(5,4,2)=9
1960 W(5,3,0)=3
1970 W(5,4,0)=3
1980 'WAIT TO PLAY
1990 PRINT@960,"PRESS SPACE BAR TO PLAY....";
2000 IF INKEY$<>" " THEN 2000
2010 'SETUP PICTURE
2020 CLS
2030 FOR J=0 TO 127
2040  SET(J,36)
2050 NEXT J
2060 FOR I=1 TO 12
2070  FOR J=1 TO 6
2080   IF W(I,J,0)=0 THEN 2100
2090   ON W(I,J,2) GOSUB 3220,3250,3280,3310,3330,3350,3380,3420
,3440
2100  NEXT J
2110 NEXT I
2120 FOR I=1 TO 400:NEXT
2130 'MAIN LOOP
2140 'QU IS NUMBER OF QUESTION ANSWERED OK IN ONE PASS
2150 QU=0
2160 FOR QX=1 TO NC
2170  PRINT@22,NC-QX+1;"CHANCES LEFT ";
2180  PRINT@86,"SCORE TO BEAT=";NB;
2190  I=RND(4)
2200  IF RND(0)<.3 THEN I=I+5
2210  J=RND(6)
2220  IF W(I,J,0)<>1 THEN 2190
2230  'PICK A QUESTION
2240  M=RND(NQ-NU)
2250  FOR K=1 TO NQ
2260   IF Q(K,0)>1 THEN 2290
2270   M=M-1
2280   IF M=0 THEN 2300
2290  NEXT K
2300  L=W(I,J,1)
2310  PRINT@L,Q$(K);
2320  'DUMMY INKEY READ TO CLEAR
2330  D$=INKEY$
2340  GOSUB 2990
2350  'PRINT POSSIBLE ANSWERS
2360  ON Q(K,1) GOSUB 3460,3500,3540,3580,3620,3660,3700,3740,37
80,3820,3860
2370  'GET SELECTED ANSWER
2380  FOR AX=1 TO Q(K,3)
2390   PL=P-4+(AX-1)*64
2400   PRINT@PL,"===>";
2410   'FLASH THE QUESTION WHILE LOOKING FOR ANSWER
2420   FOR KK=1 TO 8
2430    FOR LL=1 TO 6
2440     IF KK=INT(KK/2)*2 THEN PRINT@L,H$; ELSE PRINT@L,Q$(K);
2450     IF INKEY$=" " THEN 2520
2460    NEXT LL
2470   NEXT KK
2480   PRINT@PL,"    ";
2490  NEXT AX
2500  GOTO 2380
2510  'SEE IF ANSWER IS RIGHT
2520  PRINT@PL,"***>";
2530  IF AX=Q(K,2) THEN 2660
2540  'WRONG ANSWER
2550  FOR KK=1 TO 5
2560   FOR LL=1 TO 40
2570    PRINT@L," NOPE! ";
2580   NEXT LL
2590   FOR LL=1 TO 40
2600    PRINT@L,Q$(K);
2610   NEXT LL
2620  NEXT KK
2630  PRINT@PL,"    ";
2640  ON W(I,J,2) GOSUB 3220,3250,3280,3310,3330,3350,3380
2650  GOTO 2900
2660  'RIGHT ANSWER
2670  PRINT@L," RIGHT! ";
2680  FOR AX=1 TO 800:NEXT
2690  PRINT@L,Q$(K);
2700  FOR AX=1 TO 400:NEXT
2710  'MARK ANSWER RIGHT
2720  Q(K,0)=Q(K,0)+1
2730  NU=NU+1
2740  IF NU<NQ THEN 2800
2750  FOR M=1 TO NQ
2760   Q(M,0)=1
2770  NEXT M
2780  NU=0
2790  'REMOVE COLUMN DOWN TO BLOCK
2800  GOSUB 3000
2810  PRINT@PL,"    ";
2820  W(I,J,0)=0
2830  'SEE IF ANYTHING IS LEFT
2840  FOR II=1 TO 12
2850   FOR JJ=1 TO 6
2860    IF W(II,JJ,0)=1 THEN 2900
2870   NEXT JJ
2880  NEXT II
2890  GOTO 3120
2900 NEXT QX
2910 'END OF THE GAME -- JERICHO WINS
2920 GOSUB 2990
2930 PRINT@P+R-3,"THE WALL SURVIVED -- BETTER LUCK NEXT TIME!";
2940 FOR I=1 TO 1800:NEXT
2950 CLS
2960 PRINT@468,CHR$(23);"NEXT PLAYER";
2970 FOR I=1 TO 1000:NEXT
2980 GOTO 960
2990 FOR N=P TO P+S STEP R:PRINT@N,CHR$(242);:NEXT:RETURN
3000 'SHAKEDOWN PICTURE
3010 FOR II=12 TO I STEP -1
3020  IF W(II,J,0)=0 GOTO 3090
3030  LL=W(II,J,1)
3040  W(II,J,0)=0
3050  FOR K=1 TO 6
3060   PRINT@LL,S$(K);
3070   FOR AX=1 TO 20:NEXT
3080  NEXT K
3090 NEXT II
3100 GOSUB2990
3110 RETURN
3120 PRINT@707,CHR$(23);"YOU WIN-WELCOME TO JERICHO!";
3130 PRINT@22,CHR$(232);
3140 PRINT@86,CHR$(232);
3150 'CHECK BEST SCORE SO FAR
3160 IF NC-QX>NB THEN NB=NC-QX
3170 GOTO 2940
3180 'DELETE A BLOCK
3190 W(I,J,0)=0
3200 PRINT@W(I,J,1),H$;
3210 RETURN
3220 'DRAW BLOCK IN WALL
3230 PRINT`W(I,J,1),BF$;
3240 RETURN
3250 'DRAW GATE
3260 PRINT@W(I,J,1),GL$;
3270 RETURN
3280 'DRAW TURRET
3290 PRINT@W(I,J,1),T3$;
3300 RETURN
3310 PRINT@W(I,J,1),T2$;
3320 RETURN
3330 PRINT@W(I,J,1),T1$;
3340 RETURN
3350 'RIGHT HAND BLOCK
3360 PRINT@W(I,J,1),BR$;
3370 RETURN
3380 'RIGHT SIDE OF GATE
3390 PRINT@W(I,J,1),GR$;
3400 RETURN
3410 'NAME -- FREE LOCATION
3420 PRINT@W(I,J,1),N1$;
3430 RETURN
3440 PRINT@W(I,J,1),N2$;
3450 RETURN
3460 PRINT@P,"A. A BOX CONTAINING THE LAWS OF GOD";
3470 PRINT@P+R,"B. A STORAGE CONTAINER";
3480 PRINT@P+S,"C. A MUSICAL INSTRUMENT";
3490 RETURN
3500 PRINT@P,"A. AN EGYPTIAN PHARAOH";
3510 PRINT@P+R,"B. LEADER OF THE ISRAELITES";
3520 PRINT@P+S,"C. DEFENDER OF JERICHO";
3530 RETURN
3540 PRINT@P,"A. A CITY IN EGYPT";
3550 PRINT@P+R,"B. A WALLED FORTRESS CITY";
3560 PRINT@P+S,"C. LEADER OF THE ISRAELITES";
3570 RETURN
3580 PRINT@P,"A. RULER OF THE ISRAELITES";
3590 PRINT@P+R,"B. A PERSON WHO MOVES FROM PLACE TO PLACE";
3600 PRINT@P+S,"C. A PERSON WHO SPEAKS OR WRITES FOR GOD";
3610 RETURN
3620 PRINT@P,"A. WHERE GOD HELPED THE PEOPLE ESCAPE FROM SLAVERY
";
3630 PRINT@P+R,"B. A WALLED FORTRESS CITY";
3640 PRINT@P+S,"C. THE WILDERNESS EAST OF THE JORDAN";
3650 RETURN
3660 PRINT@P,"A. THE NUMBER OF TRIBES";
3670 PRINT@P+R,"B. THE NUMBER OF YEARS WANDERING IN THE WILDERNE
SS";
3680 PRINT@P+S,"C. THE NUMBER OF TIMES TO MARCH AROUND THE CITY"
;
3690 RETURN
3700 PRINT@P,"A. THE RIVER CROSSED TO GET TO JERICHO";
3710 PRINT@P+R,"B. A WALLED FORTRESS CITY";
3720 PRINT@P+S,"C. THE LEADER OF THE ISRAELITES";
3730 RETURN
3740 PRINT@P,"A. LED THE ISRAELITES TO JERICHO";
3750 PRINT@P+R,"B. THE PHARAOH WHO RULED EGYPT";
3760 PRINT@P+S,"C. LED THE ISRAELITES OUT OF EGYPT";
3770 RETURN
3780 PRINT@P,"A. OBJECT REMINDING US OF A FEELING OR EVENT";
3790 PRINT@P+R,"B. KEEPS A MEMORY OF A PERSON OR EVENT ALIVE";
3800 PRINT@P+S,"C. A MUSICAL INSTRUMENT";
3810 RETURN
3820 PRINT@P,"A. USED TO ANNOUNCE THE ARRIVAL OF THE ISRAELITES"
;
3830 PRINT@P+R,"B. USED TO KNOCK DOWN THE WALLS OF JERICHO";
3840 PRINT@P+S,"C. USED TO ANNOUNCE THE CAPTURE OF JERICHO";
3850 RETURN
3860 PRINT@P,"A. A SEA BESIDE JERICHO";
3870 PRINT@P+R,"B. DRIED UP FOR JOSHUA SO THE PEOPLE COULD CROSS
";
3880 PRINT@P+S,"C. PARTED FOR MOSES SO THE ISRAELITES COULD ESCA
PE";
3890 RETURN
```

So you followed the article's instructions like a puppy but. . .

Lowercase Done Right

John Burgan
734 Flamingo Way
North Palm Beach, FL 33408

Several recent computer magazine articles have explained how to install a lowercase modification for under $20. All these modifications differ from Radio Shack's, require switches, and have missing lowercase descenders. One even lost the graphics characters.

Using a little logic, I deduced that the Radio Shack modification, which has lowercase descenders and graphics, must contain something more than the changes in these mods. A careful look at the parts list in the *TRS-80 Technical Manual* revealed only one chip with a name that sounds like it has anything to do with lowercase: the Z-29, an MCM6670 character generator chip.

What I Did

With this piece of information, I went to Radio Shack and placed an order for this chip, and after several weeks picked up my new chip with a piece of paper attached stating that its National Parts stock number was AXX3027 and its cost was $12.95. I also picked up an 18-pin socket for this chip for 49 cents, and a 2102L 1K static RAM chip, as required in the Electric Pencil documentation, for $2.69.

First I installed the Electric Pencil lowercase mod, minus the switches (I wired the upper/lower switch in the lower position, and omitted the control key completely). Then I carefully unsoldered and removed the Z-29 chip, installed the 18-pin socket and inserted my AXX3027 chip. (I later found out that the newer keyboards already have a socket installed for Z-29, as well as one for the 2102L (Z-45), thus eliminating the nerve-racking job of unsoldering an 18-pin chip.)

I quickly reassembled the keyboard, booted up NEWDOS-80, and executed the LCDVR/CMD lowercase driver program. My TRS-80 had lowercase descenders and the graphics worked just like before. This is the same mod that Radio Shack sells for $59, but I installed it for only $16.13. Some savings!

How It Is Done

This not a job for someone lacking soldering experience. If you make a mistake, you have voided your warranty and Radio Shack will charge you to fix it (if they can). If you don't know how to work a soldering iron to perfection, take this job to someone who can!

Go to Radio Shack and get the following: a 2102L static RAM chip (RS #276-2501), an 18-pin socket (RS # 276-1992), and order a AXX3027 character-generator chip for the Model I computer from National Parts. You may order directly from National Parts by phone at (817) 870-5600.

When the parts arrive, carefully open your keyboard and find the Z-29 chip. *Carefully* unsolder and replace the Z-29 chip with the 18-inch socket. Place the AXX3027 chip in the socket.

Take the 2102L and bend pins 11 and 12 up parallel with the surface of the chip. (Be careful with this chip, as it's static susceptible.) Solder a piece of light-weight wire (wire-wrap wire preferred) to pins 11 and 12.

Locate the Z-45 chip (one down, and two to the left of the Z-29 chip). If it's in a socket, remove it; piggyback the 2102L to it, and reinsert it with the piggybacked chip back into its socket. If it's not in a socket, carefully piggyback the 2102L to it where it sits, being especially careful not to get any solder bridges between pins.

Run the wire from the 2102L pin 11 to the Z-44 pin 13, and solder it to the trace from the Z-44 pin 13 that goes through the board.

Locate the Z-29 and Z-30 chips. Cut the trace that goes under the Z-29 chip between pins 11 and 12, but cut it out in the open where you can jumper again if necessary.

Solder the wire from the 2102L pin 12 to the Z-27 pin 13.

Reassemble the keyboard and turn it on. You should now have standard uppercase characters, and if a lowercase driver is available and loaded, you should have upper and lowercase with descenders.

A lowercase driver comes with NEWDOS-80, as well as several other DOSes, or one is available from Radio Shack. Until you can get one, Program Listing 1 can be used in a Level II 16K machine. This program takes only 30 bytes of high memory and produces lowercase when the shift key is pressed. ■

Since this article was written, Radio Shack has raised the price of their chip to $34.95. The same chip is available from E.B. Garcia and Assoc., Chicago, IL, for $19.95.

```
10 REM – FOR 16K MACHINES, SET MEMORY SIZE TO 32737
20 POKE 16553, 255: FOR I = 32738 TO 32767 :READJ:POKEI,J:NEXT
30 POKE 16414,226:POKE 16415.127
40 DATA 221,110,3,221,102,4,218,154,4,221,126,5,183,40,1
50 DATA 119,121,254,32,218,6,5,254,128,210,166,4,195,125,4
```

Program Listing.

The Captain 80 Book of

BASIC ADVENTURES

- Over a dozen famous adventures, listed in large easy to read type, ready to be keyed in. Adventures by Boner, Kepner, Powers, Micklus, Forsythe, Greg Hassett and others.
- Includes *Temple of the Sun, Dog Star, Lost Ship, Spider Mountain, Lost Dutchman's Gold, Thunder Road, Sorcerer's Castle, Deadly Dungeon, Atlantean Oddysey* and others. These adventures would cost over $200 is purchased individually!
- Includes a unique *Adventure Generator* program - not available anywhere. YES, this program will actually write another BASIC adventure program, which you may then run. Not even the author of the generator program knows the outcome! *This program alone is worth the price of the entire book!*
- Includes chapters on what an adventure is, how to play adventures, how to write adventures, how to sell your adventure, ten adventure program ideas, and more!
- Study the techniques and methods used by the masters of adventure program writing. All adventure programs listed in this book are in their original form and in full length. Although specifically written for the TRS-80 Model I and III, these programs are adaptable to other computers using Microsoft's BASIC.
- Available at your dealer approximately November 15th, 1981. The suggested retail price of this book is $19.95.
- The BASIC Adventure Book was compiled and edited by Bob Liddil, of The Programmers Guild. It measures 8½ by 11 inches and will contain approximately 250 pages.
- To reserve your copy of this unique book, order NOW for only $14.95 and save. This special pre-publication offer is good only with paid orders until Nov 1st, 1981.
- *ORDER YOURS TODAY!*

Dealer discounts available - please inquire

TRS-80® is a registered trademark of the Tandy Corp.

Pre-Publication SPECIAL $14.95

Send To: 80-U.S. Journal ✓33
3838 South Warner Street
Tacoma, Washington 98409

Please reserve ____ Copy (Copies) of the Basic Adventure Book at the pre-publication price of only $14.95. Payment must be enclosed with this order. Offer good in the United States only.

Name ____________________

Address ____________________

City __________ State ______ Zip __________

Visa/MC ____________________

Exp. Date ______ Signature ____________________

(206) 475-2219 Phone orders accepted for Visa/MC only.

ORDER # 8M

It's done with strings, not mirrors.

The Magic Trick

David D. Busch
515 E. Highland Ave.
Ravenna, OH 44266

Magic Trick is a quick party program that can be used to mystify your non-computer owning friends. I even managed for a few minutes to puzzle a few of my more astute programmer buddies at a recent meeting of my user's group. They watched the trick unfold, tried to think of a fast way of achieving the effect presented using assembly language, and then suddenly realized what was going on.

The trick works like this: The person demonstrating the computer rattles off some patter about how the TRS-80 has marvelous capabilities which include the power to communicate through phone lines, via an RS-232 or through the air, using brainwaves.

Someone is certain to question a computer's ESP ability. Quickly, the magician loads this program from cassette or disk. If saved on disk, a mysterious name such as DEMO/ESP may be used. Questions are tossed out from the onlookers, and the operator proceeds to type innocuous messages into the keyboard, such as "Hello there, computer," or "Are you ready for another question?"

As if by magic, the computer responds with a correct answer. . . then another. . . and another.

You may get four or five questions answered before someone catches on. As with all magic tricks, the fewer times it is presented, the better the effect. And, as with most good magic, the secret is in misdirection. Though the operator appears to be typing in innocent questions, he or she is actually entering the *answers* at that time. Because everyone will be raptly reading the CRT screen, no one will notice that the keys that are being hit do not match those which are appearing on the screen.

The Secret

Here's how the trick is set up: seven (or more—add your own) messages are stored in data lines within the program itself. As the trick is run, and INKEY$ strobing loop repeatedly looks at the keys for input. If a key is hit, in most cases (the exceptions will be discussed later) the character is added to a string variable, AN$.

In addition, as each character is entered, one letter is printed to the screen. It is not the same character as the one entered. Instead, one more letter of the preselected message is revealed. To the unsuspicious, it looks as if the CRT is echoing the characters typed by the person doing the demonstration.

A dummy message is read from data in line 130. The INKEY$ loop repeats in line 140 until a key is hit. Then, A$ is checked in line 150 to see if the key struck was Enter (CHR$(13)). If so, entry is complete, and the routine goes to lines 210–220. Otherwise, the program checks to see if the character entered was a period.

If not, then Flag = 0, and A$ is added to AN$. A counter, CU, is incremented by one and used to command the next character of the pre-selected message, B$, to be printed to the screen (line 190).

Control returns to line 140 for additional input. The operator should be a touch typist, but at least do not obviously stare at the keyboard. It helps the misdirection to look the audience in the eyes, or to guide their gazes to the CRT.

The operator can enter any message or answer that he or she wishes, as long as it is shorter than the dummy message string being printed to the screen. When the end of the input message is reached, the magician should enter a period. When this is done, then Flag is set to one, and all other input is *not* added to AN$. After entering the period, the operator should just strike random keys to allow the rest of the dummy message to print out. A punctuation mark of some sort should be included in each message in the data lines so that the demonstrator will have a quick means of knowing when to stop typing.

The trick should not continue more than three or four messages, or else the onlookers will start to get suspicious. They may tell you what messages to type into the computer, and, of course, you have no control over what appears on the screen. However, to keep the over-enthusiastic from blowing their cover with an Out-of-Data message, this program loops through all seven messages, then repeats through a Restore and GOTO 120 in lines 300–310.

A good quickie to liven up parties, Magic Trick provides an easy way to mystify the unwary with the wondrous powers of the TRS-80. ■

```
100 CLEAR 1000
110 CLS
120 FOR N = 1 TO 7
130 READ B$
140 A$ = INKEY$:IF A$ + ""GOTO 140
150 IF A$ = CHR$(13) GOTO 210
160 IF A$ = "." FLAG = 1
170 IF FLAG< >1 THEN AN$ = AN$ + A$
180 CU = CU + 1
190 PRINT MID$(B$,CU,1);
200 GOTO 140
210 PRINT
220 PRINT AN$
230 INPUT "PRESS ENTER FOR ANOTHER QUESTION";D$
240 AN$ = ""
250 CU = 0
260 FLAG = 0
270 NEXT N
280 RESTORE
290 GOTO 120
300 DATA "HELLO THERE, COMPUTER.","ARE YOU READY FOR ANOTHER
    QUESTION?","WE HAVE ANOTHER ONE FOR YOU.","NOW TRY THIS
    ONE.","PRETTY GOOD FOR A DUMB MACHINE.","PLEASE TRY ONE
    MORE.","YOU'VE GOT US FOOLED PRETTY GOOD."
```

Program Listing

Poor Man's Floppy

HIGH SPEED CASSETTE SYSTEM

FOR TRS-80*

Now the widely acclaimed JPC Cassette System is available for your TRS-80 computer. The price is only $90.00*

[Reprint of June 1980 Review, *80 Microcomputing*]

TC-8 Cassette System
JPC Products
Albuquerque, NM
Kit: $90
Assembled: $120

by Carl A. Kollar

I guess I don't have to tell any TRS-80 owners how frustrating the cassette system that comes with the computer can be. Even with the factory mod that's available, the annoyance of loading and checking programs becomes just barely tolerable.

If you're like me, after you've just plunked down a chunk of money for a Level II 16K machine, "you ain't got nuttin left" for even one disk drive at 500 bucks apiece. So you suffer.

A reasonable alternative is the Exatron Stringy Floppy (ESF). This will cost you about 250 bucks and totally eliminates your loading and saving problems, automatically and fast. I've had one of these for about six months and love it!

But, if the price is still too steep, have I got a device for you!

The Device

The February 1980 issue of *Microcomputing* had an ad that intrigued the hell out of me. It was a high-speed cassette system by JPC Products acclaimed as a "poor man's floppy." It made all sorts of seemingly ridiculous claims such as "loads five times faster," "stores 50,000 bytes on a 10-minute cassette," "less than one bad load in a million bytes with the volume control anywhere between one and eight."

All this for a measly [90] bucks? How could this be? A call to Albuquerque answered a few questions: Yes, it had its own power supply, and, it stored programs five times faster because it utilized higher density data. The computer outputs the information at a higher rate out of the rear keyboard connector.

The ad had even claimed anyone could build it even if you have never soldered before. JPC would make it work, if you couldn't—for free. I was sold. I placed my order, and it arrived about two months later (parts shortage).

I work in electronics, so I found the unit exceptionally easy to build. It took about an hour. The manual is superb. (That's better than great.) It was clear, concise and exact with no ambiguities. Important parts placements are stressed (polarity markings on electrolytics, bands on diodes, etc.).

JPC was right! With these instructions, you couldn't go wrong. The board quality is excellent. It is double-sided and parts locations are clearly marked on the component side of the board. There are no jumper wires to install. JPC utilizes PC traces and plated-through holes for connections to traces on the other side of the board.

Also, there are absolutely no adjustments or settings to bother with.

The documentation is a sheaf of 8½ × 11 papers stapled together. It is written in the nicest format I've seen in a while. Each command and/or subjects is covered on its own sheet in large type. All explanations are in easy to read English—not computerese.

Commands and Features

SAVE"filename": Saves your BASIC program on cassette.
LOAD: Reads the next BASIC program from the cassette.
LOAD"filename": Searches for and loads the specified file from cassette.
LOAD? and LOAD?"filename": Reads file from cassette, and compares contents to memory.
LOADN: Prints a list of all the programs on a cassette, until interrupted by the "break" key.
LOADN"filename": Same as above except the tape will stop at the end of the program named.
KILL: Removes the file manager program from memory so that the extra memory can be used by large programs.
RSET: Allows the operator to rewind and position the tape on tape recorders that have these functions tied to the motor control jack.
RUN"filename": TC-8 searches for a specified program and runs it immediately.
PUT"filename": Same as SAVE "filename", except it is for use with system tapes.
GET: Same as LOAD, except it is for use with system tapes.
GET"filename": Same as LOAD "filename", except it is for use with system tapes.
GET? and GET?"filename": Same as LOAD? and LOAD?"filename", except it is for use with system tapes.
GETN and GETN"filename": Same as LOADN and LOADN"filename", except it is for use with system tapes.
OPEN: Required before cassette input or output of a data file can be attempted.
CLOSE: Required to end a cassette data file.
PRINT#: Allows numerical or string data to be output to a cassette file.
INPUT#: Allows numerical or string data to be input from a cassette file.

I haven't counted them, so I don't know about the "one load in a million bytes" claim, but my son, Anthony (age 11), loaded about 30 of his programs from his Radio Shack format tape to a new TC-8 format tape. He's run them all and found no bad loads.

Unlike the standard tape system, you can position your tape anywhere before the program you want and not have to look for a blank spot between programs. The TC-8 patiently waits for the program you want and then starts loading without getting confused by the portion of the previous program you just fed it.

Try that on your regular cassette system; you'll wear out the reset button. ■

ORDER NOW

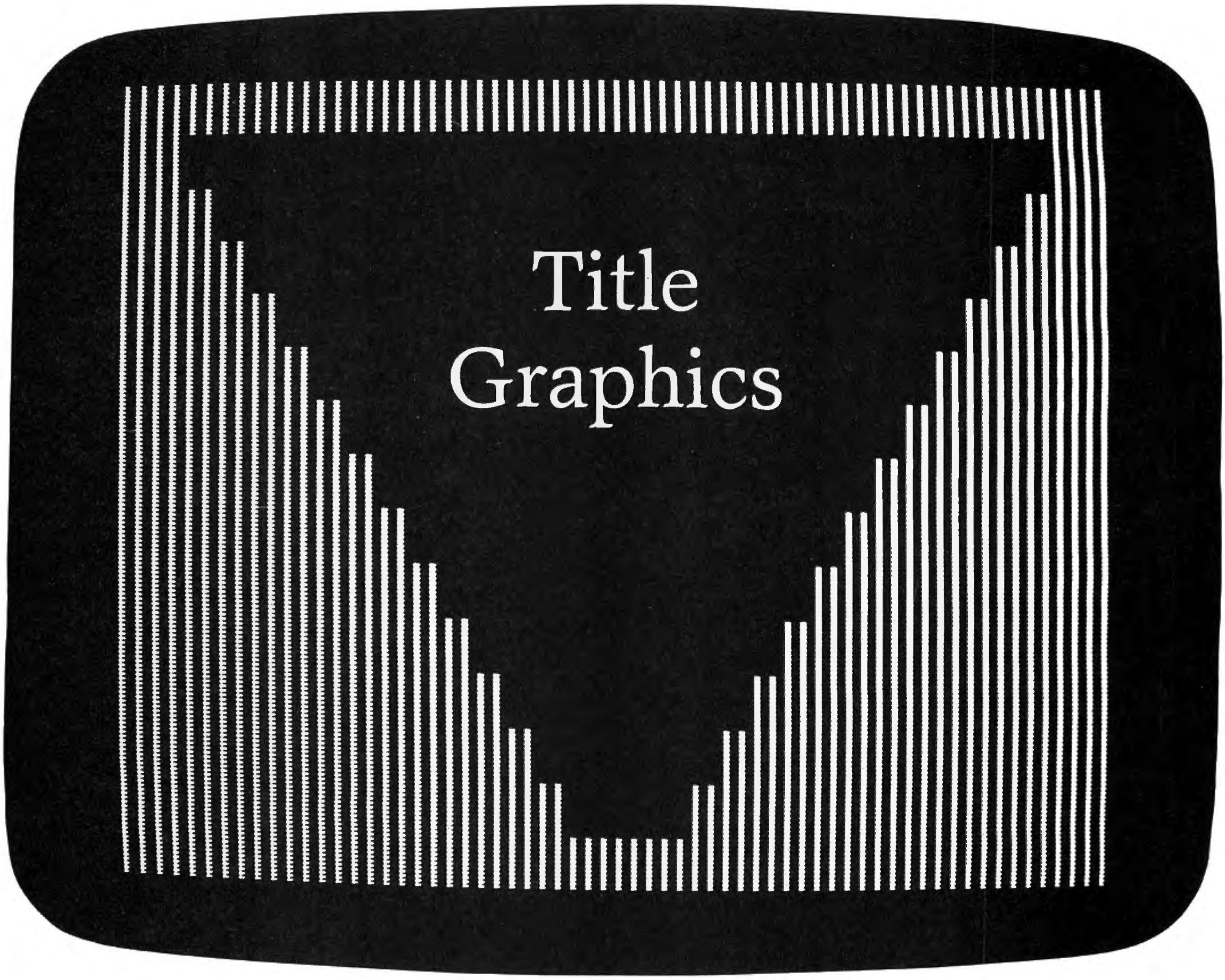
To order your TC-8 kit, send your check or money order for $90.00 plus $3.50 postage and handling to JPC PRODUCTS CO., 12021 Paisano Ct., Albuquerque, NM 87112 (New Mexico residents add 4% sales tax). Credit card orders accepted by phone or mail. Personal checks will delay shipment. We will otherwise immediately ship you the TC-8 kit, the cabinet, the ribbon cable, the power adapter, an instruction manual, and a cassette containing the software.

✓190

JPC PRODUCTS CO.
Phone (505) 294-4623
12021 Paisano Ct.
Albuquerque, N.M. 87112

✓See List of Advertisers on page 354

Title Graphics



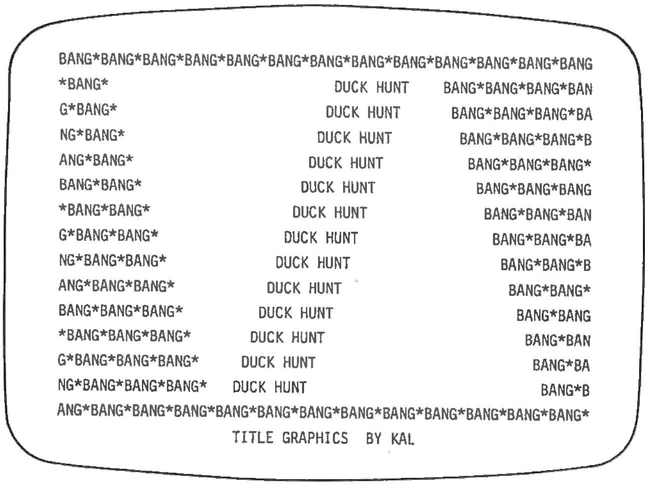


Fig. 1

Francis S. Kalinowski
16 N. Alder Drive
Orlando, FL 32807

If you have programs that start abruptly with small, insignificant titles, you might like to add graphic title display routines to them, so that your Level II Basic programs will identify themselves conspicuously.

All the graphic title programs in this article begin with a statement 0. This statement clears the display screen and necessary string space, and jumps to a graphic title display routine. Display routines start at statement 65500 or higher, where they won't alter existing program statements.

A display routine will have seven to fifteen statements, depending on the complexity of its graphics. Each routine displays its uniquely formatted graphic title, holds it steady for

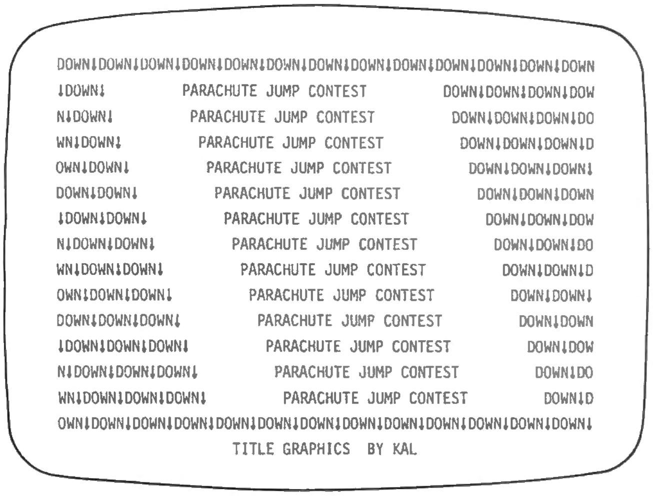


Fig. 2

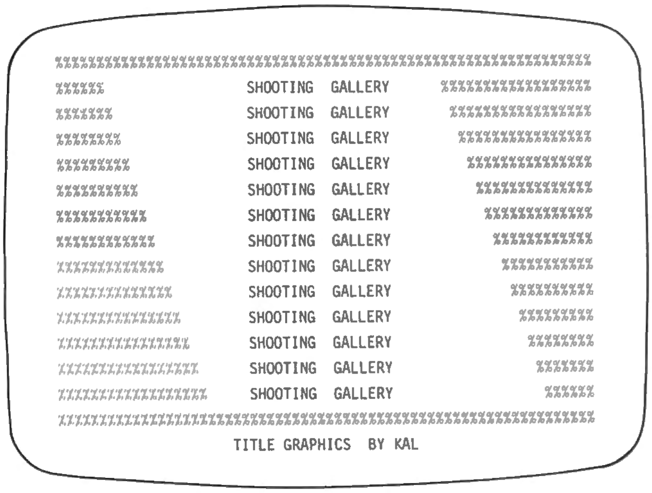


Fig. 3

convenient reading, and returns display control to statement 0.

Statement 0 adds more title-hold timing if needed, restores string space to 50, and drops display control to your existing program's first statement.

Statement 1 GOTO1 is included in each graphic title program for debugging convenience. It holds a title displayed on the screen until the Break key is pressed. Statement 1 must be deleted after your program title has been fitted and adjusted within a display routine and you are satisfied with the routine's operation.

Parallelogram Titles

Right and left-slant parallelogram formats can be varied to accommodate short, medium or long program titles. Program Listing 1 displays a 13-line title stack within a right-slant parallelogram window. A repeated five-character word or group of characters makes up the background frame. B$ in statement 65508 defines the five-character string that prints the background. Statement 65516 specifies the title stacking format and the program title.

Program Listing 1a prints the title stack Duck Hunt downward on a left slant (Fig. 1). Listing 1a can handle titles up to 14 characters above eight in your title—subtract one from 97 and from 854 in statement 65518; add one to 97 and to 854 for every two characters below eight. Statement 65520 may be deleted or used to display a program credit line at the bottom of the screen.

Program Listing 1b prints its title stack straight upward within a percent sign background frame (Fig. 2). This variation can handle titles up to 26 character positions long. For every two characters above 16 in your program title, subtract one from 855 and 87 in statement 65516. Add one to 855 and 87 for every two characters below 16.

Program Listing 1c prints its title stack downward on a right slant (Fig. 3). This variation can handle titles up to 38 character positions long. For every two characters above 22 in your title, subtract one from 79 and 859 in statement 65516. Add one to 79 and 859 for every two characters below 22.

Program Listing 1d prints its title stack straight downward within a graphic character background frame. You may use any graphic character ASCII code (129 through 191) instead of the 132 code in statement 65508. Using B$ = STRING$(5,191), for example, provides a solid white background frame. Listing 1d can

```
Ø CLS: CLEAR75: GOSUB65508: FORU=1TO1999: NEXT: CLEAR5Ø
1 GOTO1:'(DELETE THIS STATEMENT WHEN TITLING A PROGRAM)
655Ø8 B$="BANG*":              '(VARIATION 1, SEE FIG. 1)
6551Ø FORU=ØTO65STEP5: PRINT@U,B$: NEXT
65512 FORU=11ØTO845STEP65: PRINT@U,B$+B$+B$+B$+B$;:NEXT
65514 FORU=89ØTO955STEP5: PRINT@U,B$;: NEXT:
       FORI=1TO2ØØ: NEXT
65516 FORU=97TO854STEP63: PRINT@U,"DUCK HUNT";
65518 FORI=1TO9: NEXTI,U
6552Ø PRINT@981,"TITLE GRAPHICS BY KAL";: RETURN
65522 '(STATEMENT CHANGES FOR VARIATION 2,  SEE FIG. 2)
      '655Ø8 B$="%%%%%":
      '65516 FORU=855TO87STEP-64: PRINT@U,"SHOOTING
             GALLERY";
65524 '(STATEMENT CHANGES FOR VARIATION 3,  SEE FIG. 3)
      '655Ø8 B$="DOWN"+CHR$(92):
      '65516 FORU=79TO859STEP65: PRINT@U,"PARACHUTE
             JUMP CONTEST";
65526 '(STATEMENT CHANGES FOR VARIATION 4, SEE PHOTO 1)
      '655Ø8 B$=STRING$(5,132):    '(USE ANY ASCII CODE)
      '65516 FORU=84TO852STEP64: PRINT@U,"GRAPHIC
             CHARACTER TITLE";
```

Program Listing 1a

```
Ø CLS:CLEAR75:GOSUB655Ø8:FORU=1TO1999:NEXT:CLEAR5Ø:
1 GOTO1
655Ø8 B$="%%%%%":
6551Ø FORU=ØTO65STEP5: PRINT@U,B$: NEXT
65512 FORU=11ØTO845STEP65: PRINT@U,B$+B$+B$+B$+B$;:NEXT
65514 FORU=89ØTO955STEP5: PRINT@U,B$;: NEXT:
       FORI=1TO2ØØ: NEXT
65516 FORU=855TO87STEP-64: PRINT@U,"SHOOTING  GALLERY";
65518 FORI=1TO9: NEXTI: NEXTU
6552Ø PRINT@981,"TITLE GRAPHICS BY KAL";: RETURN
```

Program Listing 1b

```
Ø CLS:CLEAR75:GOSUB655Ø8:FORU=1TO1999:NEXT:CLEAR5Ø:
1 GOTO1
655Ø8 B$="DOWN"+CHR$(92)
6551Ø FORU=ØTO65STEP5: PRINT@U,B$: NEXT
65512 FORU=11ØTO845STEP65: PRINT@U,B$+B$+B$+B$+B$;:NEXT
65514 FORU=89ØTO955STEP5: PRINT@U,B$;: NEXT:
       FORI=1TO2ØØ: NEXT
65516 FORU=79TO859STEP65: PRINT@U,"PARACHUTE JUMP CONTEST";
65518 FORI=1TO9: NEXTI: NEXTU
6552Ø PRINT@981,"TITLE GRAPHICS BY KAL";: RETURN
```

Program Listing 1c

```
Ø CLS: CLEAR75: GOSUB655Ø8: FORU=1TO1999: NEXT: CLEAR5Ø
1 GOTO1
655Ø8 B$=STRING$(5,132):          '(USE ANY ASCII CODE)
6551Ø FORU=ØTO65STEP5: PRINT@U,B$: NEXT
65512 FORU=11ØTO845STEP65: PRINT@U,B$+B$+B$+B$+B$;:NEXT
65514 FORU=89ØTO955STEP5: PRINT@U,B$;: NEXT:
       FORI=1TO2ØØ: NEXT
65516 FORU=84TO852STEP64: PRINT@U,"GRAPHIC CHARACTER TITLE";
65518 FORI=1TO9: NEXTI: NEXT
6552Ø PRINT@981,"TITLE GRAPHICS BY KAL";: RETURN
```

Program Listing 1d

Program Listing 1. Right Slant Parallelograms handle titles up to 26 character positions long. Numbers 84 and 852 in statement 65516 define the title stack start and end points. Decrease or increase those numbers by one for every two characters above or below 22, respectively, in your program title.

Program Listing 2 displays a 13-line title stack within a left-slant parallelogram window. As in Listing 1, B$ in statement 65508 defines the five-character background printing string. Also, statement 65516 specifies the title and its stacking format.

Program Listing 2a prints its title stack downward on a left slant within a slash symbol background frame (Fig. 4). Listing 2a can handle titles up to 41 character positions long. Numbers 92 and 848 in statement 65516 define the title stack start and end points. Decrease or increase both numbers by one for every two characters above or below 17, respectively, in your title.

Program Listing 2b prints its title stack straight upward within an up-arrow background frame (Fig. 5). This variation can handle titles up to 29 character positions long. Numbers 855 and 87 in statement 65516 define the title stack start and end points. Decrease or increase both numbers by one for every two characters above or below 17, respectively, in your title.

Program Listing 2c prints its title stack downward on a right slant within a graphic character background frame. You may use any graphic character ASCII code (129 through 191) instead of 179 in statement 65508. This variation can handle titles up to 17 character positions long. Add one to statement 65516 numbers 81 and 861 for every two characters below 17 in your program title.

Moving Title

Program Listing 3 displays a moving title stack within a changing-pattern background frame. Four graphic character strings in statements 65510 through 65516 provide the changing maze-like patterns. A 13-line title stack moves left to right across the background frame's window as the frame pattern continually changes.

This program can handle moving titles of up to 20 character length. Subtract one from T = 854 and from R = 74 in statement 65508 for every two characters above 16 in your program title. Add one to 854 and 74 for every two characters below 16. Replace "The Moving Maze" in statement 65524 with your program title.

Diamond Titles

The Program Listing 4 displays one title line within a diamond-shaped background frame. Statement 65510

Photo 1. Program Listing 1d Variation 4

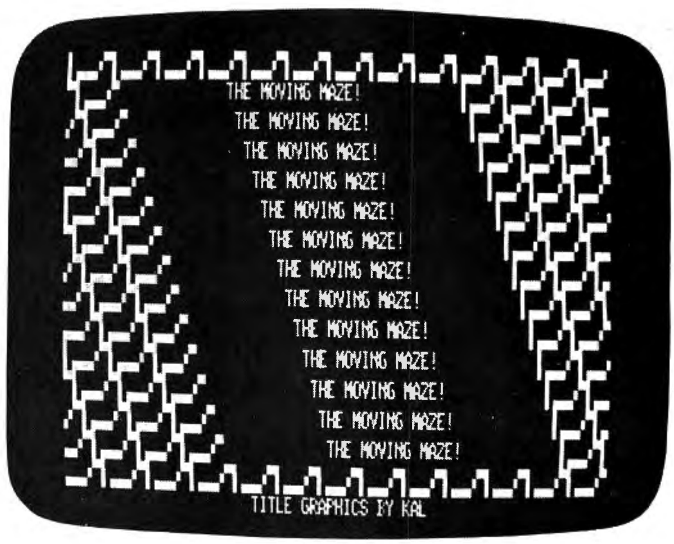


Photo 3. Program 3 Moving Title

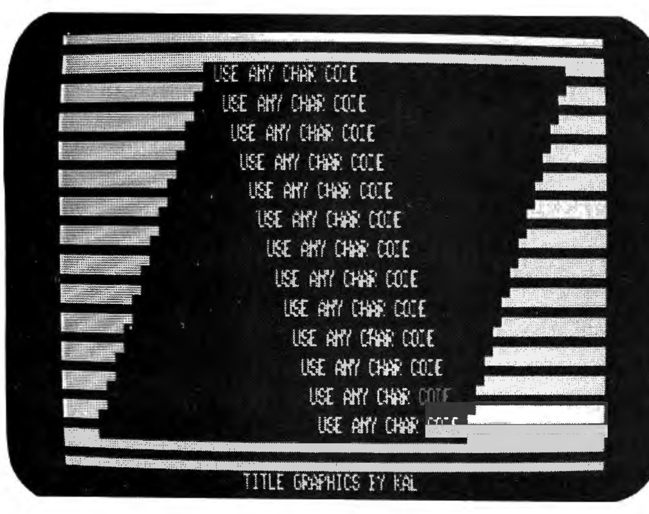


Photo 2. Program Listing 2c Variation 3

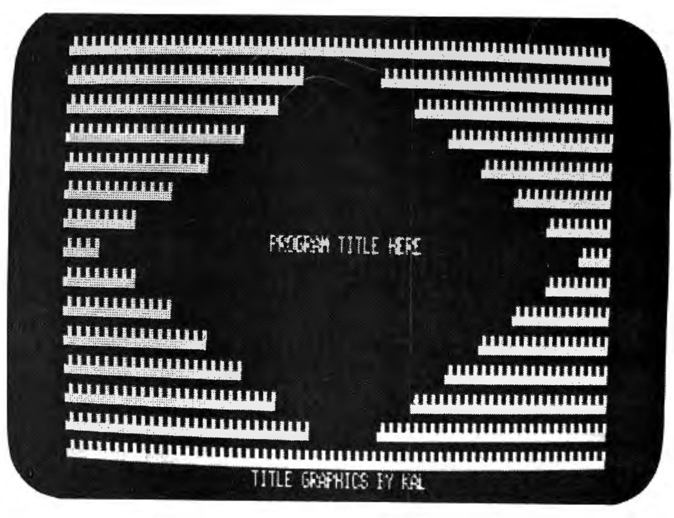


Photo 4. Program Listing 4 Random Graphics

See List of Advertisers on page 354

specifies random characters for printing the background frame. Each new character reprints the background from top and bottom to its center. The characters may be letters, numbers, symbols, or graphic patterns. You may select any type character group by changing statement 65110 as described in REM statements 65525 and 65526.

The 10-second countdown variation (Program Listing 4e) starts with a background frame made up of the numeral nine (Fig. 6), and counts down at one second intervals to an all zeros background frame. The diamond title program can handle 52 character positions. Increase or decrease the statement 65518 PRINT@ value to accommodate your program title. The credit line statement (65511) may be used or omitted.

Wedge Title

Program Listing 5 displays two title lines within a wedge-shaped background frame (Photo 5). The background frame may be made up of any printable character, described in REM statement 65514. Increase or decrease the statement 65522 and 65524 PRINT@ values to accommodate your title and credit lines. Either statement may be omitted.

Spiral Title

Program Listing 6 displays one title line within a graphic spiral background frame. This program can handle three title or credit lines up to 30 characters long. Statement 65522 prints the single line title. Increase or decrease that statement's PRINT@ value to accommodate your program title.

Additional title lines may be displayed with their centers at PRINT@ 415 and PRINT@ 543. Decrease these PRINT@ values by one for every two characters in your additional title lines. Include the upper and lower lines by using statement numbers 65521 and 65523.

Lattice Title

Program Listing 7 displays two title lines within a latticed background frame. Each line can handle a title up to 30 character positions long. Adjust statement 65522 and 65524

```
Ø CLS: CLEAR75: GOSUB65508: FORU=1TO1999: NEXT: CLEAR5Ø
1 GOTO1:'(DELETE THIS STATEMENT WHEN TITLING A PROGRAM)
655Ø8 B$="/////":                '(VARIATION 1, SEE FIG. 4)
6551Ø FORU=ØTO55STEP5:PRINT@U,B$: NEXT
65512 FORU=6ØTO816STEP63: PRINT@U,B$+B$+B$+B$;: NEXT
65514 FORU=879TO959STEP5: PRINT@U,B$;: NEXT
65515 PRINT@96Ø,"     ";
65516 FORU=92TO848STEP63: PRINT@U,"PROGRAM TITLE HERE";
65518 FORI=ØTO9: NEXTI,U
6552Ø PRINT@981,"TITLE GRAPHICS BY KAL";: RETURN
65522 '(STATEMENT CHANGES FOR VARIATION 2,  SEE FIG. 5)
      '655Ø8 B$=STRING$(5,91):
      '65516 FORU=855TO87STEP-64: PRINT@U,"UP, UP, AND
             AWAY!";
65524 '(STATEMENT CHANGES FOR VARIATION 3, SEE PHOTO 2)
      '655Ø8 B$=STRING$(5,179):    '(USE ANY ASCII CODE)
      '65516 FORU=81TO861STEP65: PRINT@U,"USE ANY CHAR
             CODE";
```

Program Listing 2a

```
Ø CLS: CLEAR75: GOSUB655Ø8: FORU=1TO999: NEXT: CLEAR5Ø
1 GOTO1
655Ø8 B$=STRING$(5,91):
6551Ø FORU=ØTO55STEP5:PRINT@U,B$: NEXT
65512 FORU=6ØTO816STEP63: PRINT@U,B$+B$+B$+B$;: NEXT
65514 FORU=879TO959STEP5: PRINT@U,B$;: NEXT
65515 PRINT@96Ø,"     ";: FORI=1TO199: NEXT
65516 FORU=855TO87STEP-64: PRINT@U,"UP, UP, AND AWAY!";
65518 FORI=ØTO9: NEXTI: NEXTU
6552Ø PRINT@981,"TITLE GRAPHICS BY KAL";: RETURN
```

Program Listing 2b

```
Ø CLS: CLEAR75: GOSUB655Ø8: FORU=1TO1999: NEXT: CLEAR5Ø
1 GOTO1
655Ø8 B$=STRING$(5,179):          '(USE ANY ASCII CODE)
6551Ø FORU=ØTO55STEP5:PRINT@U,B$: NEXT
65512 FORU=6ØTO816STEP63: PRINT@U,B$+B$+B$+B$;: NEXT
65514 FORU=879TO959STEP5: PRINT@U,B$;: NEXT
65515 PRINT@96Ø,"     ";: FORI=1TO199: NEXT
65516 FORU=81TO861STEP65: PRINT@U,"USE ANY CHAR CODE";
65518 FORI=ØTO9: NEXTI: NEXTU
6552Ø PRINT@981,"TITLE GRAPHICS BY KAL";: RETURN
```

Program Listing 2c

Program Listing 2. Left Slant Parallelograms

```
////////////////////////////////////////////////////////////////
////////////////       PROGRAM TITLE HERE                  /////
///////////////       PROGRAM TITLE HERE                  //////
//////////////       PROGRAM TITLE HERE                  ///////
/////////////       PROGRAM TITLE HERE                  ////////
////////////       PROGRAM TITLE HERE                  /////////
///////////       PROGRAM TITLE HERE                  //////////
//////////       PROGRAM TITLE HERE                  ///////////
/////////       PROGRAM TITLE HERE                  ////////////
////////       PROGRAM TITLE HERE                  /////////////
///////       PROGRAM TITLE HERE                  //////////////
//////       PROGRAM TITLE HERE                  ///////////////
/////       PROGRAM TITLE HERE                  ////////////////
////       PROGRAM TITLE HERE                  /////////////////
////////////////////////////////////////////////////////////////
                     TITLE GRAPHICS  BY KAL
```

Fig. 4

```
Ø CLS: CLEAR75: GOSUB655Ø6: CLEAR5Ø:      '(SEE PHOTO 3)
1 GOTO1:'(DELETE THIS STATEMENT WHEN TITLING A PROGRAM)
655Ø6 PRINT@981,"TITLE GRAPHICS BY KAL";: T=854: R=74
655Ø8 T=T+1: R=R+1: Q=Q+1: ONQ GOTO 65512,65514,65516,
      6551Ø,65512,65514,65516,6551Ø,65512,65514,65516
6551Ø B$=CHR$(15Ø)+CHR$(141)+CHR$(172)+CHR$(178)+
      CHR$(174): GOTO65518
65512 B$=CHR$(137)+CHR$(14Ø)+CHR$(174)+CHR$(171)+
      CHR$(131): GOTO65518
65514 B$=CHR$(151)+CHR$(171)+CHR$(186)+CHR$(162)+
      CHR$(167): GOTO65518
65516 B$=CHR$(149)+CHR$(176)+CHR$(176)+CHR$(152)+
      CHR$(131)
65518 FORU=ØTO65STEP5: PRINT@U,B$;: NEXT
6552Ø FORU=11ØTO845STEP65: PRINT@U,B$+B$+B$+B$+B$;:NEXT
65522 FORU=89ØTO955STEP5: PRINT@U,B$;: NEXT
65524 FORU=T TO R STEP-65:PRINT@U," THE MOVING MAZE!";:
      NEXT
65526 IF T<866 GOTO655Ø8
65528 FORU=1TO2ØØ: NEXT: RETURN
```

Program Listing 3. Moving Title

```
↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑
↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑       UP, UP, AND AWAY!                  ↑↑↑↑↑
↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑        UP, UP, AND AWAY!                 ↑↑↑↑↑↑
↑↑↑↑↑↑↑↑↑↑↑↑↑↑         UP, UP, AND AWAY!                ↑↑↑↑↑↑↑
↑↑↑↑↑↑↑↑↑↑↑↑↑          UP, UP, AND AWAY!               ↑↑↑↑↑↑↑↑
↑↑↑↑↑↑↑↑↑↑↑↑           UP, UP, AND AWAY!              ↑↑↑↑↑↑↑↑↑
↑↑↑↑↑↑↑↑↑↑↑            UP, UP, AND AWAY!             ↑↑↑↑↑↑↑↑↑↑
↑↑↑↑↑↑↑↑↑↑             UP, UP, AND AWAY!            ↑↑↑↑↑↑↑↑↑↑↑
↑↑↑↑↑↑↑↑↑              UP, UP, AND AWAY!           ↑↑↑↑↑↑↑↑↑↑↑↑
↑↑↑↑↑↑↑↑               UP, UP, AND AWAY!          ↑↑↑↑↑↑↑↑↑↑↑↑↑
↑↑↑↑↑↑↑                UP, UP, AND AWAY!         ↑↑↑↑↑↑↑↑↑↑↑↑↑↑
↑↑↑↑↑↑                 UP, UP, AND AWAY!        ↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑
↑↑↑↑↑                  UP, UP, AND AWAY!       ↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑
↑↑↑↑                   UP, UP, AND AWAY!      ↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑
↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑
                      TITLE GRAPHICS BY KAL
```

Fig. 5

PRINT@ values to accommodate your program title lines.

Animated Title

Program Listing 8 displays an animated title, shown in the three frames of Fig. 7. The program first fills the display screen with grass (exclamation points). Next, two caterpillars,

```
Ø CLS: CLEAR1ØØ: W=58: GOSUB655Ø8: CLEAR5Ø
1 GOTO1:'(DELETE THIS STATEMENT WHEN TITLING A PROGRAM)
655Ø8 X=52: Y=1ØØ: Z=8Ø8
6551Ø W=RND(5Ø)+136:    '(RANDOM GRAPHICS, SEE PHOTO 4)
65511 PRINT@981,"TITLE GRAPHICS  BY KAL";
65512 PRINT@Ø,STRING$(92,W);: PRINT@868,STRING$(92,W);
65514 PRINT@Y,STRING$(X,W);: PRINT@Z,STRING$(X,W);
65516 X=X-8: Y=Y+68: Z=Z-6Ø: IF X>4 GOTO65514
65518 PRINT@471,"PROGRAM TITLE HERE";
65522 FORU=1TO11Ø: NEXT
65524 V=V+1: IF V<1Ø GOTO655Ø8 ELSE RETURN
65525 '-----------------------------------------------
      'FOR RANDOM LETTERS, CHANGE 6551Ø TO W=RND(1Ø)+47
      'FOR RANDOM NUMBERS, CHANGE 6551Ø TO W=RND(26)+64
65526 'FOR A 1Ø-SECOND COUNTDOWN, CHANGE 6551Ø TO W=W-1
      'FOR RANDOM SYMBOLS, CHANGE 6551Ø TO W=RND(15)+32
           OR TO W=RND(7)+57
      '-----------------------------------------------
```

Program Listing 4a. RND Graphics

```
Ø CLS: CLEAR1ØØ: W=58: GOSUB655Ø8: CLEAR5Ø:
1 GOTO1
655Ø8 X=52: Y=1ØØ: Z=8Ø8
6551Ø W=RND(26)+64
65511 PRINT@981,"TITLE GRAPHICS  BY KAL";
65512 PRINT@Ø,STRING$(92,W);: PRINT@868,STRING$(92,W);
65513 PRINT@959,CHR$(W);
65514 PRINT@Y,STRING$(X,W);: PRINT@Z,STRING$(X,W);
65516 X=X-8: Y=Y+68: Z=Z-6Ø: IF X>4 GOTO65514
65518 PRINT@471,"PROGRAM TITLE HERE";
65522 FORU=1TO11Ø: NEXT
65524 V=V+1: IF V<1Ø GOTO655Ø8 ELSE RETURN
```

Program Listing 4b. RND Letters

```
Ø CLS: CLEAR1ØØ: W=58: GOSUB655Ø8: CLEAR5Ø
1 GOTO1
655Ø8 X=52: Y=1ØØ: Z=8Ø8
6551Ø W=RND(1Ø)+47
65511 PRINT@981,"TITLE GRAPHICS  BY KAL";
65512 PRINT@Ø,STRING$(92,W);: PRINT@868,STRING$(92,W);
65514 PRINT@Y,STRING$(X,W);: PRINT@Z,STRING$(X,W);
65516 X=X-8: Y=Y+68: Z=Z-6Ø: IF X>4 GOTO65514
65518 PRINT@471,"PROGRAM TITLE HERE";
65522 FORU=1TO11Ø: NEXT
65524 V=V+1: IF V<1Ø GOTO655Ø8 ELSE RETURN
```

Program Listing 4c. RND Numbers

```
Ø CLS: CLEAR1ØØ: W=58: GOSUB655Ø8: CLEAR5Ø
1 GOTO1
655Ø8 X=52: Y=1ØØ: Z=8Ø8
6551Ø W=RND(15)+32
65511 PRINT@981,"TITLE GRAPHICS  BY KAL";
65512 PRINT@Ø,STRING$(92,W);: PRINT@868,STRING$(92,W);
65514 PRINT@Y,STRING$(X,W);: PRINT@Z,STRING$(X,W);
65516 X=X-8: Y=Y+68: Z=Z-6Ø: IF X>4 GOTO65514
65518 PRINT@471,"PROGRAM TITLE HERE";
65522 FORU=ØTO11Ø: NEXT
65524 V=V+1: IF V<1Ø GOTO655Ø8 ELSE RETURN
```

Program Listing 4d. RND Symbols

```
Ø CLS: CLEAR1ØØ: W=58: GOSUB655Ø8: CLEAR5Ø
1 GOTO1
655Ø8 X=52: Y=1ØØ: Z=8Ø8
6551Ø W=W-1
65511 PRINT@981,"TITLE GRAPHICS  BY KAL";
65512 PRINT@Ø,STRING$(92,W);: PRINT@868,STRING$(92,W);
65514 PRINT@Y,STRING$(X,W);: PRINT@Z,STRING$(X,W);
65516 X=X-8: Y=Y+68: Z=Z-6Ø: IF X>4 GOTO65514
65518 PRINT@471,"PROGRAM TITLE HERE";
65522 FORU=ØTO11Ø: NEXT
65524 V=V+1: IF V<1Ø GOTO655Ø8 ELSE RETURN
```

Program Listing 4e. Countdown

Program Listing 4. Diamond Titles

working in opposite directions, start chomping out a clearing in the grass (frame 1). The caterpillars eventually meet at the clearing's center and disintegrate in a series of explosions (frame 2). Finally, a program title appears in the clearing (frame 3).

The clearing can handle a title of up to 33 characters. Adjust the statement 65524 PRINT@ value as needed to accommodate your program title.

You may also add upper and lower title lines with their centers at PRINT@ 351 and PRINT@ 479. Subtract one from the specified PRINT@ value for every two characters in an added title line. Add the upper and lower title lines by using statement numbers 65523 and 65525.

Using the Programs

Select a graphic title program that suits your existing program. Load, CLOAD, or @LOAD your program and List it to check its first and last statement numbers. If your program has a statement 0, renumber it. Also renumber any statement above 65499 that conflicts with a statement number in the selected title program.

When a statement ends with a Return command, trace backward through your program for the calling GOSUB command. Change the GOSUB command to correspond with the revised statement number.

After the renumbering is completed, key in the title program's statements. Put your program title and credit lines in the title and credit line statements. Adjust title PRINT@ values or title stack start and end numbers as prescribed in the title program's description.

Run the combined program to verify that the graphic title display routine works. At this time, readjust title PRINT@ values or title stack start and end numbers, if needed. Delete GOTO1 in statement 1.

When titling a program that

```
9999999999999999999999999999999999999999999999999999999999999999
9999999999999999999999999999        9999999999999999999999999999
999999999999999999999999                999999999999999999999999
99999999999999999999                        99999999999999999999
9999999999999999                                9999999999999999
999999999999                                        999999999999
99999999                                                99999999
9999                   PROGRAM TITLE HERE                   9999
99999999                                                99999999
999999999999                                        999999999999
9999999999999999                                9999999999999999
99999999999999999999                        99999999999999999999
999999999999999999999999                999999999999999999999999
9999999999999999999999999999        9999999999999999999999999999
9999999999999999999999999999999999999999999999999999999999999999
                     TITLE GRAPHICS  BY KAL
```

Fig. 6 10-Second Countdown

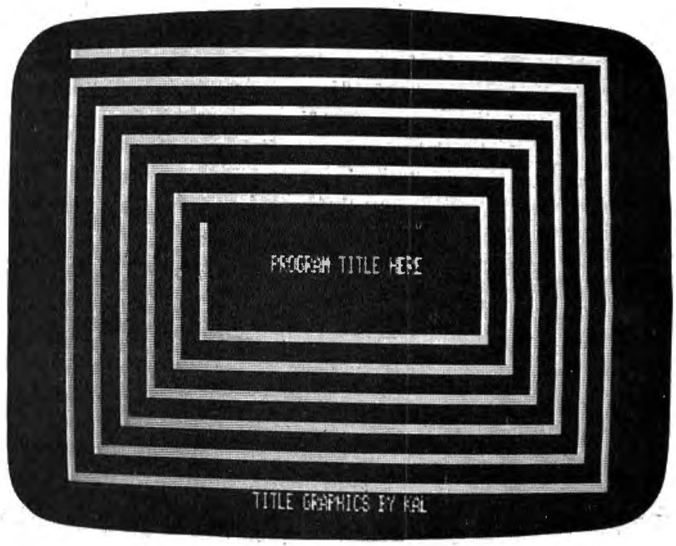


Photo 6. Program Listing 6 Spiral Title

```
0 CLS: CLEAR99: GOSUB65512: CLEAR50:     '(SEE PHOTO 5)
1 GOTO1:'(DELETE THIS STATEMENT WHEN TITLING A PROGRAM)
65512 Y=124: X=10: Z$=CHR$(149)
65514 '(IN 65512 ABOVE, SUBSTITUTE  ANY LETTER, NUMBER,
      SYMBOL, OR  GRAPHIC CHARACTER ASCII CODE  FOR THE
      (149) CODE.  SEE PAGES C/2 AND C/6  IN  RS TRS-80
      LEVEL II BASIC REFERENCE MANUAL.)
65516 PRINT@0,STRING$(68,Z$);
65518 PRINT@Y,STRING$(X,Z$);: Y=Y+62: X=X+4
65520 IF X<58 GOTO65518 ELSE PRINT@Y,STRING$(92,Z$);
65522 PRINT@217,"TITLE GRAPHICS";:  '(FIRST TITLE LINE)
65524 PRINT@348,"BY:  KAL";:     '(TITLE OR CREDIT LINE)
65526 FORU=1TO1999: NEXT: RETURN
```

Program Listing 5. Wedge Titles

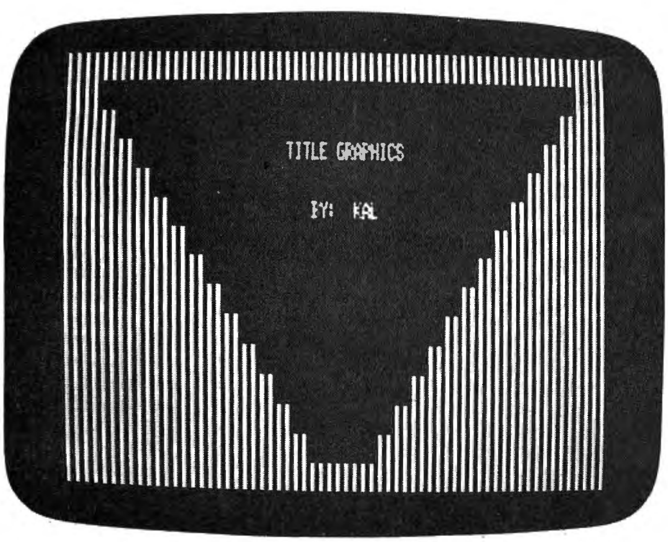
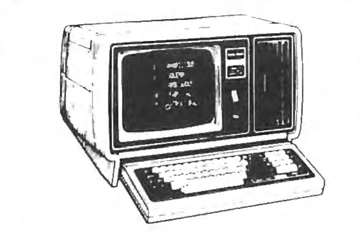


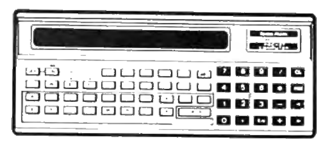
Photo 5. Program Listing 5 Wedge Title

Photo 7. Program Listing 7 Lattice Title

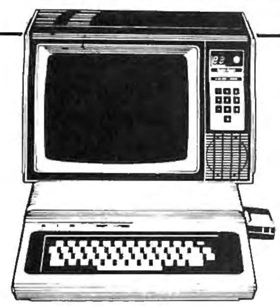
Here's a REAL PLUS



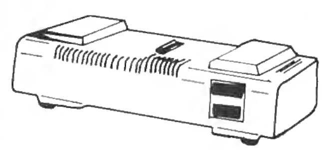
Model II 64K $3300



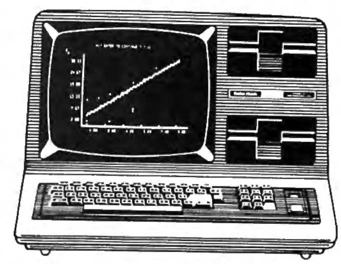
Pocket Computer $189



Color Computer 4K $310
w/16K Ext. Basic $489



32K Exp. Int. $334



Model III 16K $839

LOOK Here are just a few of our fine offers . . . call toll-free for full information.

COMPUTERS

Model II 64K	$3300
Model III 4K LEV I	599
MODEL III 16K	839
MODEL III 32K	945.50
*MODEL III 32K	881.50
MODEL III 48K	1052
*MODEL III 48K	924
Model III 48K 2 Disc & RS232 c	2100
Color Computer 4K	310
Color Computer 16K	416.50
*Color Computer 16K	352.50
Color Computer 16K w/extended basic	489
Pocket Computer	189
VIDEOTEX	310
APPLE 48K only	1279
ATARI 800 32K	789

*Computer Plus New Equipment.
180 Day Extended Warranty

PERIPHERALS

Expansion Interface OK	$249
Expansion Interface 16K	355.50
*Expansion Interface 16K	291.50
Expansion Interface 32K	462
*Expansion Interface 32K	334
16K RAM N.E.C. 200 N.S. chips	25

MODEMS

Lynx Direct Connect MI/MIII	269
Auto Ans./Dial Telephone Interface II	169
R.S. Modem I D.C.	130
Atari 830 Modem	159
Atari 850 Interface	183

PRINTERS

Daisy Wheel II	1695
Line Printer VI	999
Epson MX80	499
Epson MX80 FT	599
Epson MX100	799
Line Printer VII	315
Line Printer V	1610
Microline 80	365
Microline 82A	549
Microline 83A	839
Pocket Computer Printer	130

DISK DRIVES

R.S. Model III 1ST-Drive	712
TEAC 40 Track MI	329
R.S. 1 Drive Exp MII	999
R.S. 2 Drive Exp MII	1518
R.S. 3 Drive Exp MII	2040
Seimans 8" MII	799
Atari 810 Disk	499

SOFTWARE

R.S. Software 10% off list	
Atari Software 10% off list	
Newdos + (40) track MI	88
Newdos 80 MIII	149
ST80III	149

ETC.

Verbatum 5" Double Density	32
Verbatum 8" Data Life	49.95
Ctr-80A recorder	52
C. C Joysticks	22

PLUS real back-up warrantees —

Factory warrantees on Apple and Atari equipment. Other equipment carries manufacturer's warranty or Computer Plus 180 day extended warranty. Combined warrantees carry Computer Plus 180 day warranty or original manufacturer's warranty.

DEALER INQUIRIES ARE INVITED

Prices subject to change without notice.
Not responsible for typographical errors.

TRS-80 is a registered trademark of Tandy Corp.

call TOLL FREE **1-800-343-8124**

computer plus

Dept. D
245A Great Road
Littleton, MA 01460
617-486-3193

Write for your free catalog

✓130

✓See List of Advertisers on page 354

already displays a title, find the statement that prints it. Check the first few statements. Also check statements that are initially called by a GOTO or GOSUB. When found, delete the PRINT"TITLE" portion of the statement; leave all other statement segments intact. In some GOTO and GOSUB cases, the number of the title printing statement must be retained with a REM to ensure program *continuity*.

Now run the combined program. If everything runs smoothly, save, CSAVE or @SAVE your newly titled program on diskette, cassette or wafer. ■

```
0 CLS: CLEAR250: GOSUB65510: CLEAR50:    '(SEE PHOTO 7)
1 GOTO1:'(DELETE THIS STATEMENT WHEN TITLING A PROGRAM)
65510 DEFSTRA-F: F=STRING$(32,128): A=CHR$(191)+STRING$
      (2,131): B=A+A: C=B+B: D=C+C: E=D+D+C+A+CHR$(191)
65512 FORY=0TO832STEP64: PRINT@Y,E;: NEXT: PRINT@896,
      STRING$(64,131);
65514 FORY=592TO236STEP-64: PRINT@Y,F;: NEXT:
      PRINT@272,STRING$(32,131);
65516 PRINT@407,"PROGRAM TITLE HERE";
65518 PRINT@472,"SECOND LINE HERE";
65520 PRINT@981,"TITLE GRAPHICS  BY KAL";
65522 FORY=1TO1999: NEXT: RETURN
```

Program Listing 7. Lattice Title

```
0 CLS: CLEAR65: GOSUB65500: CLEAR50:      '(SEE FIG. 7)
1 GOTO1:'(DELETE THIS STATEMENT WHEN TITLING A PROGRAM)
65500 DEFSTRS-Z: V=STRING$(30,128): S=STRING$(30,33)
65502 W=" 'QQQQQQQQQQ@=": X=" ,0000000000@<": K=335
65504 Y="=@QQQQQQQQQQ' ": Z=">@0000000000, ": J=483
65506 FORR=0TO930STEP30: PRINT@R,S;: NEXT
65508 PRINT@J,Y;:PRINT@K,W;: FORR=1TO6:NEXT: PRINT@J,Z;
65510 PRINT@K,X;: FORR=1TO3:NEXT: K=K+1:J=J-1: IF K<404
      GOTO65520
65512 PRINT@414,"(**)";: FORR=1TO4: NEXT: PRINT@414,
      "*()*";: T="$++$": U="?+  +?
65514 PRINT@350,T;: PRINT@478,T;: FORR=1TO4: NEXT
65516 PRINT@285,U;: PRINT@541,U;: FORR=1TO4: NEXT
65518 FORR=273TO529STEP64: PRINT@R,V;: NEXT: GOTO65522
65520 FORR=241TO562STEP64: PRINT@R,S;: NEXT
65522 PRINT@J,Y;: PRINT@K,W;: IF K<410 GOTO65508
65524 PRINT@407,"CATERPILLAR TRAILS";
65526 PRINT@981,"TITLE GRAPHICS  BY KAL";
65528 FORR=1TO1999: NEXT: RETURN
```

Program Listing 8. Animated Title

```
0 CLS: CLEAR99: GOSUB65506: CLEAR50:     '(SEE PHOTO 6)
1 GOTO1:'(DELETE THIS STATEMENT WHEN TITLING A PROGRAM)
65506 W=60: S=63: A=1: T=959: U=896: R=62
65508 PRINT@0,STRING$(63,131);: GOTO65514
65510 R=64
65512 R=R+1: PRINT@R,STRING$(W,131);: W=W-6: R=R+66:
      A=A+1
65514 FOR X=S TO T STEP64: PRINT@X,CHR$(191);: NEXT:
      T=T-1: S=S+61
65516 FOR X=T TO U STEP-1: PRINT@X,CHR$(176);: NEXT:
      T=T-66
65518 FOR X=U TO R STEP-64: PRINT@X,CHR$(191);: NEXT
65520 U=U-61: IF A<2 GOTO65510 ELSE IF R<398 GOTO65512
65522 PRINT@471,"PROGRAM TITLE HERE";
65524 PRINT@981,"TITLE GRAPHICS  BY KAL";
65526 FORX=1TO1999: NEXT: RETURN
```

Program Listing 6. Spiral Title

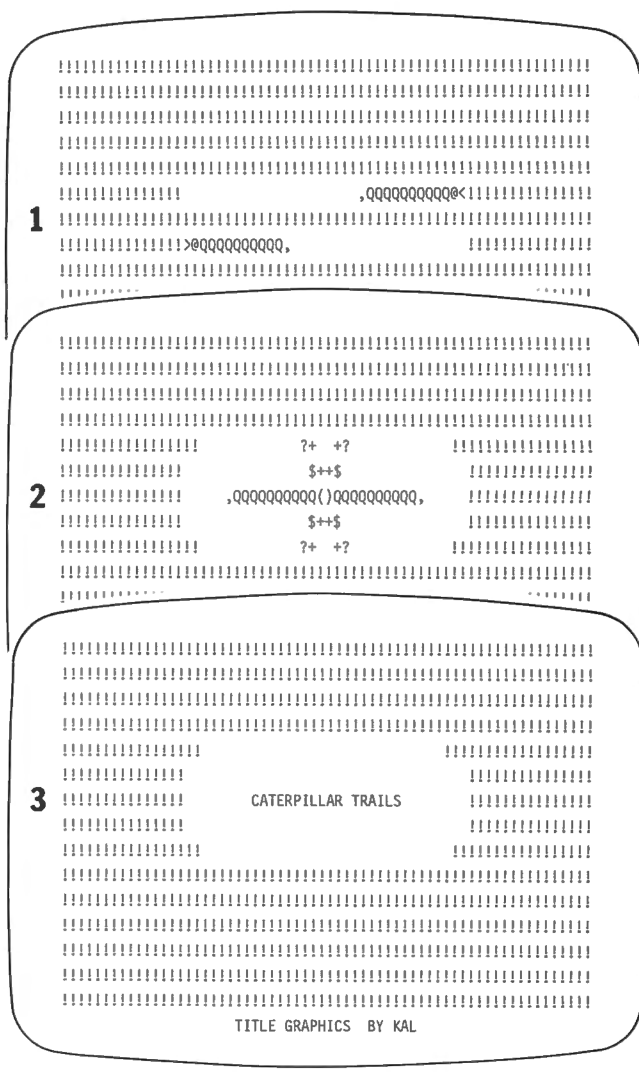


Fig. 7 Animated Title

✓See List of Advertisers on page 354

You'd be surprised what it really means.

Clear N

Nate Salsbury
608 Madam Moore's Lane
New Bern, NC 28560

One night at approximately 10:18, my 26-1006D got amnesia. The memory-tank gauge said OM. Sure enough, the dipstick read ?MEM 102! I should have bought the expansion interface and 32K more memory—my Radio Shack dealer had been right.

While investigating the problem, I came across the following interesting action of the Clear n statement. The instruction manual says that it (a) sets aside n bytes for string storage and (b) sets all variables to zero.

The Unmentioned (c)

Any intelligent person would conclude Clear does (a) and (b) and nothing else, but I discovered an unmentioned (c). Refer to Program Listing 1, the short program I developed which led me to (c). If you want to try this yourself, be very careful how you enter line 10. Note the following:

- There is no space between Clear and 100. This statement will be moved around and you will change your printout if you don't keep it compacted.
- There are *two* colons after Defint A–Z. There is no space between these colons.
- There is a trailing colon at the end. This, at first glance, appears unnecessary, but put it in.

Setting up in this fashion allows you to edit line 10 easily (we are going to reposition the Clear100 part) without changing the locations of any of the variables in memory. Line 120 is included to make this edit easy to do. The program shows how the placement of Clear 100 affects the representation of variables and/or arrays and how this, in turn, affects your free memory.

Line 10 implies that I want all variables to be integers. Clear100 appears as the first statement. Notice that X(0,0) is specified as 1.1 (a decimal). Line 20 can be any input, as long as it has a decimal part. I used 2.2 for this article. Line 40 shows how the computer handles the desired integer M, VARPTR(M) and what will be found when you PEEK there. Line 60 *does* look strange. In line 30, the variable A is equated to VARPTR(M). The instruction manual states that for all variables (except arrays), VARPTR( )-3 will contain a code (02 equals integer and 04 equals single precision) to indicate what type of variable follows. Therefore, the first B chosen by line 60 will display that code number for M.

The TRS-80 stores variables in the sequence they appear in the program and then stores arrays after that. In our program, M is the first variable specified followed by B and A. Therefore, the For. . . Next loop in lines 60–80 will print out all the pertinent data on the variables and end with the stored value for X(0,0). In the housekeeping information stored before VARPTR(X(0,0)), there are two bytes which store the total number of bytes used for the array—in this case it is specified, calculated and printed by lines 90–100. Taken all together, lines 60–100 are the spotlight which will uncover that (c) action.

Line 110 will cause a printout of the amount of free memory we have available and it also prints out the stored value for X(0,0). Finally, line 120 makes it easy to do our editing of line 10. Fig. 1

```
? 2.2
M = 2          ADDRESS = 17692          WHAT'S THERE? 2
 17689 2      17690 0       17691 77      17692 2
 17693 0      17694 2       17695 0       17696 66
 17697 33     17698 69      17699 2       17700 0
 17701 65     17702 28      17703 69      17704 2
 17705 0      17706 88      17707 247     17708 0
 17709 2      17710 11      17711 0       17712 11
 17713 0      17714 1       17715 0
NO. OF BYTES USED FOR ARRAY = 247
MEMORY LEFT = 14695
10 CLEAR100:DEFINT A-Z::X(0,0) = 1.1:
10
READY
>.
```

Fig. 1.

```
5 X$ = INKEY$: IF X$ = "" GOTO 5
6 CLS
10 CLEAR100:DEFINT A-Z::X(0,0) = 1.1:
20 INPUT M
30 B=0: A = VARPTR(M)
40 PRINT "M ="; M, "ADDRESS ="; A TAB(POS(0) + 5) "WHAT'S THERE?"; PEEK(A)
50 PRINT
60 FOR B = A-3 TO VARPTR(X(0,0)) + 1
70 PRINT B; PEEK(B),
80 NEXT B: PRINT
90 A = VARPTR(X(0,0)) - 7
100 PRINT "NO. OF BYTES USED FOR ARRAY ="; PEEK(A) + 256*PEEK(A+1)
110 PRINT "MEMORY LEFT ="; MEM, "X(0,0) ="; X(0,0)
120 EDIT 10
150 END
```

Program Listing 1

shows the result of running the program as shown in Program Listing 1. (Note that I listed line 10 at the end, in the Edit mode, so as to identify the placement of Clear 100 in Figs. 1, 2 and 3.)

Let's examine Fig. 1 carefully. The first line is my input of 2.2 for M. Next we see the TRS-80 has this stored as two. M is stored at 17692 and we find a two right there. Here's the ASCII relationship for our variables (A/65, B/66, M/77 and X/88).

In the tabular printout, 17692 does contain two (as stated above), 17691 has 77 (that's our M) and 17689 is the (integer) code 2. 17696 identifies B with the 66, 17701 does the same for A and then follows the information for array X. Some key locations here are 17706 (that is our ASCII X), 17704 (code 2 for integer), 17707/8 which contain the memory-used data for the array and the last two items which show the stored value of X(0,0). This time we find just the integer part of 1.1. Finally, Fig. 1 shows we have 14695 bytes of free memory left and notes the array has had 247 bytes set aside for future entries.

Now we're going to edit line 10. Delete Clear 100 from the beginning (leave the colon) and insert it between the two colons after Defint A-Z. When you now run this version, you should get the same address for M (see Fig. 2). If you get a different one from the one in Fig. 1, it means you have changed the number of bytes in your program by not keeping everything bunched together.

Fig. 2 looks like more of the same stuff, but now the computer says M is 2.2! It is still stored at 17692. 17691 is comforting—(there's our M/77) but look at 17689; it has a four—"M" is now single precision. And so is B (17696) and A (17703) and array X (17710). But we only have 14447 bytes of memory left. We have lost 248 bytes just by moving Clear 100.

(c) Revealed

We can see where 242 of them went because the array now has had 489 bytes set aside while before it only needed 247. The other lost bytes are taken up by our three variables which now each use four bytes (as single-precision numbers) instead of two (when they were stored as integers). We have discovered the unmentioned (c).

In addition to setting all variables to zero, Clear (anything) also causes all variables to be stored as single-precision values unless specified otherwise by a statement which follows the Clear. In my original program, I used four large arrays which I thought were integer types. I had set them up on the first program line with Defint and DIM statements and then, later, added Clear 120 when I found I needed some string storage. By merely placing that Clear 120 at the beginning of the line, I suddenly found about 1,100 lost bytes!

Okay—back to the laboratory. Edit line 10 again—take Clear100 out of the middle (leave both colons) and snuggle it up against the right-hand colon which I asked you to put there. Run again and get Fig. 3. The changes here are a little more subtle—same MEM, same number of bytes for the array, M is still 2.2. But note these points:

- Since Clear100 followed the definition of X(0,0), that value is now a 0 (as promised by the definition of Clear); but
- Even though there is nothing stored in the array, the computer still seems to have space reserved for it—look at the housekeeping info from 17710 to 17719.

```
? 2.2
M = 2.2            ADDRESS = 17692              WHAT'S THERE? 205
  17689 4          17690 0          17691 77         17692 205
  17693 204        17694 12         17695 130        17696 4
  17697 0          17698 66         17699 0          17700 72
  17701 10         17702 143        17703 4          17704 0
  17705 65         17706 0          17707 56         17708 10
  17709 143        17710 4          17711 0          17712 88
  17713 233        17714 1          17715 2          17716 11
  17717 0          17718 11         17719 0          17720 205
  17721 204
NO. OF BYTES USED FOR ARRAY = 489
MEMORY LEFT = 14447
10 :DEFINT A-Z:CLEAR100:X(0,0) = 1.1:
10 .
```

Fig. 2

```
5 X$ = INKEY$: IF X$ = "" GOTO 5 ELSE CLEAR 50
6 CLS
10 A = 1
20 PRINT "MEM", "VARPTR(B(0))", "VARPTR(A)"
30 PRINT MEM, VARPTR(B(0)), VARPTR(A)
40 PRINT MEM
50 CLEAR 50
60 PRINT MEM, VARPTR(B(0)), VARPTR(A)
150 END
```

Program Listing 2

Program Listing 2

This strange result led me to some further tests and I finally tied all the information together with Program Listing 2. Notice that the only variable specified here is 'A'. The printout for the run of this program is shown in Fig. 4. Let's analyze the results.

We print three numbers identified as MEM, VARPTR(B(0)) and VARPTR(A). We see that after assigning A its value of one we have MEM = 15200. We find an address for B(0) even though we haven't specified B in any way. A is stored at 17497. On the next line, by itself, we have 15148 bytes of MEM (we appear to have lost 52 bytes somewhere). Then the program restates what Basic does automatically if not told anything else, it Clears 50. The final line shows: MEM = 15207; B(0) has a new address (7 bytes earlier than before); and A doesn't exist any more (the blank space under VARPTR(A) plus the "?FC error in 60" line both tell us this). Here's what this data reveals:

- Unless otherwise specified, the computer dimensions single-variable arrays for 10 (0–10 or 11 elements).
- We didn't specify INT so the computer stores numerical data in single-precision format.
- Therefore, a single-variable, undefined array needs four bytes/elements by 11 elements equals 44 bytes, plus eight housekeeping bytes equals 52 bytes.
- By merely asking for an array address, even it if has never been specified, the computer sets aside space for that array. That's where the missing 52 bytes went.
- When we executed Clear 50 the array and variable A were wiped out. Since A was stored in single-precision format, it required seven bytes of memory. We had 15200 bytes of memory with A stored and, therefore, 15207 when it was erased.
- Finally, as soon as we asked for VARPTR(B(0)), the computer reestablished a location for it (but now it was set up seven bytes lower because A didn't exist any more). This is proved by the blank under VARPTR(A) plus the FC error message.

Conclusion

For maximum memory conservation, always make your Clear statement first. Once you have assigned a variable any value, the computer keeps its assigned space forever (with the exception of a new Clear statement). You cannot free those bytes of memory for any other purpose. The instruction manual points out how many variables you have available to play with (around 900). Even in integer form, each variable uses five bytes. So be very conservative in your use of variable names—especially in the Command mode. Any new variables you use are stored also and continue to take up space—even if they are used just once. Be very careful in Command mode when requesting VARPTR for arrays. If you enter Print VARPTR(AN(0,0)) by mistake when you really meant AB(0,0), you have just irrevocably lost at least 247 bytes (or 489 if A has not been declared an integer).■

```
? 2.2
M = 2.2          ADDRESS = 17692                  WHAT'S THERE? 205
  17689 4        17690 0          17691 77        17692 205
  17693 0        17698 66         17699 0         17700 72
  17701 10       17702 143        17703 4         17704 0
  17705 65       17706 0          17707 56        17708 10
  17709 143      17710 4          17711 0         17712 88
  17713 233      17714 1          17715 2         17716 11
  17717 0        17718 11         17719 0         17720 0
  17721 0
NO. OF BYTES USED FOR ARRAY = 489
MEMORY LEFT = 14447
10 :DEFINT A-Z::X(0,0) = 1.1:CLEAR100
10
```

Fig. 3.

```
RUN
MEM         VARPTR(B(0))   VARPTR(A)
  15200        17509          17497
  15148
  15207        17502
?FC Error in 60
READY
>
```

Fig. 4.

MULTI-USER OASIS HAS THE FEATURES PROS DEMAND. READ WHY.

(THEN COMPARE.)

NOW AVAILABLE FOR TRS80 MOD II

Computer experts (the pros) usually have big computer experience. That's why when they shop system software for Z80 micros, they look for the big system features they're used to. And that's why they like Multi-User OASIS. You will too.

DATA INTEGRITY: FILE & AUTOMATIC RECORD LOCKING

The biggest challenge for any multi-user system is co-ordinating requests from several users to change the same record at the same time.

Without proper co-ordination, the confusion and problems of inaccurate or even destroyed data can be staggering.

Our File and Automatic Record Locking features solve these problems.

For example: normally all users can view a particular record at the same time. But, if that record is being updated by one user, automatic record locking will deny all other users access to the record until the up-date is completed. So records are always accurate, up-to-date and integrity is assured.

Pros demand file & automatic record locking. OASIS has it.

SYSTEM SECURITY: LOGON, PASSWORD & USER ACCOUNTING

Controlling who gets on your system and what they do once they're on it is the essence of system security.

Without this control, unauthorized users could access your programs and data and do what they like. A frightening prospect isn't it?

And multi-users can multiply the problem.

But with the Logon, Password and Privilege Level features of Multi-User OASIS, a system manager can specify for each user which programs and files may be accessed— and for what purpose.

Security is further enhanced by User Accounting—a feature that lets you keep a history of which user has been logged on, when and for how long.

Pros insist on these security features. OASIS has them.

EFFICIENCY: RE-ENTRANT BASIC

A multi-user system is often not even practical on computers limited to 64K memory.

OASIS Re-entrant BASIC makes it practical.

How?

Because all users use a single run-time BASIC module, to execute their compiled programs, less memory is needed. Even if you have more than 64K, your pay-off is cost saving and more efficient use of all the memory you have available—because it services more users.

Sound like a pro feature? It is. And OASIS has it.

AND LOTS MORE...

Multi-User OASIS supports as many as 16 terminals and can run in as little as 56K memory. Or, with bank switching, as much as 784K.

Multi-Tasking lets each user run more than one job at the same time.

And there's our BASIC— a compiler, interpreter and debugger all in one. An OASIS exclusive.

Still more: Editor; Hard & Floppy Disk Support; Keyed (ISAM), Direct & Sequential Files; Mail-Box; Scheduler; Spooler; all from OASIS.

Our documentation is recognized as some of the best, most extensive, in the industry. And, of course, there's plenty of application software.

Put it all together and it's easy to see why the real pros like OASIS. Join them. Send your order today.

OASIS

MAKES MICROS RUN LIKE MINIS

OASIS IS AVAILABLE FOR

SYSTEMS: Altos: Compucorp: Cromemco; Delta Products; Digital Group: Digital Microsystems; Dynabyte: Godbout. IBC; Index; Intersystems; North Star; Onyx; SD Systems; TRS 80 Mod II; Vector Graphic. Vorimex.

CONTROLLERS: Bell Controls; Cameo; Corvus. Konan; Micromation; Micropolis; Tarbell; Teletek; Thinkertoys; X-Comp.

Write for complete, free Application Software Directory.

PLEASE SEND ME:

Product	Price with Manual	Manual Only
OPERATING SYSTEM (Includes: EXEC Language; File Management; User Accounting; Device Drivers; Print Spooler; General Text Editor; etc.)		
SINGLE-USER	$150	$17.50
MULTI-USER	350	17.50
BASIC COMPILER/ INTERPRETER/DEBUGGER	100	15.00
RE-ENTRANT BASIC COMPILER/INTERPRETER/ DEBUGGER	150	15.00
DEVELOPMENT PACKAGE (Macro Assembler; Linkage Editor; Debugger)	150	25.00
TEXT EDITOR & SCRIPT PROCESSOR	150	15.00
DIAGNOSTIC & CONVERSION UTILITIES (Memory Test; Assembly Language; Converters, File Recovery; Disk Test; File Copy from other OS; etc.)	100	15.00
COMMUNICATIONS PACKAGE (Terminal Emulator; File Send & Receive)	100	15.00
PACKAGE PRICE (All of Above)		
SINGLE-USER	500	60.00
MULTI-USER	850	60.00
FILE SORT	100	15.00
COBOL-ANSI '74	750	35.00

Order OASIS from:

Phase One Systems, Inc.
7700 Edgewater Drive, Suite 830
Oakland, CA 94621
Telephone (415) 562-8085
TWX 910-366-7139

NAME ____________
STREET (NO BOX #) ____________
CITY ____________
STATE ________ ZIP ________
AMOUNT $ ________
(Attach system description; add $3 for shipping; California residents add sales tax)
☐ Check enclosed ☐ VISA
☐ UPS C.O.D. ☐ Mastercharge
Card Number ____________
Expiration Date ____________
Signature ____________

Apparat users, you don't have to copy an entire system of data disks just because you've changed its name.

Rename

David D. Busch
515 E Highland Avenue
Ravenna, OH 44266

Rename is a utility allowing NEWDOS+ and NEWDOS-80 users to change the name of any System or data disk without having to recopy the disk.

Why would you need to rename a disk? Perhaps in reorganizing your files, you happen to move all the programs from a disk you have named Disk2 onto other disks. Rather than keep an empty disk in your file, it might be tidier to take the highest numbered disk and rename it Disk2. There are many other reasons for changing or updating a disk name and for correcting or altering the date on a disk.

Making the change doesn't really require a special utility. Rename just streamlines the task actually carried out using Superzap.

TRSDOS, VTOS, NEWDOS 2.1 and NEWDOS 80 all store the disk name and date in the first sector of the directory track; this is track 17 (11 hex). The disk name is stored in eight bytes, relative bytes D0H to D7H in track 17. The date immediately follows at relative bytes D8H to DFH. To change this information, it is necessary only to replace the values stored in those bytes with the hex equivalents of the ASCII codes for the characters you wish to substitute.

Rename does all the work for you. The user enters the substitute strings, and the program calculates the hex codes. Complete instructions for making the fix using Superzap are printed to the screen, so even a user who has never used Superzap can rename or redate disks with no trouble.

An Example

If someone wished to name a disk Testdisk and change the date to 01/05/81, he or she would input those strings when prompted by the program, in lines 50 and 70, respectively. No complex error traps are used. It is assumed that the user has some basic familiarity with the naming requirements of disk. The program does check that the entered name is eight characters or less. If the input date does not equal eight characters exactly, the user is asked to re-input the string, using MM/DD/YY format. In fact, any eight characters may be input for the date, but the two slashes must be in their proper positions on the disk to avoid later problems.

The new name and date are examined one character at a time in a routine located in lines 100–180. A For. . . Next loop of 1 to 8 assigns the character of each position in turn of NME$ (line 110) to A$. Control then branches to a subroutine in lines 410–560.

Here, the ASCII code (decimal) for the character is determined (line 410), and that figure converted to hex. To do this, the code (B) is divided by 16 and the result, minus the remainder, is assigned to D1$. In hexadecimal notation, the first position to the left of the decimal point (the one's place) shows the number of digits from 1 to 16 (from 1 to 9, then A,B,C,D,E,F in hex). The second position to the left of the decimal point (the ten's place in decimal notation) shows the number of 16's. D1$ is a string representation of the number of 16's in the ASCII code. Because Rename accepts only uppercase disk names with each character having a decimal equivalent of less than 9F hex, no provision has to be made in this program for converting numbers higher than that (A0, etc.).

The remainder produced when B is divided by 16 is multiplied by 16 to find the number in the ones column. If this number is higher than nine, it is converted to the equivalent hex figure at lines 470–540. That is, 10 becomes A, 11 becomes B, etc.

This second digit's string representation (D2$) is then added to D1$ to produce the final hex equivalent of the decimal ASCII code. Control returns to the parsing subroutine.

If the new name chosen for the disk is less than eight characters long, at some point MID $(NME$,N,1) will equal " " (null), and 20H (32 decimal), which is a space, is added to NME$(n), which stores the finished hex conversion of the name.

During the same For. . . Next loop, DATES$ is also parsed, converted to ASCII in hex and stored in DATES$(n).

When the conversion is complete, the program prints out the following instructions, in this case using the string provided in our Testdisk example:

Carry out the following steps:
1) Load Superzap
2) Hit 'Enter'
3) Load disk to change in drive
4) Enter (drive spec),17,0 to prompts
5) Enter 'MOD D0'
6) Enter the characters shown below
7) Hit Enter,'Y",and Enter again
8) Exit Superzap

54 45 53 54 44 49 53 4B
30 31 2F 30 35 2F 38 31
Hit 'JKL' to send these instructions to printer

Note: the program runs as-is on NEWDOS 80. Those using NEWDOS 2.1 with the Basic version of Superzap, which requires hex entries, change line 240 to read:

240 PRINT" 4) Enter (drive spec),II,0 TO PROMPTS"

In both cases, the user should substitute the drive number in which the target disk is contained (drive specification) in the above examples. Because Rename was written for users of Superzap,I assumed that the JKL function would obviate the need to have a special printing subroutine. Those using other zapping programs with DOS that don't have a JKL equivalent will have to make necessary changes on their own. Because the I6 bytes of hex numbers are difficult to remember, the user will usually want to send the instructions to a printer, if available. Those without a printer will have to copy the figures on a piece of paper. ■

```
 1 '**************************************************
   *                                                *
   *             DISK NAME/DATE CHANGER             *
   *                                                *
 2 '*            ·BY: DAVID D. BUSCH                *
   *                 515 E. HIGHLAND AVE.           *
   *                 RAVENNA, OHIO 44266            *
 3 '*                                               *
   **************************************************

 10 CLEAR 500
 20 CLS:PRINT
 30 PRINT "THIS PROGRAM CAN BE USED TO CHANGE THE NAME AND DATE OF "
 40 PRINT "A DISK, USING SUPERZAP AS AN AID."
 45 ' ****** ENTER NEW NAME AND DATE ******
 50 INPUT "PLEASE ENTER THE NEW NAME FOR THE DISK";NME$
 60 IF LEN(NME$)>8 THEN PRINT "SORRY, ONLY EIGHT CHARACTERS ALLOWED":GOTO 50
 70 INPUT "PLEASE ENTER THE NEW DATE FOR THE DISK (USE MM/YY/DD)";DATE$
 80 IF LEN(DATE$)>8 THEN PRINT"SORRY, ONLY EIGHT CHARACTERS ALLOWED":GOTO 70
 90 IF LEN(DATE$)<8 THEN PRINT "USE MM/DD/YY FORMAT":GOTO 70
 95 ' ****** PARSE EACH CHARACTER OF NAME/DATE STRINGS ******
100 FOR N=1 TO 8
110 A$=MID$(NME$,N,1)
120 IF A$="" THEN NME$(N)=" 20 ":GOTO 150
130 GOSUB 410
140 NME$(N)=HX$
150 A$=MID$(DATE$,N,1)
160 GOSUB 410
170 DATE$(N)=HX$
180 NEXT N
185 ' ****** PROVIDE CHANGING INSTRUCTIONS ******
190 CLS:PRINT:PRINT
200 PRINT "CARRY OUT THE FOLLOWING STEPS:"
210 PRINT "   1.) LOAD SUPERZAP"
220 PRINT "   2.) HIT 'ENTER'"
230 PRINT "   3.) LOAD DISK TO CHANGE IN DRIVE"
240 PRINT "   4.) ENTER <DRIVE SPEC>,17,0 TO PROMPTS"
250 PRINT "   5.) ENTER 'MOD D0'"
260 PRINT "   6.) ENTER THE CHARACTERS SHOWN BELOW."
270 PRINT "   7.) HIT ENTER, 'Y', AND ENTER AGAIN"
280 PRINT "   8.) EXIT SUPERZAP"
290 PRINT
300 FOR N=1 TO 8
310 PRINT NME$(N);
320 NEXT N
330 PRINT
340 FOR N=1 TO 8
350 PRINT DATE$(N);
360 NEXT N
370 PRINT
380 PRINT "HIT 'JKL' TO SEND THESE INSTRUCTIONS TO PRINTER"
390 GOTO 390
400 ' ****** CONVERT TO HEX ******
410 B=ASC(A$)
420 H1=B/16
430 H2=H1-INT(H1)
440 H3=H2*16
450 D1$=STR$(INT(H1))
460 IF H3<10 THEN D2$=MID$(STR$(H3),2,1):GOTO 550
470 ON H3-8 GUIU 480,490,500,510,520,530,540
480 D2$="0":GOTO 550
490 D2$="A":GOTO 550
500 D2$="B":GUIU 550
510 D2$="C":GOTO 550
520 D2$="D":GOTO 550
530 D2$="E":GOTO 550
540 D2$="F":GOTO 550
550 HX$=D1$+D2$
560 RETURN
```

Program Listing. Rename

✓See List of Advertisers on page 354

A great program for Saturday mornings.

80 Cartoon Capers

Darren DeVigili
439 South River St.
Wilkes-Barre, PA 18702

The TRS-80 cartoon is here. This program uses 2.7K bytes, and can be adapted by even the novice programmer. I've included several subroutines to save time and memory while adapting the program.

Disk sound (almost) prints WHRRRR... and goes to a delay subroutine. Clear desk clears the graphic desk (it has a bar for graphics across it), and clear screen clears the 80's screen.

The graphics are generated using the CHR$, STRING$ and POKE functions using the (X) and (Z) variables. The variable (X) saves memory in assigning character codes, but is also used in all loops. (Z) is used in a double loop.

Before any lengthy adapting, I suggest writing down the line numbers and functions of the subroutines to save time. ■

```
10 REM * TRS-80 CARTOON *
20 GOSUB730:REM * LOOP THROUGH AND RUN ANY ADDITIONS *
30 REM *** START ADDING HERE ***
700 REM * WHITE SCREEN *
710 FORX=276TO483STEP64:PRINT@X,STRING$(16,191);:NEXTX:FORX=578T
O637:PRINT@X,CHR$(191);:NEXTX:GOSUB920:FORX=15360TO16383:POKEX,1
91:NEXTX
720 REM * TRS-80 GRAPHICS *
730 CLS:PRINT@210,CHR$(131);:PRINT@211,CHR$(163);:PRINT@512,STRI
NG$(21,176);:PRINT@212,STRING$(16,179);:PRINT@228,CHR$(144);:PRI
NT@531,CHR$(178);:PRINT@532,STRING$(16,179);:PRINT@228,CHR$(147)
;:PRINT@229,STRING$(5,131);:PRINT@548,CHR$(177);
740 X=170:PRINT@549,CHR$(176);:PRINT@550,CHR$(177);:PRINT@551,CH
R$(178);:PRINT@552,STRING$(2,176);:PRINT@209,CHR$(X);:PRINT@273,
CHR$(X);:PRINT@337,CHR$(X);:PRINT@401,CHR$(X);:PRINT@465,CHR$(X)
;:PRINT@529,CHR$(X);
750 PRINT@275,CHR$(X);:PRINT@339,CHR$(X);:PRINT@403,CHR$(X);:PRI
NT@467,CHR$(X);:X=149:PRINT@292,CHR$(X);:PRINT@356,CHR$(X);:PRIN
T@420,CHR$(X);:PRINT@484,CHR$(X);
760 PRINT@234,CHR$(X);:PRINT@298,CHR$(X);:PRINT@362,CHR$(X);:PRI
NT@426,CHR$(X);:PRINT@490,CHR$(X);:PRINT@554,CHR$(X);:PRINT@294,
CHR$(156);:PRINT@295,CHR$(140);:PRINT@296,CHR$(172);:PRINT@358,C
HR$(141);:PRINT@359,CHR$(140);:PRINT@360,CHR$(142);
770 REM * DISK GRAPHICS *
780 PRINT@555,CHR$(181);:PRINT@491,CHR$(X);:PRINT@427,CHR$(X);:P
RINT@562,CHR$(X);:PRINT@498,CHR$(X);:PRINT@434,CHR$(X);
790 PRINT@364,STRING$(6,140);:PRINT@556,STRING$(6,176);:PRINT@36
3,CHR$(156);:PRINT@370,CHR$(148);:PRINT@494,CHR$(171);:PRINT@430
,CHR$(168);
800 PRINT@563,CHR$(181);:PRINT@499,CHR$(X);:PRINT@435,CHR$(X);:P
RINT@570,CHR$(X);:PRINT@506,CHR$(X);:PRINT@442,CHR$(X);
810 PRINT@372,STRING$(6,140);:PRINT@564,STRING$(6,176);:PRINT@37
1,CHR$(156);:PRINT@378,CHR$(148);:PRINT@502,CHR$(171);:PRINT@438
,CHR$(168);:PRINT@571,STRING$(5,176);
820 REM * DESK GRAPHICS *
830 PRINT@576,STRING$(2,191);:PRINT@638,STRING$(2,191);:FORX=160
00TO16383:POKEX,191:NEXT:X=219:PRINT@721,CHR$(X);:PRINT@785,CHR$
(X);:PRINT@849,CHR$(X);:PRINT@913,CHR$(X);:PRINT@977,CHR$(X);:GO
SUB930:GOSUB920:GOSUB960
840 REM * PRINT ROUTINES *
850 PRINT@277,STRING$(14,42);:PRINT@340,"***PRESENTING***";:PRIN
T@469,STRING$(14,42);:PRINT@404,"*****TRS-80*****";
860 GOSUB930:PRINT@404,"****GRAPHICS****";:PRINT@340,"****COMPUT
ER****";:GOSUB930:GOSUB920:GOSUB950
870 PRINT@277,"RADIO SHACK";:PRINT@341,"LEVEL II BASIC";:PRINT@4
05,"READY";:PRINT@469,">";CHR$(95);:GOSUB930:PRINT@470,"RUN";
880 REM * BOUNCING DOT GRAPHICS *
890 FORX=578TO635STEP4:PRINT@X,CHR$(131);:FORZ=1TO25:NEXTZ:PRINT
@X,CHR$(193);:PRINT@X+1,CHR$(140);:FORZ=1TO25:NEXTZ:PRINT@X+1,CH
R$(193);:PRINT@X+2,CHR$(176);
900 FORZ=1TO25:NEXTZ:PRINT@X+2,CHR$(193);:PRINT@X+3,CHR$(140);:F
ORZ=1TO25:NEXTZ:PRINT@X+3,CHR$(193);:NEXTX:GOTO30:REM * RELOOP *
910 REM * SUBROUTINES * DISK * DELAY * CLS DESK * CLS * PLUG *
920 PRINT@303,"WHRRRRR...";:GOSUB930:PRINT@303,CHR$(202);:FORX=1
TO50:NEXTX:RETURN
930 FORX=1TO1500:NEXTX:RETURN
940 FORX=578TO637:PRINT@X,CHR$(193);:NEXTX:RETURN
950 FORX=276TO483STEP64:PRINT@X,CHR$(208);:NEXTX:RETURN
960 PRINT@579,"- TRS-80 IS A REGISTERED TRADEMARK OF TANDY CORPO
RATION. -";:GOSUB930:GOTO940
970 REM * TRS-80 CARTOON BY DARREN DEVIGILI 439 S RIVER ST *
          * WILKES-BARRE PA 18702 PHONE (717) 823-9128 *
```

Program Listing

DISK DRIVE SALE

MDX-2 System Expansion for TRS 80's

The new MDX-2 system expansion board from COMPUTEX provides more capabilities than any other system expansion currently available.

FEATURES INCLUDE

- Capability to build only what you need.
- Silk screened and solder masked P.C.B.
- Dip switches are used instead of hardware jumpers.
- No messy vertically mounted termination resistors.

• On board direct connect modem kit	$39.95
• 2K/4K Eprom kit	$17.95
• RS232/20m.A. serial interface	$17.95
• Real time clock kit	$17.95
• Floppy disk controller kit	$31.95
• 32K RAM expansion kit (less Ram)	$11.95
• Cassette port	$3.95
• Analog power supply	$29.95
• MDX-2 printed circuit board	$74.95
• Hardware and socket kit	$19.95
• Centronics line printer port	$8.95

Buy just the system expansion board and those kits you want, or save even more and buy the complete kit.

MDX-2 KIT COMPLETE
$269.00

TANDON TM100-1

Bare Drive $235.00 — **With case/supply $285.00**
Single sided 40 track • 102KB Single Density, 180KB Double Density

TANDON TM100-2

Bare Drive $335.00 — **With case/supply $385.00**
Double sided 40 track • 204KB Single Density, 360KB Double Density

TANDON TM100-3

Bare Drive $335.00 — **With case/supply $385**
Single sided 80 track • 204KB Single Density, 360KB Double Density

TANDON TM100-4

Bare Drive $455 — **With case/supply $505.00**
Double sided 80 track • 408KB Single Density, 735KB Double Density

*All TANDON drives from **COMPUTEX** come with comprehensive users manual, TANDON maintenance manual and schematics. In addition, all drives are electrically tested before, during and after their 48 hour burn in.*

DISK DRIVE ACCESSORIES
Two Drive Cable $24.95
Four Drive Cable $34.95
Drive Extender Cable $14.95
Model III Second Drive Cable $4.95

Model III Disk Drive Kits & Drives

Now available from **COMPUTEX**, a Radio Shack Disk Drive Kit for your Model III for **only $429.95.**
Compare these standard features on our list to the others.

1. Drive Controller Board
2. All necessary interconnecting cables
3. Switching Power Supply
4. Disk Drive Mounting Brackets
5. Internal Disk Drive Cable
6. Comprehensive installation and theory manual
7. 16K RAM

COMPLETE MODEL III DRIVE PACKAGES
These packages all include, the Model III Disk Drive Kit and the Micro Systems Software Model III DOS.

M3DK-1	(2-TM100-1 Drives) provides over 350K Bytes storage	919.95
M3DK-2	(2-TM100-2 double headed drives) provides over 700K Bytes storage	1,149.95
M3DK-3	(2-TM100-3 80 track drives) provides over 700K Bytes storage	1,199.95
M3DK-4	(2-TM100-4 double headed 80 track drives) provides over 1.4 M Bytes storage	1,399.95

COMPUTEX Comprehensive Test Program

As one of the leading suppliers of disk drives and TRS80 systems, we had to have a systems test program that we knew worked. Since no complete systems test program already existed, we wrote our own. The *COMPUTEX COMPREHENSIVE TEST* (**CCT**) program was written exclusively for **COMPUTEX** and is now available to everyone.

Features

DISK DRIVE SPEED TEST • DISK DRIVE ALIGNMENT PROGRAM*
EXPANSION INTERFACE RAM TEST • KEYBOARD RAM TEST • VIDEO RAM TEST
KEYBOARD BOUNCE TEST • ROM CHECKSUM PROGRAM
FULL SCREEN RAM DISPLAY PROGRAM
*The Drive Alignment Program requires an Alignment Disk.

The CCT program is a must for disk drive users and should be a part of everyones library.

CCT Program $21.95
Dysan Alignment Disk $39.95

MEMORY/SPECIAL FUNCTION

FD1771B-01	23 95	FD1793B-01	44 95
TR1602	3 95	AY5 1013A	4 95
MC1372	6 48	MC6674	13 00
2716-450	12 95	2114-450	5 95
Z80A	14 95	2102	1 75
CC74S188	Programmed for LNW Color CPU		9 95
CC2716-6	Level II Roms for Color CPU		99 95

LINEAR

LM1488	1 20	TL084	1 75
LM1489	1 20	MC1458	58
UA7805	1 10	MLM311P1	90
UA7812	1 10	MC14412	17 00
UA7912	1 10	78H05	6 00
79L12	1 50	75452	35

DIODES TRANSISTORS / SEMICONDUCTORS

1N914/1N4148	05	KBP02	79
1N4001	09	2N3904	25
1N5231	25	2N3906	25
1N5244	25	MPU131	89
1N5244	25	J175	89

CRYSTALS

1 0 MHZ	4 00	3 579 MHZ	3 95
4 0 MHZ	4 00	16 0 MHZ	4 95

1% PRECISION RESISTORS

267 OHMS	324 OHMS	619 OHMS
845 OHMS	1 24 K OHMS	2 55 K OHMS
8 66 K OHMS	8 87 K OHMS	18 7 K OHMS
21 5 K OHMS	95 3 K OHMS	97 6 K OHMS
165 K OHMS	210 K OHMS	237 K OHMS

All 1% Resistors are $ 18 each

I.C. SOCKETS

8 PIN Low Profile	12	20 PIN Low Profile	23
14 PIN Low Profile	14	22 PIN Low Profile	27
16 PIN Low Profile	16	24 PIN Low Profile	30
18 PIN Low Profile	.21	40 PIN Low Profile	47

7400 SERIES TTL

7438	20	74157	60
7442	45	74161	83
7451	56	74164	.83
7474	32	74165	83
7486	32	74166	1 15
7490	42	74173	1 15
7492	42	74175	.75
7493	42	74176	75
7495	58	74194	83
74121	32	74195	64
74123	45	74151	.54
74125	45	74366	64
74132	41	74367	66

74LS00 SERIES - LOW POWER SCHOTTKY

74LS00	27	74LS139	75
74LS02	27	74LS153	75
74LS04	29	74LS155	90
74LS05	39	74LS157	.75
74LS08	27	74LS161	1.10
74LS09	32	74LS164	1 18
74LS10	27	74LS166	1.85
74LS11	71	74LS174	1.69
74LS13	56	4LS175	.95
74LS14	1 15	74LS193	1.10
74LS15	31	74LS240	1.85
74LS20	27	74LS241	1.85
74LS21	31	74LS244	1.85
74LS30	26	74LS245	2 25
74LS32	31	74LS257	85
74LS74	44	74LS367	69
74LS86	41	74LS373	1.89
74LS93	70	74LS374	1 85
74LS123	1 18	74LS393	2 35
74LS124	1.69	74LS138	86
74LS132	95		

74S00 SCHOTTKY

74S04	55	74S05	47
74S22	47	74S32	47
74S64	47	74S74	61
74S112	71	74S161	1 75
74S174	1 59	74S175	1.50
74S280	2 45	74S387	5.95

LNW COLOR COMPUTER KIT
$499.95
(less P.C.B.)
Includes all components to assemble the LNW color computer: high resolution video RAM and 16K user RAM, keyboard, and Level II compatible ROMS. Printed circuit board not included.

LNW EXPANSION INTERFACE KIT
$219.95
(less P.C.B.)
Includes all components to assemble the LNW Expansion Interface plus 32K of RAM. It does not include the printed circuit board, dual cassette relay or cabinet.

•

A beautiful custom all wood cabinet is available for housing the LNW Expansion Interface. **$99.95**

ONE YEAR GUARANTEE
on our 200 N.S. 16K Memory Kits
16K - $23.95 — 32K - $45.95 — 48K - $74.95
These are top quality, new memory chips at unbelievably low prices.

(713) 488-8022

COMPUTEX
ELECTRONICS COMPUTERS ✓392
321 El Dorado Blvd. • Webster, Tx. 77598
Visa and Mastercard accepted
Personal and company checks require two weeks to clear

*Please visit our new showroom just off the Gulf Freeway. We have expanded, changed our name back to **COMPUTEX**, added more personnel in order to offer you complete service and reliability.*

✓See List of Advertisers on page 354

Let your programs perform surgery on themselves.

Slice & Dice Basic

J. Stewart Schneider
Rt. 6 Box 244
Ashland, KY 41101

Did you ever wish you could make a Basic program alter itself? Assembly language programmers do it all the time, and just because you're a sensible sort who programs in Basic doesn't mean you have to take a back seat to the hexadecimal set.

Radio Shack's Level II Basic is a wondrous critter. You can learn it in no time, it's relatively fast, and it'll give you your answers to 17 decimal places. Unfortunately, the machine holding this wonderful language is hooked up to the damnedest, most frustrating piece of machinery known to man—that blasted tape recorder. As if that weren't enough, those of us without the necessary $500 are stuck with 16K.

So why develop a technique for making Basic lines alter themselves? Accuracy and memory is why. You need accuracy because if data lines could alter themselves, the CLOAD function could be used to verify that the tape machine had recorded things properly. Memory is important because altering a line to do more than one thing means a reduction in the number of lines in a program, and that's good.

Before we find out how to do this, we'll have to check in briefly with the assembly language group to better understand what we're talking about. A given hunk of code, while executing, can make decisions about what later code should be. Because both assembly data and assembly code are just numbers located at specific addresses, an assembly program can easily elect to treat program code as if it were data and alter it to meet changed circumstances. This is called self-modifying code.

We'll use a simple example. Let's imagine a line of assembly code in a larger program. At the line in which we are interested, the program must compare two numbers to find which is larger. We know that if the first number is larger, then a later line must read, "Subtract 13 from the accumulator". If the second number is larger, the same line must later say, "Add 37 to the accumulator".

We could, of course, write two routines, one to subtract 13 and one to add 37, and branch to the correct one. However, this particular example wouldn't illustrate my point.

Assembly codes reside in definite addresses in memory, like pigeons in a bunch of pigeon holes. This is our program, so we know what address the add or subtract instruction will be found in and also where its data will be. So instead of writing two routines, we can jam the proper instruction and the proper data byte into the proper pigeon holes, and have one routine doing the work of two.

Self-Modifying Basic

We can pull the same thing off in Basic! If we knew where the pigeon holes were that Basic uses, and what should be in them, we could jam other things in, using POKE, just like we did above, and the Basic line would modify itself. What we need to know is "where?" and "what?". Both questions are answered in the way a Basic line is stored.

Let's take a sample line, the first and only line, in a program: 10 INPUT A. This line is stored in memory starting at address 17129. (This is stated in the Level II handbook.) What is stored at 17129, however, is not what we see on the screen, but the low byte of a pointer to the beginning of the next line. 17130 contains the high byte. I know this seems silly, but it's how a Z-80 chip thinks.

The next two addresses, 17131 and 17132, contain the low and high bytes of the line number 10 respectively. Next comes the program text, but in code. Basic instructions are stored in memory in the form of one-byte tokens which Basic expands into words for a list. Humans don't understand tokens and computers don't understand words. Basic understands both.

The next byte, therefore, contains the token for input, which is listed in the Level II handbook as 137, followed by a 32 (space) and a 65 (ASCII "A"). The line ends with a zero to signal end-of-line. End-of-program is signaled by two zeros in place of the pointers in the next line, so all programs end with three zeros.

Making it Work

We now know to make one routine do the work of two. I had written a long program in Basic which dealt in large arrays. So large, in fact, that reading them from cassette or writing them to cassette, one element at a time, took nearly an hour. If I was going to make the program useful at all, I would have to cut tape time radically. To do it, I wrote the tape routines so they would tape 14 elements at a pass, which resulted in a very long tape routine (see Program Listing 1). My machine cheerfully advised me that I was out of memory.

Glaring balefully at my program for places to pare, I realized that my two tape routines were identical except that one said Print#-1 and the other Input#-1.

"Hmmm," I thought..."if I knew where that print token was in memory, maybe I could POKE the token for input and make the one routine function as both an input and an output routine. But would mucking about with a command token destroy my arrays in the same way Editing does?" (It does, you know, which is most inconvenient.)

First, I wrote Program Listing 2, which I added to the end of my program. Without dimensioning my arrays, there was plenty of room. Listing 2 follows the pointers from line to line, calculating the line numbers and comparing them with the target line number. If the target line number is found to begin at address N, then N and N+1 would be the pointer for the next line. N+2 and N+3 would be the line number. Address N+4, the first address in the text portion of the line, would contain the token for print. Running Listing 2 for each of the lines containing Print#-1 in Listing 1 gave me my addresses. What remained was to POKE the input token into these addresses and see if my arrays would come through this harrowing process unscathed. The result is in Program Listing 3.

When Listing 3 is run, the first order of business is to load the tape files into the proper arrays, so line 70 POKEs the input token into the addresses found by Listing 2, altering lines 110, 140 and 170 to input lines. If the program were Listed at this point, these lines should read Input#-1 instead of Print#-1. After the tape files have been read, line 180 POKEs 178, the print token,

✓See List of Advertisers on page 354

```
10 '   DOCKET CONTROL VERS. 4.0X 16K LEVEL II BASIC 4/28/80
20  REM * COPYRIGHT 1980 BY J. STEWART SCHNEIDER *
30 CLEAR1500:Z8=3:Z9=38:DIM D(200,2):DIM AT$(Z9):DEFINTP,C,X:DIM
 N$(200):DIM C(200,2):FORX=0TOZ9:READAT$(X):NEXT:REM ** Z9 = # O
F ATTORNEYS +1 **
40 U1$="##.####":U2$="##.###":HD$="IND. NO.    JUDGE/NO.    ARRA
IGNED    B/P DUE    TRIAL DATE      ============================
================================"
50 J1$="(ASB.)":J2$="(SIN.)":REM ** J1$ & J2$ ARE THE JUDGES' NA
MES
60 '                TAPE READ ROUTINE
70 TAPE%=1:CLS:I$="READING: ":INPUT"REWIND DATA TAPE, PRESS PLAY
 ONLY, THEN TAP <ENTER> KEY";A$:INPUT#-1,F:GOTO90
90 TAPE%=TAPE%-1:CLS:PRINT@405,"THERE ARE";F;"CASES IN THE FILE"

100 FORX=1TOFSTEP7:PRINT@469,I$;X;"TO";X+6
110 INPUT#-1,D(X,1),D(X,2),D(X+1,1),D(X+1,2),D(X+2,1),D(X+2,2),D
(X+3,1),D(X+3,2),D(X+4,1),D(X+4,2),D(X+5,1),D(X+5,2),D(X+6,1),D(
X+6,2)
120 NEXTX
130 FORX=1TOFSTEP7:PRINT@469,I$;X;"TO";X+6
140 INPUT#-1,C(X,1),C(X,2),C(X+1,1),C(X+1,2),C(X+2,1),C(X+2,2),C
(X+3,1),C(X+3,2),C(X+4,1),C(X+4,2),C(X+5,1),C(X+5,2),C(X+6,1),C(
X+6,2)
150 NEXTX
160 FORX=1TOFSTEP7:PRINT@469,I$;X;"TO";X+6
170 INPUT#-1,N$(X),N$(X+1),N$(X+2),N$(X+3),N$(X+4),N$(X+5),N$(X+
6)
180 NEXTX:GOTO500
185 '               TAPE RECORD ROUTINE
190 INPUT"REWIND DATA TAPE, ERASE, PRESS PLAY & RECORD THEN TAP
<ENTER>";A$:PRINT#-1,F
200 FORX=1TOFSTEP7:PRINT@469,I$;X;"TO";X+6
210 PRINT#-1,D(X,1),D(X,2),D(X+1,1),D(X+1,2),D(X+2,1),D(X+2,2),D
(X+3,1),D(X+3,2),D(X+4,1),D(X+4,2),D(X+5,1),D(X+5,2),D(X+6,1),D(
X+6,2)
220 NEXTX
230 FORX=1TOFSTEP7:PRINT@469,I$;X;"TO";X+6
240 PRINT#-1,C(X,1),C(X,2),C(X+1,1),C(X+1,2),C(X+2,1),C(X+2,2),C
(X+3,1),C(X+3,2),C(X+4,1),C(X+4,2),C(X+5,1),C(X+5,2),C(X+6,1),C(
X+6,2)
250 NEXTX
260 FORX=1TOFSTEP7:PRINT@469,I$;X;"TO";X+6
270 PRINT#-1,N$(X),N$(X+1),N$(X+2),N$(X+3),N$(X+4),N$(X+5),N$(X+
6)
280 NEXTX
```

Program Listing 1

into these same addresses, changing the Input#-1 back to a Print#-1 so that the updated files can be recorded at the end of the run.

To my astonishment, it worked. The lines altered themselves docilely and my arrays went untouched. Listing 3 does the work of Listing 1, but with only one tape routine. The out-of-memory error went away to plague someone else.

Data Statements

So what about data statements, you ask? As places to store information, data statements are gems. They load at the full 500 baud and they can be verified by a CLOAD. They do not, however, lend themselves to easy alteration.

Since it is a part of the program, a data line must be altered through the editing function. This involves stopping the program, entering the Edit mode, deleting the old data, and inserting the new. If the data is frequently updated this rapidly becomes a source of potential error. And, if the program is to be run by persons with little training in programming, chaos will probably result.

But if the program could be induced to find the appropriate data statement, the updated data could be POKEd to replace the old, and the program re-saved and verified. We only need to know how information is stored in a data statement.

Data statements are no different from other lines in the layout of pointers and line numbers. Some judicious PEEKing about in a sample data line shows that numerical data is stored as ASCII codes instead of the actual value, presumably to save space.

For whatever reason it is done that way, ASCII representation means that we can mimic a data line with the string functions. If there is room enough in the data line, this string of data can be simply POKEd into it following the data token of one character at a time by using the MID$ function of Level II BASIC.

For those not familiar with it, MID$(L$,X,Y) returns a string of length Y, beginning with the character in position X of L$. In Listing 4, we use MID$(P$,X,1) which returns the character at position X in P$.

Our approach is to first verify that there is enough room in the data line to contain the new information. You must leave enough room by spacing past the least data element when wiring the line. POKEing new data into the final zero and the following line pointers will result in untold unpleasantness. If enough room is available, we will clear all addresses following the data token except that final zero by POKEing an ASCII blank (32) into each address. This removes the old data. Next, we convert our new data from numerical form to strings using the STR$ function. The string representations of our numbers will then be con-

```
10000 INPUT"LINE NUMBER SOUGHT";R:N=PEEK(17129)+PEEK(17130)*256
10010 N1=PEEK(N+3)*256+PEEK(N+2)
10020 IF NOT N1=R THEN N=PEEK(N+1)*256+PEEK(N):GOTO10010
10030 PRINT"TOKEN FOUND AT";N+4;"CODED";PEEK(N+4):GOTO10000
```

Program Listing 2

catenated into P$, with each item being followed by a comma.

At this point, if we printed P$ we would see all our new data, looking for all the world like a data statement without the word data or a line number. We would also see an unnecessary trailing comma and leading blanks before all positive numbers. Lines 10050 through 10090 remove the comma at the end and the leading blanks that STR$ inserted in place of a sign. It's not necessary to do this, but it appeals to my aesthetic sensibilities.

Our new string, P1$, is then scanned character by character and the ASCII value of each character is POKEd into successive addresses in the data line.

The main body of Listing 4, lines 10 through 100, reads 10 numbers into array A, then doubles them. The original numbers, from data line 60, represent the original data. The doubled numbers represent the updated data. All the action, of course, is in the subroutine beginning at line 10000.

Running Listing 4 changes data line 60 so that it winds up containing the doubled data. A final prompt instructs the user to load a blank cassette, and the program CSAVEs itself under file name X.

That's it, neighbors, the BASIC Zap; a way to make BASIC remedy itself.■

```
10 '   DOCKET CONTROL VERS. 4.ØX 16K LEVEL II BASIC 4/28/8Ø
2Ø  REM * COPYRIGHT 198Ø BY J. STEWART SCHNEIDER *
3Ø CLEAR15ØØ:Z8=3:Z9=38:DIM D(2ØØ,2):DIM AT$(Z9):DEFINTP,C,X:DIM
 N$(2ØØ):DIM C(2ØØ,2):FORX=ØTOZ9:READAT$(X):NEXT:REM ** Z9 = # O
F ATTORNEYS +1 **
4Ø U1$="##.####":U2$="##.###":HD$="IND. NO.    JUDGE/NO.    ARRA
IGNED    B/P DUE    TRIAL DATE     ==============================
==============================="
5Ø J1$="(ASB.)":J2$="(SIN.)":REM ** J1$ & J2$ ARE THE JUDGES' NA
MES
6Ø '               TAPE I/O DRIVER
7Ø TAPE%=1:CLS:I$="READING: ":INPUT"REWIND DATA TAPE, PRESS PLAY
 ONLY, THEN TAP <ENTER> KEY";A$:POKE17935,137:POKE181Ø6,137:POKE
18277,137:INPUT#-1,F:GOTO9Ø
8Ø INPUT"REWIND DATA TAPE, ERASE, PRESS PLAY & RECORD THEN TAP <
ENTER>";A$:PRINT#-1,F
9Ø TAPE%=TAPE%-1:CLS:PRINT@4Ø5,"THERE ARE";F;"CASES IN THE FILE"

1ØØ FORX=1TOFSTEP7:PRINT@469,I$;X;"TO";X+6
11Ø PRINT#-1,D(X,1),D(X,2),D(X+1,1),D(X+1,2),D(X+2,1),D(X+2,2),D
(X+3,1),D(X+3,2),D(X+4,1),D(X+4,2),D(X+5,1),D(X+5,2),D(X+6,1),D(
X+6,2)
12Ø NEXTX
13Ø FORX=1TOFSTEP7:PRINT@469,I$;X;"TO";X+6
14Ø PRINT#-1,C(X,1),C(X,2),C(X+1,1),C(X+1,2),C(X+2,1),C(X+2,2),C
(X+3,1),C(X+3,2),C(X+4,1),C(X+4,2),C(X+5,1),C(X+5,2),C(X+6,1),C(
X+6,2)
15Ø NEXTX
16Ø FORX=1TOFSTEP7:PRINT@469,I$;X;"TO";X+6
17Ø PRINT#-1,N$(X),N$(X+1),N$(X+2),N$(X+3),N$(X+4),N$(X+5),N$(X+
6)
18Ø NEXTX:IF PEEK(17935)=137 THEN POKE17935,178:POKE181Ø6,178:PO
KE18277,178:I$="WRITING: "
```

Program Listing 3

```
1Ø CLS:CLEAR175
2Ø DIM A(1Ø)
3Ø FOR X=1TO1Ø
4Ø READ A(X)
5Ø NEXT X
6Ø DATA 128,256,384,512,64Ø,768,896,1Ø24,1152,128Ø

7Ø FORX=1TO1Ø
8Ø A(X)=A(X)*2
9Ø NEXT
1ØØ GOSUB1ØØØØ
11Ø PRINT"DATA NOW UPDATED"
12Ø LIST6Ø
13Ø END
1ØØØØ N=PEEK(1713Ø)*256+PEEK(17129)
1ØØ1Ø LN=PEEK(N+3)*256+PEEK(N+2):IF LN>6Ø THEN STOP
1ØØ2Ø IF LN<>6Ø THEN N=PEEK(N+1)*256+PEEK(N):GOTO1ØØ1Ø
1ØØ3Ø E=PEEK(N)+256*PEEK(N+1)-2:AV=E-N-5
1ØØ4Ø P$="":FORX=1TO1Ø:P$=P$+STR$(A(X))+",":NEXT
1ØØ5Ø L=LEN(P$)-1
1ØØ6Ø FORX=1TOL
1ØØ7Ø IF ASC(MID$(P$,X,1))=32 THEN 1ØØ9Ø
1ØØ8Ø P1$=P1$+MID$(P$,X,1)
1ØØ9Ø NEXTX:L=LEN(P1$):IF L>AVTHEN PRINT"TOO MUCH DATA":STOP
1Ø1ØØ FORX=N+5TOE:POKEX,32:NEXT
1Ø11Ø FORX=1TOL
1Ø12Ø Q=ASC(MID$(P1$,X,1))
1Ø13Ø POKEX+N+5,Q
1Ø14Ø NEXT:RETURN
```

Program Listing 4

Hey all you heads, it's time to clean up your act!

Head Bright

Joseph Hesse
353 West Church St.
River Falls, WI 54022

Have you been getting more disk input/output errors recently? The problem may be due to a dirty read/write head in one of your disk drives. No matter how good your drives and diskettes are, if the heads are dirty the signals to and from the computer will be affected.

One way to clean the heads is to partially disassemble the drive to expose the read/write head. This should only be done by a qualified technician.

Fortunately, there is a better way. Use a disk drive head cleaning diskette. It looks exactly like an ordinary diskette except it has a cleaning surface that rotates in place of the magnetic disk. Moisten the cleaning surface with the solution that is supplied with the head cleaning diskette and insert it in the drive. Turn on the drive for about half a minute or so and that's it—the head is clean.

Head cleaning diskettes are available in computer stores and are advertised in computer magazines. I use the 3M company's (the Scotch tape people) diskette. It is easy to apply the cleaning solution on those and there are printed boxes on the diskette that can be checked off after each use—up to 15.

Suppose you have the 3M head cleaning diskette or a similar product. You moisten the cleaning surface and insert the diskette so that the exposed surface faces the red indicator light and is towards the rear of the drive. Now, we need a program to get the disk drive motor to turn on and the head to load against the cleaning surface.

The Head Cleaning Program

Carefully key in the following Basic program. Run the program; it will ask which drive you want to use. At this point you should take the diskettes out of every drive that you plan to clean. Now enter the appropriate drive number (0,1,2,3). The program will prompt you to insert the moistened cleaning diskette and (enter).

To use the cleaning surface most effectively, the program starts the head at the outermost track and moves it toward the center. While the head is being cleaned, the program tells you what track the head is over. After the head is cleaned, you have the option of starting over so that another head may be cleaned.

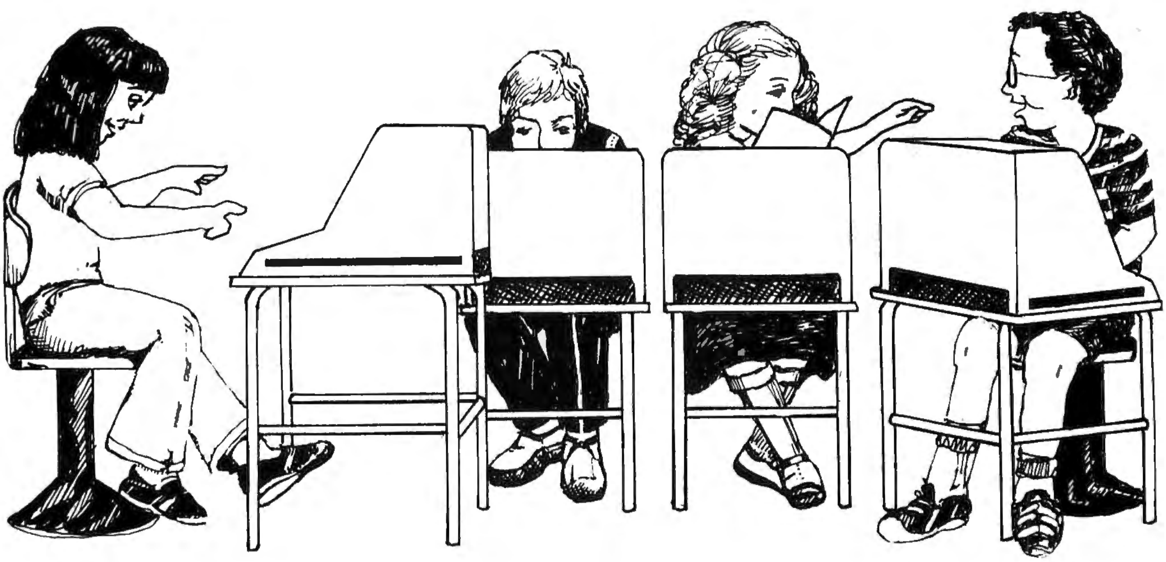
The program listing is for a TRS-80 with two drives. If you have three or four drives change the value of N in line 130 to 3 or 4 and change the prompting message in line 160. If you have only one drive, delete lines 160 through 190 and change line 130 to read 130 N = 2 : T = 0 : D = 1. ■

```
100 REM DISK HEAD CLEANING PROGRAM FOR TRS-80 MODEL I.
110 REM TO BE USED WITH 3M HEAD CLEANING DISKETTE OR SIMILAR PRO
DUCT.
120 DEFINT A-Z
130 N=2 : T=0
140 CLS : PRINT CHR$(23);
150 PRINT@136,"** DISK HEAD CLEAN **" : PRINT
160 LINE INPUT"WHICH DRIVE? (0,1)   ";A$ : PRINT
170 IF LEN(A$)=0 OR LEN(A$)>1 THEN GOTO 150
180 D=ASC(A$)-48 : IF D<0 OR D>N-1 THEN GOTO 150
190 D=INT(2[D + 0.5) : REM [=UP ARROW
200 PRINT"INSERT CLEANING DISKETTE, THEN" : PRINT" PRESS ENTER."

210 A$=INKEY$ : IF A$<>CHR$(13) THEN GOTO 210
220 POKE 14304,D
230 POKE 14316,3 : GOSUB 340
240 PRINT@576,"HEAD OVER TRACK";PEEK(14317)
250 FOR J=1 TO 90 : POKE 14304,D : NEXT J
260 IF T=34 THEN GOTO 290
270 POKE 14316,83 : GOSUB 340
280 T=T+1 : GOTO 240
290 IF (PEEK(14316) AND 128)<>128 THEN GOTO 290
300 PRINT@704,"REPEAT PROGRAM? (Y OR N)"
310 A$=INKEY$ : IF A$="" THEN GOTO 310
320 IF A$="Y" THEN GOTO 130
330 IF A$="N" THEN END ELSE GOTO 310
340 IF (PEEK(14316) AND 1)=1 THEN GOTO 340  ELSE RETURN
```

Program Listing 1

This program is like a game.

Analogies



Damon L. Spencer
33 Steppingstone Road
Dover, NH 03820

I'm a high school student at Oyster River in Durham, NH. At our school, we have a small computer room containing a single Level II 16K TRS-80, and a dual terminal 8K PDP-8/E mini for beginners.

Lisa Ogden, a senior, developed an educational, hands-on Basic computer literacy course, and ran it using select students from the fourth grade of an elementary school.

ANALO-80 is a program I created to aid in the education of these students.

ANALO-80 is a Basic program that combines the speed of a computer with the logistics of analogy quizzes. ANALO-80 allows a teacher (or anyone else), to create a module containing a set of 20 analogies and present them to a student, friend, etc. The program randomly selects an analogy from

Program Listing

```
10 CLS:RESTORE
15 DIM B2(20)
20 FOR V3=15360TO16383
30 POKEV3,191
40 NEXTV3
50 FORA1=1TO126
60 RESET(A1,1):RESET(A1,46)
70 NEXTA1
80 FORA2=1TO46
90 RESET(2,A2):RESET(3,A2):RESET(124,A2):RESET(125,A2)
100 NEXTA2
110 PRINT@347,"ANALOGY #1";
120 FORA3=1TO500:NEXTA3
130 PRINT@409,"CUSTOM MADE FOR";
140 FORA3=1TO500:NEXTA3
150 PRINT@471,"MASTWAY ELEMENTARY";;
160 FORA3=1TO500:NEXTA3
170 PRINT@536,"BY DAMON SPENCER";
```

Program continues

the module, records its code number, and presents the analogy to the user. The user selects what he or she thinks is the correct answer (from a group of four displayed), and inputs the corresponding letter. The computer checks the inputted answer against its record of proper answers, then places the following display at the bottom of the video monitor: Right = 1 Wrong = 0.

If the inputted answer is correct, "Right" will flash seven times. If the answer is incorrect, "Wrong" will flash.

As analogies are presented, the code number for that analogy is recorded. As each analogy is brought out of the module, its code is checked to the list of previously picked analogies. This subroutine assures that no analogy is used twice, and that the sequence in which the analogies are displayed is random.

After all analogies have been presented, the computer clears the screen and draws a bar graph which shows the user's score against the best possible score.

Some unique features which could be added to ANALO-80 are:

- a subroutine to allow the creation, saving and loading of individual modules without reloading the entire base. This subroutine would involve the use of the 'Print # -1' and 'Input # -1' commands.
- the For. . . Next loops which regulate the length of the modules can be changed to allow for longer or shorter modules. This can be especially useful if used in conjunction with the above feature, whereas the length of the module being inputted is given as the first piece of data from the load off the cassette.
- The user can be given more than one chance to guess the correct answer if the first answer was wrong.
- scores may be saved on a data cassette so that a teacher may plot the students' advancement.

```
DATA #17
PROBLEM #5

WING IS TO BIRD AS FIN IS TO . . .
  A) SHIP
  B) ROBIN
  C) WATER
  D) FISH
ANSWER?

RIGHT = 3                  WRONG = 1
```

Fig. 1.

Fig. 1 shows the video monitor display during an analogy. Fig. 2 shows how to organize a module and individual analogies so the computer can read them. This is a bare listing without any of the above features added. I leave the challenge to you.

If you discover another feature or have any ideas about this program, I would appreciate a listing or a taped copy so that I may keep up with new versions of ANALO-80. I would also appreciate other educational programs to use in our computer literacy program. ■

```
200 DATA "WING IS TO BIRD AS FIN IS TO...","A) SHIP","B) ROBIN","C)
WATER","D) FISH","D"
```

Explanation: Data line 200 contains one analogy of the module (in this case, analogy 5—see sample 1). The first piece of data is the unfinished analogy. The second piece of data contains choice A, etc. The last piece of data (in this case 'D') is the correct answer to the analogy. The answer to the analogy on line 200 would be FISH (D).
Only one analogy, complete with choices and answer should be on each program line. This allows you to quickly correct a mistake without searching through piles of data. If, while running ANALO-80, you see an error in the presentation or answer, quickly look at the screen and jot down the Data Number. Break out of the program, list it, and when you come to the data section, simply count down the appropriate number of lines to the analogy.

Fig. 2

Program continued

```
180 FORA3=1TO500:NEXTA3
190 PRINT@606,"ORHS";
200 FORA3=1TO1200:NEXTA3
210 CLS
220 PRINT@465,"SEE TEACHER FOR INSTRUCTIONS"
230 FORA4=1TO1100:NEXTA4
240 CLS
250 PRINT"HELLO !! WHAT IS YOUR NAME ";:INPUTA$
260 CLS:PRINT"THANK YOU ";A$;" !! LET'S START.....":FORX=1TO800:
NEXT
270 CLS
275 FOR B=1 TO 20
277 RESTORE
280 IF B=20 THEN 289
285 GOTO 305
289 X=0
290 FOR B1=1 TO 20
295 IF B2(B1)<>B1 THEN C1=B1
300 NEXT B1:GOTO 330
305 LET C1=INT(RND(19)+1)
306 IF C1>20 OR C1<1 THEN 305
310 IF B2(C1)>0 THEN 305
320 B2(C1)=C1
330 FOR K=0 TO (C1-1)
335 READ D1$,D2$,D3$,D4$,D5$,D6$
340 NEXT K
365 READ E1$,E2$,E3$,E4$,E5$,E6$
390 PRINT@0,"PROBLEM #";B
400 PRINT@64,"DATA #";C1
403  J7=64-LEN(E1$)
404  J8=FIX(J7/2)
405 J9=J8+191
410 PRINT@J9,E1$
420 PRINT@330,E2$

430 PRINT@394,E3$
440 PRINT@458,E4$
450 PRINT@522,E5$
460 INPUT"YOUR ANSWER ";E7$
470 IF E7$=E6$ THEN M=M+1 ELSE N=N+1
471 GOSUB 640
475 E7$=""
480 CLS:NEXTB
490 CLS:PRINT@470,"AND NOW YOUR SCORE..."
500 FORX=1TO1100:NEXTX
510 CLS
520 PRINT@20,A$;"'S GRAPHIC RESULTS"
530 PRINT@840,"RIGHT=";M:PRINT@860,"WRONG=";N
550 PRINT@434,"B":PRINT@498,"E":PRINT@562,"S"
560 PRINT@626,"T"
570 FORX=0TO102:SET(X,12):SET(X,35):NEXTX
580 FORY=13TO17:SET(101,Y):SET(102,Y):NEXTY
590 FORY=34TO30STEP-1:SET(101,Y):SET(102,Y):NEXTY
610 FORX=0TO((M*5)-2)
620 SET(X,21):SET(X,22):SET(X,23)
630 NEXT X
633 FOR T=1 TO 2000:NEXT T
635 GOTO 900

640 GOSUB 650
641 IF E7$=E6$ THEN GOSUB643 ELSE GOSUB646
642 RETURN
643 FOR V9=1 TO  9:PRINT@840,"      "
644  FOR V6=1 TO 040:NEXTV6:PRINT@840,"RIGHT"
645 FOR V6=1 TO 040:NEXTV6:NEXT V9:RETURN
646 FORV9=1 TO 9:PRINT@860,"      "
647 FORV6=1TO040:NEXTV6:PRINT@860,"WRONG"
648 FORV6=1 TO 040:NEXTV6:NEXTV9:RETURN
650 PRINT@840,"RIGHT=";M:PRINT@860,"WRONG=";N:RETURN
660 PRINT@867,"<ENTER> WHEN READY:";
670 IFINKEY$=""THEN670
680 CLS:GOTO2000
690 DATA "","","","","",""
700   DATA "FLY IS TO SPIDER AS MOUSE IS TO","A) RODENT",
         "B) CAT","C) RAT","D) ANIMAL","B"
710   DATA "LOUD IS TO THUNDER AS LARGE IS TO","A) MONKEY",
         "B) MIDGET","C) WHALE","D) COAL","C"
720   DATA "PETAL IS TO FLOWER AS FUR IS TO","A) COAT",
         "B) RABBIT","C) WARM","D) WOMAN","B"
730   DATA "WEEK IS TO DAY AS DAY IS TO","A) MONTH",
         "B) SECOND","C) HOUR","D) NIGHT","C"
740   DATA "AIR IS TO SUFFOCATION AS FOOD IS TO","A) STARVATION"
         "B) INDEGESTION","C) ENERGY","D) LIFE","A"
```

Program continues

TEXAS COMPUTER SYSTEMS

Offers Lowest Prices on

Color Computer

4K Level I $329
16K Level I $439
16K Extended Basic $489

With TCS Memory:
16K Level I $369
16K Extended Basic $449

Expansion Interfaces

Zero K Interface $254
16K Interface $359
32K Interface $469

With TCS Memory:
16K Interface $318
32K Interface $388

Pocket Computer & Interface $ Call toll free

We have all Radio Shack computers and accessories, and a number of other brand computer items. Call us for your specific needs and get a copy of our catalog.

Epson MX-80 $ Call

Letter quality matrix similar to Line Printer IV & Centronics printer but has full software control of 40, 80, 66 or 132 columns. 80 cps bidirectional tractor feed, disposable printhead. $300 less than nearest competitive printer. Lists $645. Our price includes cable

SOFTWARE

MACPLUS $49.95

An Editor & Macro Assembler for Model III

We supply patches for the Macro Assembler to make it fully operative on the Model III. Our new editor includes virtually all the features of Radio Shack's SCRIPSIT plus these features: direct entry of graphics; auto tab for entering Assembly language instructions; imbedding of commands for justification, centering, margin control, change print types within text, etc. Also included on the disk are: **FORM/CMD** - Prints formated copies of your document to your specifications on your smart or dumb printer.

MODIFY/CMD — A utility which allows assembly language program modification from patches defined in an ASCII patch file.

MAC/BAS — A Basic program containing several utilities to aid in the preparation of source files.

Those described above and seven other complete programs make this an excellent value. Please specify Model I or III system.

Note: You must have, or purchase the Model I Microsoft Macro Assembler Package (Cat. #26-2202) from Radio Shack. If you don't already have this item, we are offering it at a special sale price in combination with the software package described above. Call for information and our low sale price.

ZAP III $19.95

A super zap for the Model III giving full sector and track access to any part of the TRSDOS diskettes. Jumps by tracks or sectors throughout the disk. Automatically goes to directory sectors of any disk with just one keystroke. Instant comparing of the same sectors on different disks. Full menus, lists, and descriptions of commands are available at any point. Just press clear and all commands and their descriptions are listed. Copy disk sectors, modify sectors, toggle between hex and ascii input of characters as you modify.

This disk also includes **COPYIT/CMD**, a utility program allowing the purchaser to make more than two back-up copies of SCRIPTSIT or VISICALC for private use. Good insurance for your valuable programs.

Special feature: the documentation includes a complete map of the location of all System Files on the Model III TRSDOS diskettes. Various patches also included to get rid of errors in TRSDOS 1.1 and 1.2 such as the mysterious EOF set to 0 on coping of files. This is a must purchase at $19.95.

MACPLUS-2 $39.95

A Macro Assembler Utility Kit.

Five programs for one low price:
RUN/CMD — A macro command processor.
SCREEN/CMD — Save video screen images to disk.
RELOCATE/CMD — Allows dynamic relocation of program.
PICMAC/CMD - Creates a screen of information that can be converted to an assembly language module and then used to print the screen.
PICBAS/CMD - Does the same but converts to a BASIC Program Module. Creates screens, menus, prompts, graphics, fast. It's like getting SCRIPTSIT for half price with other items and patches thrown in free.

Specify Model I or III.

EPSON MX-80 PRINTER DISKETTE

A complete diskette with 6 complete files for use with your Epson MX-80 Printer. . . ready to use.

M/CMD - Send all of the Epson Printer commands to the printer directly from DOS or as part of an auto booting system. Set up the printer from DOS to the typestyle you want; then run the program.

BIGLETT/BAS - Produce large graphic letters on your printer and screen. Dump the letters to printer or to disk. Allows 3 different size letters to the printer. Program is menu driven. You may also construct large letter titles for MERGEing into your basic program.

EPSON/SUB - Merge this small module into any Basic program and then issue any of the printer commands with a 2 letter command with letters representing the purpose of the command. You need not refer back and forth to documentation. Extremely easy to use and makes using the printer much easier than ever before.

JKL PATCH - This patches NEWDOS 2.1 and NEWDOS 80 so that the JKL function will print the TRS80 graphics while the printer remains in the standard mode rather than the TRS80 mode. This allows all of the features of the printer to be used but you can dump the screen to the printer at any time with all graphics.

DEMO/BAS - A super tutorial program with hundreds of REMs and hundreds of screen prompts that teach you how easy it is to create miracles with your Epson MX80 printer.

ARTICLE - A complete article about the Epson MX80 with all sorts of tips, concepts, etc. for getting the most out of your printer. The information for creating scheduling forms is a must for many of you

DEFSTATE/SUB - A series of one word printer commands that can be merged into any Basic Programs that allow various automatic functions to occur. Print titles with graphic borders around them with one word. Center any text with one command, etc.

Special Price: All seven of the above modules on one diskette or on tape (less JKL) for your Model I or III for only **$29.95**.

Or, take a $10 discount if you buy this item with a purchase of the Epson MX-80.

The DOS Disk Library of Programs

A library of 5 programs for NEWDOS80, NEWDOS 2.1, TRSDOS (Model I & III)

DIRMAINT—A super directory-maintenance program. Purge, copy, rename, re-format, etc., quickly with minimum typing. Super graphic displays for easy use

FILEMAINT—Creates master directory of all files on your library of diskettes. Master file is handled on disk, not on memory, for maximum speed. Access any file in 2 seconds. 48K supports 200 diskettes. 32K about 100. Takes less than hour to run 100 diskettes thru system

CHAINBLD/BAS—A super chaining file creation editor & assembler for NEWDOS80. Very fast, easy to use. Reads and edits existing files

HOMEMAIL—Unique home mailing list that stores data by modifying program itself. Fast & efficient for home user with less than 100 names

GASMILE—Maintains excellent records of auto gas use & complete reports on MPG, etc. Also provides car maintenance alerts

Model I or III Disk **$39.95** for all 5 programs

Printers

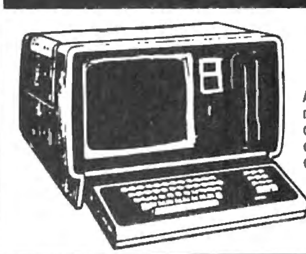
Daisy Wheel II $1695

Line Printer IV $825
Line Printer V $649
Line Printer VI $1015
Line Printer VII $350

TEXAS COMPUTER SYSTEMS

Offers Lowest Prices on

TRS-80 COMPUTERS



Model II 64K $3349

An excellent computer for your business needs. Easy expandability & compatibility. No formal operator training needed. All accessories available—disk expansions, printers, software, at our low discount prices. Our fast Dallas air freight service can assure most deliveries within seven days after payment is received

Free with purchase:
1 box of 10 double density diskettes. A $69 value.

SUGART DISK EXPANSION for the MODEL II

Uses the same reliable drive units found in the Model II CPU itself. Allows matched drives for reliable operations. Fast access time under CP/M or TRSDOS. Fully compatable with all existing software. Ready lights show disk activity. Silver/black paint matches Model II. The best drives available for less.

1 Drive, Single Cabinet $849
2 Drives, Triple Cabinet $1424
1 Drive, Triple Cabinet $949
3 Drives, Triple Cabinet $1899
Extra Drives $475

CORVUS HARD DISKS for the TRS-80 $Call

MICROSOFT

BASIC-80 - Extended disk Basic, long variable names, chaining, variable length records, ANSI compatable, interpretative version. **$294 w/manual. $30 manual only.**

BASIC COMPILER - Same Basic language as BASIC-80 except compiles Basic program into a directly executable, relocatable machine language file. Operates 3 to 10 times faster than regular Basic. Includes Macro-80 Assembler. Specify CPM or TRSDOS. **$325 w/manual. $30 manual only.**

FORTRAN-80 - Supports ANSI 66 plus many extensions including MACRO-80. Has library with manager, relocatable object compiler, and linking loader. **$400 w/manual. $30 manual only.**

COBOL-80 - Versitility gives Level 1 ANSI and most of Level 2. Allows complete sequential, relative and indexed file support with variable file names STRING, UNSTRING, COMPUTE, VARYING/UNTIL, EXTEND, CALL, COPY, SEARCH, 3 dimensional arrays, compound and abbreviated conditions, nested IF. Interactive screen-handling extensions. Compatable assemler, linking loader, and relocatable library manager. **$575 w/manual. $30 manual only.**

Special—With TCS Memory

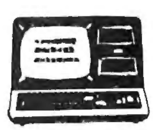
★ **Model III 48K 2 Disks $1895**

Uses proven MPI drives modified for flippy operation and Percom controller. Limited 90 day warranty. Call for details.

Model III 16K $849
Model III 32K $979
Model III 48K $1089

With TCS Memory:
Model III 32K $909
Model III 48K $969

Model III 32K 2 Disk RS232 $2100
Model III 48K 2 Disk RS232 $2230
Model III 32K 1 Disk $1729
Model III 48K 1 Disk $1849



- Payment: Money Order, Cashier's Check, Certified Check. Personal checks take 3 wks. VISA, MC, add 3%.
- Prices subject to change any time.
- No tax out-of-state. Texans add 5%.
- Delivery subject to availability.
- Shipping extra, quoted by phone.

TEXAS COMPUTER SYSTEMS ✓25

Box 951, Brady Texas 76825

For fast, efficient service, we can air freight from Dallas to major a/p near you. Call for information.

Toll Free Number 800-351-1473

Texas Residents 915-597-0673

Program continued

```
750  DATA "CALF IS TO COW AS CUB IS TO","A) SCOUT","B) BEAR",
          "C) BABY","D) WOODS","B"
760  DATA "SUMMER IS TO WINTER AS EVENING IS TO","A) SUNSET",
          "B) COOLNESS","C) MORNING","D) DARKNESS","C"
770  DATA "WORDS IS TO BOOK AS NOTES IS TO","A) PIANO",
          "B) SONG","C) SCULPTURE","D) FRAGNANCE","B"
780  DATA "FISH IS TO FINS AS BIRD IS TO","A) FLY","B) FEATHER"
,           "C) WINGS","D) PIGEON","C"
790  DATA "PIPE IS TO PLUMBER AS BOARD IS TO","A) MECHANIC",
          "B) PROPELLER","C) CARPENTER","D) LACQUER","C"
800  DATA "WIRE IS TO ELECTRICITY AS PIPE IS TO","A) PEANUTS",
          "B) OPERATION","C) ROUND","D) WATER","D"
810  DATA "HORSE IS TO HERD AS MOUNTAIN IS TO","A) VOLCANOE",
          "B) ROCK","C) HILL","D) RANGE","D"
820  DATA "SINKS IS TO ROCK AS FLOATS IS TO","A) LIGHT",
          "B) WOOD","C) FLIES","D) DROWNS","B"
830  DATA "PARACHUTE IS TO PLANE AS LIFE PRESERVER IS TO",
         "A) FISH","B) BOAT","C) WATER","D) CHEST","B"
840  DATA "ACT IS TO PLAY AS CHAPTER IS TO","A) PAGE",
          "B) BOOK","C) LIBRARY","D) TITLE","B"
850  DATA "TRAIN IS TO LAND AS STEAMSHIP IS TO","A) LANE",
          "B) OCEAN","C) PLANET","D) CAPTAIN","B"
860  DATA "PIPE IS TO WATER AS ARTERIES IS TO","A) VEINS",
          "B) BLOOD","C) BOILER","D) HEART","B"
870  DATA "COUNTRY IS TO ARGENTINA AS STATE IS TO","A) EARTH",
          "B) ASIA","C) BOSTON","D) IDAHO","D"
880  DATA "MONTH IS TO MARCH AS SEASON IS TO","A) MAY",
         "B) SNOW","C) SPRING","D) FLOWERS","C"
890  DATA "SUBMARINE IS TO FISH AS AIRPLANE IS TO","A) AQUARIUM
          ","B) BIRD","C) WING","D) HANGER","B"
900 CLS:RESTORE:CLEAR
910 PRINT@465,"TO PLAY AGAIN,TYPE:    RUN"
920 PRINT:PRINT:PRINT:PRINT:PRINT:END
```

Create Basic data statements from object code disk files.

Dateline 80

Richard A. Press, MD.
493 American Drive
Yardley, PA 19067

If you've been merging Assembly program routines with your Basic programs by means of string packing, you have no doubt run into a bit of drudgery—mainly, converting your object code from hexadecimal into decimal numbers to use in data statements.

This program will save you the trouble. It reads the object code right from the disk file you created with the editor/assembler, and then creates a disk file of Basic data statements complete with line numbers, ready to merge into your program. Dataline was written with the help of the book, *TRS-80 Disk And Other Mysteries* by Harv Pennington.

The program works as follows: First, establish the file names to be dealt with. Next, dimension an integer array to contain the object code as it is read from the object file. The object code is read in 256-byte segments and transferred from the buffer strings to the array as ASCII codes. When the end of the file is reached, the array is checked to see that the first byte is 01 (identifying it as an object code file, as opposed to a Basic file, for example, which would have a different first byte). The number of bytes to be loaded are then obtained, the address codes being skipped, and then your codes are printed with data statements into the output file in groups of seven. You can change the number of data statements per line by changing line 45.

With this program, you could build a library of Dataline files of your favorite Assembly program routines, ready to merge with your Basic programs.

One thing to note: When you assemble your routines, add three zero bytes to the end of each program, to give a quit signal to Dataline. ■

```
1 REM    ***         DATALINE/SBS       ***
2 REM    ***           12/06/80         ***
3 REM    ***         R. A. PRESS, M.D. ***
4 CLEAR 1000 : CLS
5 FI$ = "" : LINE INPUT "<FILE SPEC/EXT:DN> OF INPUT FILE? "; FI
$
6 IF FI$ = "" THEN 5
7 LINE INPUT "<FILESPEC/EXT:DN> OF OUTPUT FILE? "; FO$
8 IF FO$ = "" THEN 7
9 IF FI$ = FO$ THEN 5
10 INPUT "LINE NUMBER TO BEGIN WITH (0-60500)"; LN!
11 IF NOT (LN!> 0 AND LN!<60501) THEN  10
12 INPUT "INCREMENT LINES BY HOW MUCH"; IC%
13 IF NOT (IC% > 0) THEN  12
14 LN! = LN!-IC%
15 PRINT "OPENING FILE: "; FI$
16 OPEN "R", 1, FI$
17 PRINT "OPENING FILE: "; FO$
18 OPEN "O", 2, FO$
19 NB = LOF(1) * 256
20 DIM A%(NB)
21 FIELD 1, 255 AS A$, 1 AS B$
22 FOR Z = 1 TO LOF(1)
23 GET 1
24 FOR ZA = 1 TO 255
25 K = K + 1
26 A%(K) = ASC(MID$(A$,ZA,1))
27 NEXT ZA
28 K = K + 1
29 A%(K) = ASC(B$)
30 NEXT Z
31 FL = 0 : Z = 1 : SW = -1 : K = 0
32 IF A%(Z) <> 01 THEN PRINT" BAD LOAD." : CLOSE : END
33 LD = A%(Z+1)
34 FOR ZA = 4 TO LD+1
35 IF NOT (SW = -1) THEN  37
36 GOSUB  500
37 IF NOT (A%(ZA+Z) = 0) THEN  41
38 IF NOT (A%(ZA+Z+1) = 0) THEN  41
39 IF NOT (A%(ZA+Z+2) = 0) THEN  41
40 FL = -1
41 IF NOT (FL <> -1) THEN  55
42 PRINT USING "###"; A%(ZA+Z);
43 PRINT#2,USING"###";A%(ZA+Z);
44 K = K + 1
45 IF NOT (K > 6) THEN  50
46 K = 0 : SW = -1
47 PRINT
48 PRINT#2,""
49 GOTO  52
50 PRINT",";
51 PRINT#2,",";
52 NEXT ZA
53 Z = Z + LD + 2
54 IF Z => NB THEN FL = -1
55 IF NOT (FL = -1) THEN  32
56 IF NOT (SW = -1) THEN  58
57 GOSUB  500
58 PRINT#2, -1
59 PRINT -1
60 CLOSE
61 END
500 LN! = LN! + IC%
501 PRINT#2, LN!; "DATA ";
502 PRINT LN!; "DATA ";
503 SW = 0
504 RETURN
```

Program Listing

See List of Advertisers on page 354

A hybrid language for programmers who need structure in their lives.

Bascal

Richard C. Metzler
9512 Mesa Arriba Court NE
Albuquerque, NM 87111

Does it depress you when you read articles which criticize Basic? Unstructured, not self-documenting, and bad subroutine procedures are some of the epithets hurled at Basic by the supporters of Pascal, C and other structured languages. Would you like to use a Pascal program but find yourself stymied by the lack of While and Repeat commands?

Bascal

We can introduce some new commands into our familiar Level II Basic which, together with some new programming techniques, turns Basic into a structured language. I have christened this new dialect of Basic "Bascal".

But why not just buy a commercial tape or disk of Pascal? Well, the tiny versions restrict you to integers and cripple your string-handling capabilities while even the big 32K and disk versions don't provide double precision. I hate to give up the sophisticated string-handling and double-precision arithmetic of Level II. Also it's a nuisance to learn new names for old commands.

On the other hand, it is no longer in dispute that structured programming is easier to read, debug and change. Once you are used to it, it's even easier to write the program from the start. Let's look at some of the good features of structured programming and see if we can incorporate them into Basic.

Self-documentation

Self-documenting is one of the jargon words applied to structured languages. One aspect of this is self-explanatory variable names such as TTAL for total instead of X or Y and BLNCE for balance instead of A or B. We can already do this in Level II if we are careful to avoid two problems. The first occurs when we embed a reserved word in our variable name. For example TOTAL instead of TTAL would give us a syntax error because of the reserved word "TO" and COST bombs because of the COS function. One way to minimize this is to omit vowels so that the resulting name is recognizable but unlikely to be in the reserved word list (it doesn't work for CMD however).

The second problem is due to the fact that Level II recognizes only the first two characters. We have to be alert to avoid designating the same variable with two different names.

> *"We can introduce some new commands into our familiar Level II Basic which... turns Basic into a structured language. I have christened this new dialect of Basic 'Bascal.'"*

Another aspect of self-documentation is the subroutine (Pascal procedure) name. The Basic line GOSUB 5000 just doesn't cut it compared to GETNEXTENTRY or DISPLAYRESULTS. Here's how we can incorporate this desirable feature into our Basic programs.

Instead of GOSUB 5010, we replace the line with T$ = "PRINTTABLE":GOSUB9. At line 9 we find the heading for our subroutine table.

```
9
REM **** SUBROUTINE TABLE ****
10
:IF T$ = "FINDNEXTENTRY" THEN
   X = Y + 7
   :PRINT X,Y ...etc.
:RETURN
20
:IF T$ = PRINTTABLE" THEN
      :FOR N = 1 TO K
         :PRINT N,X(N);
      :NEXT
      :RETURN
```

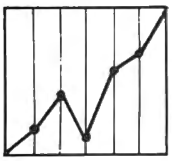
The idea should be clear from this example. Each subroutine is written out before the main program. When the main program (or another subroutine) calls a subroutine, the computer goes to line 9 and searches down until it finds a match for T$ and then executes that subroutine. At the end of the subroutine list we put in the

6 ways to give your TRS-80* a +PLUS

Exhaustive research and professional standards bring you the best in business software.

EASYTRAK tm

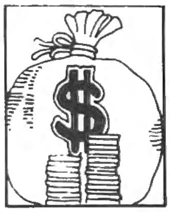
KEEP YOUR SALES CURVING UPWARD WITH EASYTRAK tm, a powerful, all-purpose activity tracking system. User-specified options enable you to monitor sales activity by territory, product line, salesman, or by combinations. Can monitor inquiries, sales calls and order booking. Production activity (by shift, by product, by day, by team, etc.), spending or income activity are simple to set up with EASYTRAK tm. Extensive video displays make available individual, group or total summaries, relative averages, and percentage data all at the touch of a button. Use of an 80-column printer is optional.



MODEL I/III $125
MODEL II $150

+ACCOUNTS RECEIVABLE

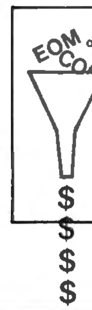
PUT MORE CASH IN YOUR BANK ACCOUNT with +Account Receivable's many collection/credit features, including average payment days, aging schedule and statements. Open-item system records sales activity this year and last, optional automatic discounts and salesman assignment. Cash receipts and A/R adjustments feature extensive on-line validation. And, yes, +Accounts Receivable handles the pesky new 9-digit zip code. Other optional features include service charges to selective customers; partial and "on-account" payments. Posts, at your option, to +General Ledger.



MODEL I/III $325
MODEL II $350

+ACCOUNTS PAYABLE

FUNNEL THAT STACK OF BILLS into +Accounts Payable system and create an orderly, manageable flow of vendor histories, cash requirements forecasts, check registers and checks, aging schedules, etc. Will accept manual checks, reversals, and adjustments. Pay by selected vendor, invoice, due date, discount dates, and optionally post directly to your +General Ledger. Supports 5-digit vendor numbers.



MODEL I/III $250
MODEL II $300

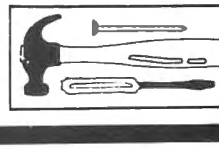
+GENERAL LEDGER

SLICE OUT MORE PROFIT by using +General Ledger to monitor your financial status. Features include fully-flexible financial statements, both budget and last year comparisons, special report options, monthly and quarterly trial balances, manual and/or automatic journal entries all with extensive audit trails. You structure the chart of accounts with up to 5 digit account numbers.

MODEL I/III $325
MODEL II $350

+FIXED ASSETS

A POWERFUL SET OF TOOLS TO CALCULATE and monitor fixed asset expenditures, book and tax depreciation, investment tax credit, etc. Will even forecast future depreciation for you. (Handy at budget time!) Code structure enables you to classify assets by departments, buildings, groups or other categories. Handles seven depreciation methods. Can post to +General Ledger for you.



MODEL I/III $325
MODEL II $350

+PAYROLL

KEEP YOUR EMPLOYEES HAPPY and smiling with prompt, accurate and professional payroll records. Will handle weekly, biweekly, semimonthly, monthly payrolls, produce checks, check registers, labor reports, earnings reports, Form 941's, W2's, etc. Can handle fixed/voluntary deductions, hourly/salary employees, bonuses, city and state taxes, etc. Easy to use and update tax tables. Automatic +General Ledger posting.

MODEL I/III $325
MODEL II $350

SPECIAL! All of the above: MODEL I/III $1250 MODEL II $1500

+PLUS SOFTWARE comes to you feature-packed from on-line validation to machine-language file handling. Extensive documentation and thorough installation guide accompany each order. All systems are furnished on disc. Source code is included. Texas residents add 5% sales tax.

VISA master charge

To order, call TOLL-FREE 24 hours
1-800-528-6050 EXT. 1562
(In Arizona, 1-800-352-0458 Ext. 1562)

+PLUS

PLUS COMPUTER TECHNOLOGY, INC ✓131
POST OFFICE BOX 1152
ANGLETON TEXAS 77515 (713) 849 1108

Dealer inquiries invited

*TRS-80 is a registered trademark of Tandy Corp.
*EASYTRAK is a trademark of Plus Computer Tech. Inc.

✓See List of Advertisers on page 354

statement 'PRINT "SUBROUTINE NOT LISTED": STOP'. This tells us if we try to call a subroutine which is not listed because of a typing error or some other problem.

Another feature of the program fragment I would like to point out is the use of indentation. In Basic, indentation is very valuable in showing the meaning of a program but is expensive in memory space if you use only one statement per line. I suggest using the down-arrow to format your programs. If every line number is immediately followed by the line feed resulting from a colon followed by a down-arrow, it's clear that we can arrange it so that all commands start in the second column, except those that are indented three or six or more spaces. The first column will contain only line numbers, spaces or colons. The memory price we have to pay for this is minimal—one byte for each down arrow and one for each space of indentation. I was very disappointed to discover that tabbing with the right arrow doesn't save any more bytes than just spacing with the space bar. Even so, this technique lets you format a 250-character line at minimal cost in memory.

"All this is helpful", the Pascal prophets might say, "But what about the lack of a While or Repeat . . . Until command and what about all those hard-to-follow GOTO statements?" We respond to this by inventing the required commands and renouncing the use of GOTOs and even GOSUBs except for the GOSUB9 which takes us to the subroutine list.

Inventing Commands

Computer scientists have shown that GOTO commands are completely superfluous if the language has the While, Repeat . . . Until and If . . . Then . . . Else commands. We need only invent the While and Repeat . . . Until commands. This can be done by using logic commands to adapt the For . . . Next loop to our needs.

The following set of lines gives us a Repeat . . . Until:

```
FORR=1T01
   :....
   :..(statements)
   :....
:R=-(condition):NEXT
```

It sure looks like a funny For loop doesn't it? Why would we bother to go from one to one? To see that there is method in this madness let's look at the final condition which will typically be something like K>0, A$ = B$ or Q = 5. For all of these statements Level II Basic returns a −1 if the y are true and a 0 if they fail to hold. Thus, if the condition holds, R becomes −(−1) and Next increments it to 2 which causes us to exit the For...Next loop. If the condition does not hold R becomes 0 and the Next makes it 1, causing us to return to the For statement and execute the loop again. Thus we repeat the statements inside until the condition is true.

It's true that we could achieve the same effect with a conditional GOTO but this involves using line numbers. The above method also has the advantage that it will work in the middle of one numbered line consisting of many statements. Level II Basic remembers where the For is located when it encounters the corresponding Next.

The While command is a little tougher to handle without using line numbers. We do it like this:

```
FORW=0TO0: IF (condition) THEN
W=-1
   :....
   :..(statements)
   :....
:NEXTELSENEXT:W=-1
```

This version is only adequate if we can fit it all into the 255 characters Basic gives us for a single numbered line. If the statements spill over into the next line we need some way of ensuring that they will only be executed while the condition is true. An economical way of doing this is to precede each numbered line until the final Next with the cryptic statement IFW as below:

```
1000:
:FORW=0TO0: IF (condition) THEN
W=-1
   :...
   :..(statements)
1010:
:IFW
   :...
   :..(statements)
1020:
:IFW
   :....
   :....
   :..(statements)
1040:
:NEXT:W=-1 REM *** END OF
WHILE LOOP ***
1050:
following statements
```

If the condition is false, then W is zero and it is not changed to −1. Then the command IFW falls through to the next line number since W=0. This will happen at every IFW so we will plummet all the way down to the Next at line 1040. This will increment W to 1, causing an exit from the For loop. On the other hand, if the condition is true, W becomes −1 and all the statements are executed since IFW checks the value of W and falls through to the next line only when W=0. The Next at the bottom increments W to 0 and sends it back to the For which tests the condition to see if it should go through the loop again.

If you can't stand the thought of all those extra IFW statements you could use a dreaded GOTO and substitute the following for line 1000 above:

```
1000:
FORW=0TO0: IF NOT(condition)
THEN GOTO 1040
  ELSE W= -1
  :....
  :..(statements)
```

Then the IFW statements can be removed. I don't recommend the alternative GOTO 1050 which would seem to have the same effect since this leaves an active For...Next loop in the program. Each such loop requires 16 bytes. If the active loop is inside another loop you run the risk of running out of memory as you store 16-byte chunks on each pass.

Ordinarily, I like to avoid running words together in my programs, but I do it on purpose in FORR=1TO1 and FORW=0TO0 to make it easier to distinguish these loops from an ordinary For loop. Also, R is always used in the Repeat loops and W in the While loops. This brings up another point. What if we have one While loop nested inside another one? We can't use the same variable W for both loops since Level II forgets the return address for the containing loop when it encounters the second FORW=0TO0. The result is a Next-Without-For error when the program reaches the outer Next. To solve this problem we use W1, W2...W9 for successive interior loops. Of course, R1...R9 would be used for nested Repeat...Until loops.

However, the W1, R1 trick is not necessary when a subroutine is called in the middle of a loop. Level II keeps track of things so that even if a subroutine in the middle of a Repeat ... Until loop has another such loop using the same variable R, no problems occur. However, with a While loop we would have trouble on returning from a subroutine While loop which had incremented W to 1. This is the reason for the assignment W= −1 after the final Next in my While loops. This ensures that the loop will work all right if it is used as a subroutine loop inside a larger While loop. Unfortunately, this technique works only when there is a subroutine call. Ordinary nested loops must use the W1, R1 method.

In order to avoid the troubles which occur when two variables are given the same name I try to use two-letter names for variables peculiar to the program and reserve single-letter and letter-number names for standard loop variables. In fact the Bascal variables are explicitly listed in the sample program in lines 110 and 140.

Collision of Variables

Another attractive feature of Pascal is the way the problem of collision of variables is avoided. With a little work we can approximate this in Basic. When we write a subroutine we keep track of all variables used and then write the following first line:

```
IF T$="PRINTTABLE" THEN
I%(I)=COUNT:I=I+1:I%(I)=
```

Program Listing

```
5 :
REM **** NUMCON IN BASCAL - VER. 6 - - NOV. 9, 1980  *******
8 :
T$="RUN"
9 :
T=0:REM **** SUB TABLE STARTS HERE ******
10 :
:IF T$="FLASHINGCURSORINPUT" LET C$(C)=IN$:C=C+1
  :IN$=""
  :FORW=0TO0:IFIN$<>CHR$(13) LETW=-1
    :T$="DISPLAY&STORE":GOSUB9
    :T$="FLASH&GETKEYPRESS":GOSUB9
    :NEXTELSENEXT:W=-1
  :PRINT CHR$(15);
  :C=C-1:IN$=C$(C)
  :RETURN
20 :
:IF T$="CHANGETODECIMAL" THEN
   I%(I)=DDIGIT:I=I+1:I%(I)=DIGIT:I=I+1
  :FOR DIGIT=1 TO LEN(IN$(CO))
    :DDIGIT=ASC(MID$(IN$(CO),DIGIT,1))
    :DD=DD-48+7*(DDIGIT>57)
    :DEC!=DEC!*RADIX(COUNT)+DDIGIT
  :NEXT
  :I=I-1:DI=I%(I):I=I-1:DD=I%(I)
  :RETURN
30 :
:IF T$="CHANGEDECIMALTORADIXBASE" THEN LET T=1
  :S!(S)=DEC!:S=S+1:S!(S)=QUOTIENT!:S=S+1:I%(I)=RMNDER:I=I+1
  :FORR=1TO1
    :QU!=INT(DEC!/RADIX(COUNT))
    :RM=DEC!-RADIX(COUNT)*QU!
    :OU$(COUNT)=CHR$(RM+48-7*(RM>9))+OU$(COUNT)
    :DEC!=QU!
35 :
:IFT
  :R=-(DEC!=0):NEXT
  :I=I-1:RMNDER=I%(I):S=S-1:QUOTIENT!=S!(S):S=S-1:DEC!=S!(S)
  :RETURN
40 :
:IF T$="FLASH&GETKEYPRESS" LET T=1
  :I%(I)=CU:I=I+1:I%(I)=WAIT:I=I+1
  :CU=15
  :FORR=1TO1
    :CU=29-CU
    :PRINT CHR$(CU);
    :FOR WAIT=1 TO 10
      :IN$=INKEY$
      :WAIT=WAIT-10*(IN$>"")
    :NEXT
    :IN=ASC(IN$+CHR$(1))
45 :
:IFT THEN
    R=-(IN=8 OR IN=13 OR (47<IN AND IN<58 AND IN<RA(CO)+48) OR
    (64<IN AND IN<RA(CO)+55)):NEXT
```

Program continues

✓See List of Advertisers on page 354

Program continued

```
    :I=I-1:CU=I%(I):I=I-1:WAIT=I%(I)
    :RETURN
50 :
:IF T$="DISPLAY&STORE" LET T=1
  :I%(I)=LE:I=I+1
  :LE=LEN(IN$(CO))
  :IF NOT((LE=0)AND(IN=8)) THEN
     LE=LE+1+2*(IN=8)
    :IN$(CO)=LEFT$(IN$(CO)+IN$,LE)
    :PRINT IN$;
55 :
:IFT
   LET I=I-1:LE=I%(I)
  :RETURN
90 :
:IF T$<>"RUN" THEN
  PRINT "SUB NOT LISTED": STOP
1000 '
:REM **** MAIN PROGRAM ****
1010 '
:***** INITIALIZATION *******
1020 :
:CLEAR 500: DEFINT A-Z
1030 :
:C=0:D=0:I=0:R=0:S=0:T=0:W=0:T$=""  '*** BASCAL VARIABLES
1040 :
:DIM C$(9):DIM D#(9):DIM I%(9):DIM S!(9) '** BASCAL STACKS
1050 :
:COUNT=0: DEC!=0: MSB!=0
:ERASE$=STRING$(16," ")+STRING$(17,8)
1060 :
:DIM RADIX(8):DIM IN$(8): DIM OU$(8):DIM AT(8)
1070 :
:FOR COUNT=1 TO 5
  :AT(COUNT)=80+(COUNT-1)*128
:NEXT
:AT(6)=AT(5)+32:AT(7)=AT(5)+128:AT(8)=AT(7)+32
1080
:RADIX(1)=10:RA(2)=2:RA(3)=8
:RA(4)=16:RA(5)=16:RA(6)=16
:RA(7)=10:RA(8)=10
1100 :
:REM **** PROGRAM LOOPS IN REMAINING PORTION *****
1110 :
:CLS
```

Program continues

```
NUMBER:I=I+1
:SI(S)=BLNCE!:S=S+1
:D#(D)=TTAL#:D=D+1
:C$(C)=LABEL$:C=C+1
:....
:..(body of subroutine)
:....
:I=I-1:NUMBER=I%(I):I=I-1:
COUNT=I%(I)
:S=S-1:BLNCE!=SI(S)
:D=D-1:TTAL#=D#(D)
:C=C-1:LABEL$=C$(C)
:RETURN
```

The variables Number, Count, BLNCE!, TTAL# and LABEL$ appear in the subroutine and may be changed there. If they are used in the main program or in some other subroutine, values will not be changed inadvertently since, on entrance to the subroutine, they are saved by being PUSHed onto a stack and are then POPped off on exit and their entrance value restored. Of course, this is only done for variables which contain data that is not supposed to be changed by the subroutine.

Program Listing

The Program Listing gives an example of the use of Bascal. It converts numbers from decimal to binary, octal and hex and vice-versa. It also provides the Z-80's most significant byte and least significant byte in both hex and decimal.

After the title line we have the statement T$ = "RUN" in line 8. This means that an initial command of Run will cause the program to fall through all the subroutines to the start of the main program. In line 9 we set T equal to zero. This is just a continue flag which makes it easier to continue a subroutine to another numbered line. It also means that if we use a remark-removing packing program we won't lose line 9.

Skipping over the subroutines for the moment we come to the start of the main program at line 1000. We start by clearing some string space and defining all variables as integers since few of our variables are anything but integers. After initializing the Bascal variables

Subscription Problem?

80 Microcomputing does not keep subscription records on the premises, therefore calling us only adds time and doesn't solve the problem.

Please send a description of the problem and your most recent address label to:

80 Microcomputing
Subscription Dept.
PO Box 981
Farmingdale, NY 11737

Thank you and enjoy your subscription.

Program continued

```
:PRINT "        <ENTER> TO CYCLE TO NEXT POSITION
DECIMAL REP. =
*
BINARY REP.  =
*
OCTAL REP.   =
*
HEX. REP.    =
*
LSB(HEX)     =                    MSB(HEX)    =
*
LSB(DEC)     =                    MSB(DEC)    ="
1120 :
:COUNT=0
:FORR=1TO1
  :COUNT=1-COUNT*(COUNT<8)
  :PRINT @ AT(COUNT),;
  :T$="FLASHINGCURSORINPUT":GOSUB9
  :R=-(IN$(COUNT)>"" AND (COUNT<5 OR (COUNT=6) OR COUNT=8)):NEXT

1130 :
:IF COUNT=6
   LET IN$(6)=RIGHT$("00"+IN$(6),2)+RIGHT$("00"+IN$(5),2)
1140 :
:IF COUNT=1 LET DEC!=VAL(IN$(1))
 ELSE IF COUNT=8 LET DEC!=256*VAL(IN$(8))+VAL(IN$(7))
 ELSE T$="CHANGETODECIMAL":GOSUB9
1150 :
:PRINT @ AT(1),ERASE$;CHR$(8);DEC!;
1160 :
:FOR COUNT=2 TO 4
  :T$="CHANGEDECIMALTORADIXBASE":GOSUB9
  :PRINT @ AT(COUNT),ERASE$;OU$(COUNT);
:NEXT
1170 :
:OU$(4)=RIGHT$("000"+OU$(4),4)
:PRINT @ AT(5),ERASE$;RIGHT$(OU$(4),2);
:PRINT @ AT(6),ERASE$;LEFT$(OU$(4),2);
1180 :
:DEC!=DEC!
:MSB!=INT(DEC!/256)
:PRINT @ AT(7),ERASE$;CHR$(8);DEC!-256*MSB!;
:PRINT @ AT(8),ERASE$;CHR$(8);MSB!
:PRINT:PRINT "PRESS ENTER FOR NEW PROBLEM  ";: INPUT IN$(1)
1190 :
:RUN
```

and stacks we initialize the variables and arrays which carry program information.

In line 1100 we start the running part of the program. Line 1110 contains our first Repeat . . . Until loop. In the loop we start with Count incrementing from zero to one and then we call the input routine which loops and accepts keyboard input until Enter is pressed. If only Enter was pressed we see that IN(COUNT) = " " and we cycle back to the For and increment Count. Note that the COUNT<8 factor returns a (−1) until Count reaches eight when it returns zero. Thus if the operator continues to press Enter he will cycle Count from one to eight and back again, endlessly. When the input string is non-empty we break out of the loop unless we still need the most significant byte.

Dropping through to the next line, if the operator just entered the lsb and msb of a hex number at Count = 5 and Count = 6 then we put them together to form one four-digit *hex number.* If Count = 1 and the operator just entered a decimal number then we simply take the numerical value of the input string. If Count = 8 and the operator just gave us the lsb and msb in decimal form we calculate the actual value as 256*msb + lsb. Finally, in any other case we call the subroutine which changes the number in some base other than 10 to a decimal number.

Now we print the decimal number in the first position and call the subroutine which changes the decimal into the various bases. We take the hex output, add some zeros on the front so that we are sure we have four digits, and peel off the first two and last two digits for the msb and lsb. Finally, we calculate the msb and lsb in decimal form. After the results are all printed we print the Press-Enter-for-New-Problem message. When any key is pressed we fall through to Run which zeros all variables and

See List of Advertisers on page 354

sends us back to line 5.

Notice how the long variable names and subroutine names help in reading the program. The IN$ array collects keyboard input and the OU$ array collects program output to be printed. The function of each subroutine is clear from its name. Now let's go back to the subroutine table to examine it more closely. If T$ = "FLASHINGCURSOR INPUT" then when we GOSUB9 we execute the commands at line 10. There we first store IN$ on the character string stack so that any calling program could use the variable IN$ with no fear of collision, since the entering value would be retrieved from the stack just before the return. Then we initialize IN$ and drop into a While loop. While the keypress is not an Enter we display it on the screen and store it in IN$(Count). We flash the cursor while waiting for each keypress. In the routine FLASH &GETKEYPRESS, after saving variables on the stack as usual, we set CU equal to the cursor off control character and enter the Repeat...Until loop.

Changing CU to 29-CU toggles CU to cursor on and it's clear that we will toggle between on and off as we cycle through the loop. The For loop gives a delay for the on and off times. If there is a keypress during the loop, Wait becomes Wait minus 10 times (– 1) so we exit the loop. The ASC function returns the ASCII number of the first character of the string inside the parentheses so if we exited normally from the For loop without a keypress it will become 1. The alternative IN = ASC(IN$) causes an Illegal Function Call if IN$ = ". The next line is our complicated Until condition. If the keypress was a backspace, an Enter, a number or letter that makes sense in the base we are using we return. Otherwise we just loop back to the Repeat.

On returning from FLASH& GETKEYPRESS we cycle back to the start of the While loop. Assuming that the keypress was not an Enter we call DISPLAY&STORE. In that routine we merely return if the keypress was the first entry and was a backspace. If not we add IN$ to the end of IN$(CO) and print it on the screen. The assignment involving 2*(I = 8) has the effect of subtracting one from the length of IN$(CO) if the keypress was a backspace. In this case the LEFT$ function will remove the character last entered.

The net effect of all these routines is to allow the operator to enter and correct only those entries which make sense in the base in use.

The subroutine "CHANGETODECIMAL" is a straightforward calculation. The digits of the entry having radix (or base) equal to Radix(Count) are peeled off from the left and changed to ASCII. The assignment involving 7*(DDIGIT>57) has the effect of skipping the ASCII codes from 58 to 64 in order to give the correct decimal equivalent when 40 is subtracted.

The last subroutine, CHANGE DECIMALTORADIXBASE, uses the same sort of trick in the OU$ line to change from decimal numbers to characters which are appropriate for the base in question. One element worthy of comment is the use of the variable DEC!. This is used as a variable which is reduced to zero during the calculation. However it is also the variable in the main routine which holds the value needed for three separate calls to this subroutine. This could lead to a hard-to-find error but our stack-saving discipline means that the problem disappears. On each exit from the routine DEC! is restored to the value it had on entrance so it is ready for another subroutine call.

So that's it, programmers. Get out that book on Pascal that's been gathering dust on your bookshelves, fire up your '80 and start programming in Bascal. ■

COMPUTER SHOWS

SECOND ANNUAL
NJ MICROCOMPUTER SHOW AND FLEAMARKET - 1981

NEWARK, NEW JERSEY
Sat. Oct. 24 & Sun. Oct. 25
10-5:30 PM & 10-3:00 PM
HOLIDAY INN (NORTH)
NJ Turnpike • Exit 14
(Take Route 1-9 Service Road - Follow Signs)
REGISTRATION $5.00 -
FLEAMARKET ONLY $3.00
Fleamarket Permits $5/Spot
(Includes Admission)
Commercial Exhibits • User Groups • Fleamarket

SECOND ANNUAL
NATIONAL "80" MICROCOMPUTER SHOW (FOR TRS-80* USERS)
April 29, 30, May 1, 1982
NEW YORK STATLER HOTEL
NEW YORK CITY
100 Exhibit Booths • Speakers • User Groups
*TRS-80 is a Trademark of Tandy Corporation

For Registration, Exhibitor or Fleamarket Information Contact:
KENGORE CORPORATION
3001 Route 27 • Franklin Park, NJ 08823
(201) 297-2526 ✓230

√ Check

our book pages for the latest books about microcomputers!

—WAYNE GREEN BOOKS—

A Division of Wayne Green Inc.
Peterborough, NH 03458

FREE 48-Page Catalog

A new, free 48-page catalog is free from Creative Computing and Peripherals Plus. To help buyers make intelligent purchasing decisions, the product descriptions are exceptionally comprehensive and include screen photos in the software section.

The catalog describes 20 books on programming, games, and educational applications; 160 software packages for Apple, Atari, TRS-80, PET, CP/M, TI, Sorcerer and Sol computers; 3 magazines (*Creative Computing*, *Microsystems*, and *SYNC*); 5 graphics and music peripherals; an LP record; board game; 8 T-shirts and an eclectic assortment of other products for the personal computer user.

To get your free copy, simply drop a card or note to the address below.

creative computing
Attn: Becky ✓512
39 East Hanover Avenue
Morris Plains, NJ 07960

Wages Is Here

FOR THE TRS-80

- **Wages** MEETS PAYROLLS
- **Wages** IS GUARANTEED
- **Wages** COST ***ONLY*** $35.00 FOR MODEL I ($45.00 FOR MODEL II)

FROM
The Maine Software Library
P.O. Box 194 ✓268
Standish, Maine 04084

*A Trade name of the Tandy Corp.

While they last . . .

the complete year of 1980 80 Microcomputing (in its own shelf box . . . a $2.00 value—Free) for only $25.00.*

Did you miss the 2,256 pages of useful information published for your TRS-80** in *80 Microcomputing* last year? Now is the perfect time to catch up for only $25.00.*

In the first year of *80 Microcomputing* here were 335 articles on your TRS-80—2½ times more than any other computing magazine. Also in 1980 there were new product reviews, news and columns and best of all hundreds of dollars worth of useable programs.

So to unlock the key to your TRS-80 and double its value. . .send in today for the complete year of 1980. *80 Microcomputing* (in its own shelf box) for only $25.00.*

We have a limited number of these complete sets so send in the card today or call our toll free #800-258-5473 and charge it to your VISA, MC or AE card.

**plus shipping and handling ($3.00) **TRS-80 is a Trademark of Tandy*

Yes, I want a year of *80 Microcomputing* for $25.00 plus $3.00 for shipping and handling. Plus I'll receive a shelf box to store them in.

☐ Check enclosed for $________
Name ____________________
Address ____________________
City ____________ State ______ Zip ______

Please allow 4 to 6 weeks for delivery
80 Microcomputing • 80 Pine Street • Peterborough, N.H. 03458
Toll free ordering number 800-258-5473
80BI

ReLoad 80

This month we offer LOAD80 tape owners a few notes about our experience with tape loading procedures. We've loaded thousands of tapes at the *80* editorial offices; here are some of the tricks we've learned.

Tape head alignment can vary from machine to machine. Similarly, on any given machine, tape head alignment can vary from day to day, due to heavy use and stress caused by poor quality cassettes. Here at *80 Micro*, we find we must align the tape head on our CTR-41 for almost every tape we load. Tape head alignment is accomplished by turning the small screw (using a Phillips head screwdriver) located on the playback head. To reach this screw, it is necessary to drill a small hole in the plastic case of your tape recorder. (Drill this hole just above the Play button—see diagram.) After access is gained to this adjustment screw, you can alter the audio quality of the tape signal by turning the screw left and right. You can hear this signal if you pull the Ear plug out of your tape recorder.

The goal of all these ministrations is to achieve a sharp, crisp tone—a muffled, muted audio signal won't load correctly.

Sometimes loading problems can be caused by failing to turn off the clock (CMD"T") when running under Model I TRSDOS.

These measures, along with frequent volume adjustments, as needed, allow us to load almost all the tapes we receive. The tapes that won't load—we send back!

If you should have difficulty loading your LOAD80 tape, in spite of all your best efforts, you may have a defective tape or one that was damaged in transit. You can obtain a free replacement by returning the defective tape to:

LOAD80
ATTN: L. Eaton
80 Pine Street
Peterborough, NH 03458
(603) 924-6532

Next month, look for hints on loading source code files using Radio Shack's Editor/Assembler. ■

Pgm #	Page #	Filename	Comments
1	154	ROTATE	None
2	154	ROTATE2	None
3	174	MEMXPD/SRC	Needs EDTASM
4	186	CONVRT/SRC	Needs EDTASM
5	204	HIDPICS	None
6	220	SORTER	None
7	228	SLPTSLPC/SRC	Needs EDTASM
8	244	HRSRCE	None
9	250	ERRPRT	None
10	252	LIFETIME	None
11	286	DISKCHGR	None
12	320	CARTOON	None
13	334	BASCAL	None
14	N/A	LASTEST	From August

September LOAD80 Directory

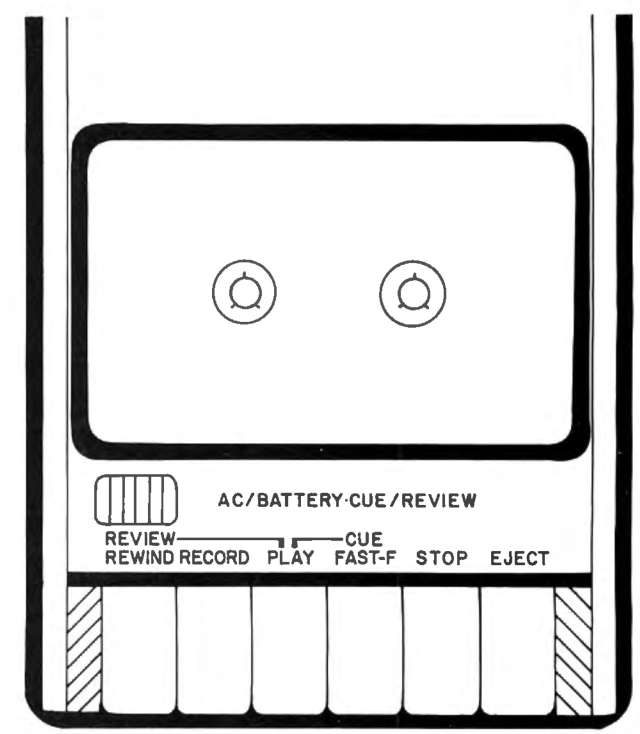


Diagram of Cassette Recorder

✓See List of Advertisers on page 354

80 microcomputing T.M. bookshelf

—BASIC & PASCAL—

new

new

● **INTRODUCTION TO TRS-80 LEVEL II BASIC AND COMPUTER PROGRAMMING**—BK1219—by Michael P. Zabinski. Written by an experienced educator, this is the book for those beginners who want to learn about computers without having to become an expert. It has practical programs, useful line-by-line comments, excellent flowcharts accompanied by line numbers and over 200 exercises which help the reader assess progress, reinforce comprehension, and provide valuable practical experience. $10.95.*

● **50 BASIC EXERCISES**—BK1192—by J. P. Lamoitier. This book is structured around the idea that the best way to learn a language is through actual practice. It contains 50 completely explained exercises: statement and analysis of the problem, flowcharts, programs and actual runs. Program subjects include mathematics, business, games, and operations research, and are presented in varying levels of difficulty. This format enables anyone to learn BASIC rapidly, checking their progress at each step. $12.95*

● **THE BASIC HANDBOOK**—BK1174—**NEW, SECOND EDITION** by David Lien. This book is unique. It is a virtual ENCYCLOPEDIA of BASIC. While not favoring one computer over another, it explains over 250 BASIC words, how to use them and alternate strategies. If a computer does not possess the capabilities of a needed or specified word, there are often ways to accomplish the same function by using another word or combination of words. That's where the HANDBOOK comes in. It helps you get the most from your computer, be it a "bottom-of-the-line" micro or an oversized monster. $19.95.*

● **LEARNING LEVEL II**—BK1175—by David Lien. Written especially for the TRS-80, this book concentrates on Level II BASIC, exploring every important BASIC language capability. Updates are included for those who have studied the Level I User's Manual. Sections include: how to use the Editor, dual cassette operation, printers and peripheral devices, and the conversion of Level I programs to Level II. $15.95.*

● **BASIC BASIC (2ND EDITION)**—BK1026—by James S. Coan. This is a textbook which incorporates the learning of computer programming using the BASIC language with the teaching of mathematics. Over 100 sample programs illustrate the techniques of the BASIC language and every section is followed by practical problems. This second edition covers character string handling and the use of data files. $10.50.*

● **ADVANCED BASIC**—BK1000—Applications, including strings and files, coordinate geometry, area, sequences and series, simulation, graphing and games. $10.75*

● **SIXTY CHALLENGING PROBLEMS WITH BASIC SOLUTIONS (2nd Edition)**—BK1073—by Donald Spencer, provides the serious student of BASIC programming with interesting problems and solutions. No knowledge of math above algebra required. Includes a number of game programs, as well as programs for financial interest, conversions and numeric manipulations. $6.95.*

● **PASCAL**—BK1188—by Paul M. Chirlian. Professor Chirlian's textbook combines a simple approach to the PASCAL language with comprehensive coverage on how a computer works, how to use a flowchart, working from a terminal as well as batch operation and debugging. Special attention is paid to idiosyncrasies of the language and syntax flowcharts abound for the convenience of the experienced programmer. Well indexed. $12.95*

● **INTRODUCTION TO PASCAL**—BK1189—by Rodnay Zaks. A step-by-step introduction for anyone wanting to learn the language quickly and completely. Each concept is explained simply and in a logical order. All features of the language are presented in a clear, easy-to-understand format with exercises to test the reader at the end of each chapter. It describes both standard PASCAL and UCSD PASCAL, the most widely used dialect for small computers. No computer or programming experience is necessary. $14.95.*

● **PROGRAMMING IN PASCAL**—BK1140—by Peter Grogono. The computer programming language PASCAL was the first language to embody in a coherent way the concepts of structured programming, which has been defined by Edsger Dijkstra and C.A.R. Hoare. As such, it is a landmark in the development of programming languages. PASCAL was developed by Niklaus Wirth in Zurich; it is derived from the language ALGOL 60 but is more powerful and easier to use. PASCAL is now widely accepted as a useful language that can be efficiently implemented, and as an excellent teaching tool. It does not assume knowledge of any other programming language; it is therefore suitable for an introductory course. $12.95.*

—GAMES—

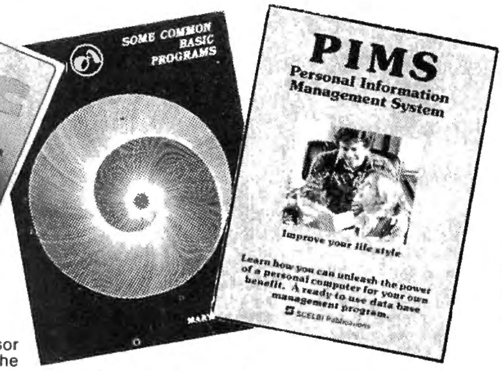
● **40 COMPUTER GAMES**—BK7381—Forty games in all in nine different categories. Games for large and small systems, and even a section on calculator games. Many versions of BASIC used and a wide variety of systems represented. A must for the serious computer gamesman. $7.95*

● **BASIC COMPUTER GAMES**—BK1074—Okay, so once you get your computer and are running in BASIC, then what? Then you need some programs in BASIC, that's what. This book has 101 games for you from very simple to real buggers. You get the games, a description of the games, the listing to put in your computer and a sample run to show you how they work. Fun. Any one game will be worth more than the price of the book for the fun you and your family will have with it. $7.50.*

● **MORE BASIC COMPUTER GAMES**—BK1182—edited by David H. Ahl. More fun in BASIC! 84 new games from the people who brought you *BASIC Computer Games*. Includes such favorites as Minotaur (battle the mythical beast) and Eliza (unload your troubles on the doctor at bargain rates). Complete with game description, listing and sample run. $7.50.*

● **WHAT TO DO AFTER YOU HIT RETURN**—BK1071—PCC's first book of computer games . . . 48 different computer games you can play in BASIC . . . programs, descriptions, many illustrations. Lunar Landing, Hammurabi, King, Civel 2, Qubic 5, Taxman, Star Trek, Crash, Market, etc. $10.95.*

—BUSINESS—



● **PAYROLL WITH COST ACCOUNTING—IN BASIC**—BK1001—by L. Poole & M. Borchers, includes program listings with remarks, descriptions, discussions of the principle behind each program, file layouts, and a complete user's manual with step-by-step instructions, flowcharts, and simple reports and CRT displays. Payroll and cost accounting features include separate payrolls for up to 10 companies, time-tested interactive data entry, easy correction of data entry errors, job costing (labor of distribution), check printing with full deduction and pay detail, and 16 different printed reports, including W-2 and 941 (in **CBASIC**). $20.00.*

● **SOME COMMON BASIC PROGRAMS**—BK1053—published by Adam Osborne & Associates, Inc. Perfect for non-technical computerists requiring ready-to-use programs. Business programs, plus miscellaneous programs. Invaluable for the user who is not an experienced programmer. All will operate in the standalone mode. $14.99 paperback.

● **PIMS: PERSONAL INFORMATION MANAGEMENT SYSTEM**—BK1009—Learn how to unleash the power of a personal computer for your own benefit in this ready-to-use data-base management program. $11.95.*

—MONEYMAKING—

● **HOW TO MAKE MONEY WITH COMPUTERS**—BK1003—In 10 information-packed chapters, Jerry Felsen describes more than 30 computer-related, money-making, high profit, low capital investment opportunities. $15.00.*

● **HOW TO SELL ANYTHING TO ANYBODY**—BK7306—According to *The Guinness Book of World Records*, the author, Joe Girard, is "the world's greatest salesman." This book reveals how he made a fortune—and how you can, too. $2.25.*

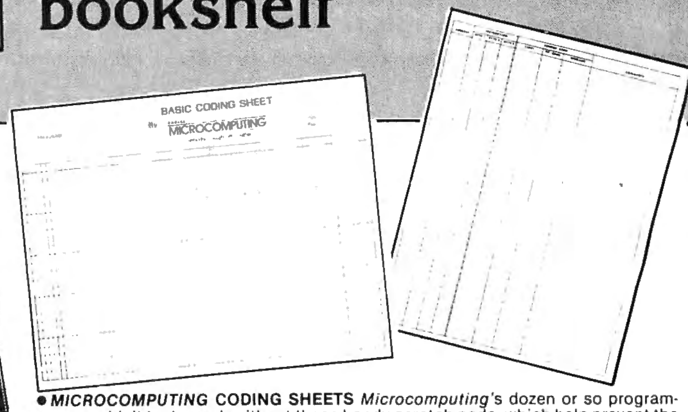
● **THE INCREDIBLE SECRET MONEY MACHINE**—BK1178—by Don Lancaster. A different kind of "cookbook" from Don Lancaster. Want to slash taxes? Get free vacations? Win at investments? Make money from something that you *like* to do? You'll find this book essential to give you the key insider details of what is really involved in starting up your own money machine. $5.95.*

*Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to 80 Microcomputing Bookshelf ● Peterborough NH 03458. Be sure to include check or detailed credit card information.

No C.O.D. orders accepted. All above add $1.50 handling. Please allow 4-6 weeks for delivery. Questions regarding your order? Please write Customer Service at the above address.

FOR TOLL FREE ORDERING CALL 1-800-258-5473

80 microcomputing™ bookshelf



—Z-80—

● **TRS-80 ASSEMBLY LANGUAGE**—BK1217—by Hubert S. Howe, Jr. This book incorporates into a single volume all the pertinent facts and information you need to know to program and enjoy the TRS-80. Included are clear presentations of all introductory concepts, completely tested practical programs and subroutines, details of ROM and RAM and disk operating systems, plus comprehensive tables, charts and appendices. Suitable for the first time user or more experienced users. $9.95.*

● **INSIDE LEVEL II**—BK1183—For machine language programmers! This is a comprehensive reference guide to the Level II ROMs, allowing easy utilization of the sophisticated routines they contain. It concisely explains set-ups, calling sequences, variable passage and I/O routines. Part II presents an entirely new composite program structure which unloads under the SYSTEM command and executes in both BASIC and machine code with the speed and efficiency of a compiler. Special consideration is given to disk systems. $15.95.*

● **PROGRAMMING THE Z-80**—BK1122—by Rodnay Zaks. Here is assembly language programming for the Z-80 presented as a progressive, step-by-step course. This book is both an educational text *and* a self-contained reference book, useful to both the beginning and the experienced programmer who wish to learn about the Z-80. Exercises to test the reader are included. $14.95.*

● **Z-80 SOFTWARE GOURMET GUIDE AND COOKBOOK**—BK1045—by Nat Wadsworth. Scelbi's newest cookbook! This book contains a complete description of the powerful Z-80 instruction set and a wide variety of programming information. Use the author's ingredients including routines, subroutines and short programs, choose a time-tested recipe and start cooking! $16.99.*

● **Z-80 ASSEMBLY LANGUAGE PROGRAMMING**—BK1177—by Lance A. Leventhal. This book thoroughly covers the Z-80 instruction set, abounding in simple programming examples which illustrate software development concepts and actual assembly language usage. Features include Z-80 I/O devices and interfacing methods, assembler conventions, and comparisons with 8080A/8085 instruction sets and interrupt structure. $16.99.*

● **VOL. I COMPONENT TESTERS**—LB7359— . . . how to build transistor testers (8), diode testers (3), IC testers (3), voltmeters and VTVMs (9), ohmmeters (8 different kinds), inductance (3), capacity (9), Q measurement, crystal checking (6), temperature (2), aural meters for the blind (3) and all sorts of miscellaneous data on meters . . . using them, making them more versatile, making standards. Invaluable book. $4.95.*

● **VOL. II AUDIO FREQUENCY TESTERS**—LB7360— . . . jam packed with all kinds of audio frequency test equipment. If you're into SSB, RTTY, SSTV, etc., this book is a must for you . . . a good book for hi-fi addicts and experimenters, too! $4.95.*

● **VOL. III RADIO FREQUENCY TESTERS**—LB7361—Radio frequency waves, the common denominator of Amateur Radio. Such items as SWR, antenna impedance, line impedance, rf output and field strength; detailed instructions on testing these items includes sections on signal generators, crystal calibrators, grid dip oscillators, noise generators, dummy loads and much more. $4.95.*

● **VOL. IV IC TEST EQUIPMENT**—LB7362—Become a troubleshooting wizard! In this fourth volume of the 73 TEST EQUIPMENT LIBRARY are 42 home construction projects for building test equipment to work with your ham station and in servicing digital equipment. Plus a cumulative index for all four volumes for the 73 TEST EQUIPMENT LIBRARY. $4.95.*

● ***MICROCOMPUTING* CODING SHEETS** *Microcomputing*'s dozen or so programmers wouldn't try to work without these handy scratch pads, which help prevent the little errors that can cost hours and hours of programming time. Available for programming is Assembly/Machine Language (PD1001), which has columns for address, instruction (3 bytes), source code (label, op code, operand) and comments; and for BASIC (PD1002) which is 72 columns wide. 50 sheets to a pad. $2.39.*

PROGRAMMING

The Microprocessor Software Engineering Series by John Zarrella provides common sense descriptions of advanced computer system topics for engineers, programmers and development managers. Each volume is a self-contained review of a software engineering topic, explaining fundamental concepts in easy-to-understand language and describing sophisticated software tools and techniques. Detailed glossary of technical jargon is included in each volume. This series will help you find the solutions to your software problems.

● **OPERATING SYSTEMS: CONCEPTS AND PRINCIPLES**—BK1193—Presents an overview of the basic operating system types, their components and capabilities. $7.95.*

● **WORD PROCESSING AND TEXT EDITING**—BK1194—Provides a firm basis for understanding word processing terminology and for comparing systems. $7.95.*

● **SYSTEM ARCHITECTURE**—BK1195—Presents a detailed overview of advanced computer system design including object architecture and capability-based addressing. $9.95.*

—6502—

* ● **THE APPLE II USER'S GUIDE**—BK1220—by Lon Poole, Martin McNiff, and Steven Cook. This guide is the key to unlocking the full power of your Apple II or Apple II Plus. Topics include: "Applesoft and Integer BASIC Programming"—especially how to make the best use of Apple's sound, color and graphics capabilities. "Machine Level Programming," "Hardware Features"—which covers the disk drive and printer, and "Advanced Programming"—describing high resolution graphics techniques and other advanced applications. Well organized and easy to use. $15.00.*

● **PROGRAMMING THE 6502 (Third Edition)**—BK1005—Rodnay Zaks has designed a self-contained text to learn programming, using the 6502. It can be used by a person who has never programmed before, and should be of value to anyone using the 6502. The many exercises will allow you to test yourself and practice the concepts presented. $12.95.*

● **6502 APPLICATIONS BOOK**—BK1006—Rodnay Zaks presents practical-application techniques for the 6502 microprocessor, assuming an elementary knowledge of microprocessor programming. You will build and design your own domestic-use systems and peripherals. Self-test exercises included. $12.95.*

● **6502 ASSEMBLY LANGUAGE PROGRAMMING**—BK1176—by Lance A. Leventhal. This book provides comprehensive coverage of the 6502 microprocessor assembly language. Leventhal covers over 80 programming examples from simple memory load loops to complete design projects. Features include 6502 assembler conventions, input/output devices and interfacing methods, and programming the 6502 interrupt system. $16.99.*

● **6502 SOFTWARE GOURMET GUIDE AND COOKBOOK**—BK1055—by Robert Findley. This book introduces the BASIC language programmer into the realm of machine-language programming. The description of the 6502 structure and instruction set, various routines, subroutines and programs are the ingredients in this cookbook. "Recipes" are included to help you put together exactly the programs to suit your taste. $12.95.*

8080-6809-68000

* ● **6809 MICROCOMPUTER PROGRAMMING AND INTERFACING**—BK1215—by Andrew C. Staugaard, Jr. Getting involved with Tandy's new Color Computer? If so, this new book from the Blacksburg Group will allow you to exploit the awesome power of the machine's 6809 microprocessor. Detailed information on processor architecture, addressing modes, register operation, data movement, arithmetic logic operations, I/O and interfacing is provided, as well as a review section at the end of each chapter. Four appendices are included covering the 6809 instruction set, specification sheets of the 6809 family of processors, other 6800 series equipment and the 6809/6821 Peripheral Interface Adapter. This book is a must for the serious Color Computer owner. $13.95.*

* ● **68000 MICROPROCESSOR HANDBOOK**—BK1216—by Gerry Kane. Whether you're currently using the 68000, planning to use it, or simply curious about one of the newest and most powerful microprocessors, this handbook has all the answers. A clear presentation of signal conversions, timing diagram conventions, functional logic, three different instruction set tables, exception processing, and family support devices provides more information about the 68000 than the manufacturer's data sheets. A stand alone reference book which can also be used as a supplement to *An Introduction to Microcomputers Vol 2—Some Real Microprocessors* $6.99.*

● **8080A/8085 ASSEMBLY LANGUAGE PROGRAMMING**—by Lance Leventhal—BK1004—Assembly language programming for the 8080A/8085 is explained with a description of the functions of assemblers and assembly instructions, and a discussion of basic software development concepts. Many fully debugged, practical programs are included as is a special section on structured programming. $15.99.*

● **8080 SOFTWARE GOURMET GUIDE AND COOKBOOK**—BK1102—If you have been spending too much time developing simple routines for your 8080, try this new book by Scelbi Computing and Robert Findley. Describes sorting, searching, and many other routines for the 8080 user. $12.95.*

*Use the order card in the back of this magazine or itemize your order on a separate piece of paper and mail to 80 Microcomputing Bookshelf ● Peterborough NH 03458. Be sure to include check or detailed credit card information.

No C.O.D. orders accepted. All above add $1.50 handling. Please allow 4-6 weeks for delivery. Questions regarding your order? Please write Customer Service at the above address.

FOR TOLL FREE ORDERING CALL 1-800-258-5473

PRICES SUBJECT TO CHANGE WITHOUT NOTICE

Why use their flexible discs:

Ampex, Athana, BASF, Caelus, Control Data, Dysan, IBM, Inmac, K-Line, Maxell, Nashua, Scotch, Shugart, Syncom, 3M, Verbatim or Wabash

when you could be using

MEMOREX

for as low as $1.99 each?

Find the flexible disc you're now using on our cross reference list...
then write down the equivalent Memorex part number you should be ordering.

Product Family	Product Description	Memorex Part Number (3201-)	CE quant. 100 price per disc ($)	Ampex	Athana	BASF	Caelus	Dysan	IBM	K-Line	Maxell	Nashua	Scotch 3M	Shugart	Syncom	Verbatim	Wabash	Inmac	Control Data
	IBM Compatible (128 B/S, 26 sectors)	3060	2.19	5FD-111110	473071	53428	CM-F11	800506	2305830	40013	FD1-128	FD-1	740-0	S/A 100	15002	FD34-1000	F111111X	7870-K	421602
Flexible Disc 1s	IBM Compatible (128 B/S, 26 sectors) w/ W P N	3062	2.24	–	–	–	–	–	–	–	–	–	740-0	–	–	–	–	–	–
	IBM Compatible (128 B/S 26 sectors) w/ W P N & Hub ring	3064	2.55	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Single-Headed Drives	IBM Compatible (128 B/S, 26 sectors) REVERSIBLE	1729	3.35	5FD-113110	473072	54431	–	–	–	40015	–	FD-2	740/2-0	–	15150	FF34-2000	F171111X	7860-K	–
Single-Density Media	IBM System 6 Compatible	3066	2.19	–	473077	54561	–	800509	1669959	40014	–	–	740-0 088	–	15003	FD60-1000	F116111X	–	–
	IBM Compatible (256 B/S 15 sectors)	3109	2.19	5FD-111210	473073	–	–	800584	2305845	40040	–	–	740-3600	–	15005	FD36-1000	F112111X	7861-K	–
	IBM Compatible (512 B/S 8 sectors)	3110	2.19	–	473074	–	–	800585	1669954	40044	–	–	–	–	15004	FD60-1000	F113111X	7889-K	–
	Shugart Compatible, 32 hard sector	3015	2.19	5FD-211010	470901	53802	CM-F21	101/1	–	40016	FH1-32	FD-132	740-32	S/A-101	15025	FD32-1000	–	7890-K	421322
	Shugart Compatible, 32 hard sector REVERSIBLE	3025	3.35	5FD-213010	–	–	–	–	–	40017	–	–	740/2-32	–	15151	FF32-2000	–	7880-K	–
	Wang Compatible, 32 hard sector w/Hub ring	3087	2.50	–	–	54491	–	–	–	–	–	–	740-32RM	–	–	–	F37A411X	–	–
	CPT 8000 Compatible	3045	2.79	–	–	–	–	–	–	–	–	–	–	–	15226	–	–	–	–
Flexible Disc 1d	IBM Compatible (128 B/S, 26 sectors)	3090	2.95	5FD-121010	474071	54568	–	3740/1D	–	40047	FD1-128/M2100	FD-1D	741-0	–	–	FD34-8000	F131111X	7857-K	423002
	Soft Sector (128 B/S, 26 sectors) REVERSIBLE	3093	3.99	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Single-Headed Drives	Shugart Compatible, 32 hard sector	3091	2.95	5FD-221010	470801	54596	–	101/1D	–	40024	FH1-32D	–	741-32	S/A-103	15075	FD32-8000	F33A410X	7887-K	423322
Double-Density Media	Shugart Compatible 32 hard sector REVERSIBLE	3094	3.99	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	Wang Compatible 32 hard sector w Hub ring	3088	3.20	–	–	–	–	–	–	–	–	–	–	–	–	–	F22A411X	–	–
Flexible Disc 2s	Soft Sector (Unformatted)	3101	3.84	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	Soft Sector (128 B/S 26 sectors)	3113	3.84	–	–	54428	–	800814	1766870	–	–	–	–	S/A-150	15153	FD10-4026	F121111X	–	–
Double-Headed Drives	Soft Sector (256 B/S, 15 sectors)	3106	3.84	–	473477	54226	–	800815	2736700	40043	FD2-256D	–	742-0	–	15154	FD10-4015	F122111X	7856-K	424612
Single Density Media	32 Hard Sector	3108	3.84	–	–	–	–	–	–	–	FH2-32	–	–	–	–	–	–	–	–
Flexible Disc 2d	Soft Sector (Unformatted)	3102	3.49	–	473485	–	–	DY150	–	40028	FD2-XDM	FD-2D	743-0	–	15103	DD34-4001	–	–	425002
	Soft Sector (128 B/S, 26 sectors)	3115	3.49	–	–	–	–	–	–	–	–	–	–	S/A-150	–	–	–	–	–
Double-Headed Drives	Soft Sector (256 B/S 26 sectors)	3103	3.49	–	473471	54325	–	800817	1766872	40019	FD2-256D	–	743-0/256	–	15101	DD34-4026	F144111X	7858-K	425602
Double-Density Media	Soft Sector (512 B/S 15 sectors)	3114	3.49	–	473472	54479	–	800818	1669044	40039	–	–	743-0/512	–	15100	DD34-4015	F145111X	–	425612
	Soft Sector (1024 B/S. 8 sectors)	3104	3.49	–	473473	54485	–	800819	1669045	40020	–	–	743-0 1024	–	15102	DD34-4008	F147111X	7859-K	425622
	32 Hard Sector	3108	3.49	5FD 321010	470851	–	–	101/2D	–	40021	FH2-32D	–	743-32	S/A-151	15126	DD32-4000	F34A411X	7861-K	425322
	Burroughs B-80 Compatible 32 Hard Sector	3097	3.49	–	–	–	–	–	–	–	–	–	–	–	–	–	F34A611X	–	–
	Soft Sector (1024 B/S. 8 sectors) w/ Hub Ring	3116	3.75	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Flexible Disc FD Memorex 651 or Equiv Drive Compatible	FD VI (Vinyl Jacket)	30712003	2.95	–	470651	–	CM-F31	FDIV	–	40002	–	FD-165	511-0	–	15026	FD65-1000	F61A111X	7910	–
Mini Flexible Disc 1s	Soft Sector (Unformatted)	3401	1.99	–	475001	54256	–	104/1	–	40500	MD1	MD 1	744-0	S/A-104	15300	MD525-01	M11A211X	7897	441002
5¼" Single-Headed Drives	10 Hard Sector	3403	1.99	–	475010	54257	–	107/1	–	40501	–	MD 110	744-10	S/A-107	15326	MD525-10	M41A211X	7898	441102
Single Density Media	16 Hard Sector	3405	1.99	–	475016	54258	–	105/1	–	40502	MH1	MD 116	744-16	S/A-105	15326	MD525-16	M51A211X	7899	441162
	Soft Sector (Unformatted) W/Hub Ring	3431	2.19	–	–	–	–	–	–	–	–	–	–	–	–	MD525-01	–	–	–
	10 Hard Sector, w/Hub Ring	3433	2.19	–	–	–	–	–	–	–	–	–	–	–	–	MD525-10	–	–	–
	16 Hard Sector, w/Hub Ring	3435	2.19	–	–	–	–	–	–	–	–	–	–	–	–	MD525-16	–	–	–
Mini Flexible Disc 1d	Soft Sector	3417	2.24	–	–	54646	–	104/1D	–	–	–	–	–	–	–	MD540-01	–	–	–
5¼" Single-Headed Drives	10 Hard Sector	3418	2.24	–	–	54649	–	107/1D	–	–	–	–	–	–	–	MD540-10	–	–	–
Double-Density Media	16 Hard Sector	3419	2.24	–	–	54652	–	105/1D	–	–	–	–	–	–	–	MD540-16	–	–	–
Mini Flexible Disc 2d	Soft Sector	3421	2.74	–	–	54624	–	104/2D	–	–	MD2-D	–	745-0	S/A-154	–	MD550-01	–	–	–
5¼" Double-Headed Drives	10 Hard Sector	3423	2.74	–	–	54627	–	107/2D	–	–	–	–	745-10	S/A-157	–	MD550-10	–	–	–
Double Density Media	16 Hard Sector	3425	2.74	–	–	54630	–	105/2D	–	–	MH2-D	–	745-16	S/A-155	–	MD550-16	–	–	–

Memorex Flexible Discs...The Ultimate in Memory Excellence

Quality
Memorex means quality products that you can depend on. Quality control at Memorex means starting with the best materials available. Continual surveillance throughout the entire manufacturing process. The benefit of Memorex's years of experience in magnetic media production, resulting, for instance, in proprietary coating formulations. The most sophisticated testing procedures you'll find anywhere in the business.

100 Percent Error Free
Each and every Memorex Flexible Disc is certified to be 100 percent error free. *Each* track of *each* flexible disc is tested, individually, to Memorex's stringent standards of excellence. They test signal amplitude, resolution, low-pass modulation, overwrite, missing pulse error and extra pulse error. They are torque-tested, and competitively tested on drives available from almost every major drive manufacturer in the industry including drives that Memorex manufacturers. Rigid quality audits are built into every step of the manufacturing process and stringent testing result in a standard of excellence that assures you, our customer, of a quality product designed for increased data reliability and consistent top performance.

Customer-Oriented Packaging
Memorex's commitment to excellence does not stop with a quality product. They are proud of their flexible discs and they package them with pride. Both their packaging and their labeling have been designed with your ease of identification and use in mind. The desk-top box containing ten discs is convenient for filing and storage. Both box labels and jacket labels provide full information on compatibility, density, sectoring, and record length. Envelopes with multi-language care and handling instructions and color-coded removable labels are included. A write-protect feature is available to provide data security.

Full One Year Warranty — Your Assurance of Quality
Memorex Flexible Discs will be replaced free of charge by Memorex if they are found to be defective in materials or workmanship within one year of the date of purchase. Other than replacement, Memorex will not be responsible for any damages or losses (including consequential damages) caused by the use of Memorex Flexible Discs.

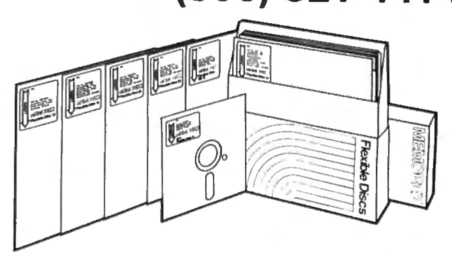
Quantity Discounts Available
Memorex Flexible Discs are packed 10 discs to a carton and 10 cartons to a case. Please order only in increments of 100 units for quantity 100 pricing. We are also willing to accommodate your smaller orders. Quantities less than 100 units are available in increments of 10 units at a 10% surcharge. **Quantity discounts** are also available. Order 500 or more discs at the same time and deduct 1%; 1,000 or more saves you 2%; 2,000 or more saves you 3%; 5,000 or more saves you 5%; 10,000 or more saves you 7%; 25,000 or more saves you 8%; 50,000 or more saves you 9% and 100,000 or more discs earns you a 10% discount off our super low quantity 100 price. Almost all Memorex Flexible Discs are immediately available from **CE**. Our warehouse facilities are equipped to help us get you the quality product you need, when you need it. If you need further assistance to find the flexible disc that's right for you, call the Memorex compatibility hotline. Dial 800-538-8080 and ask for the *flexible disc hotline* extension 0997. In California dial 800-672-3525 extension 0997.

Buy with Confidence
To get the fastest delivery from **CE** of your Memorex Flexible Discs, send or phone your order directly to our Computer Products Division. Be sure to calculate your price using the **CE** prices in this ad. Michigan residents please add 4% sales tax. Written purchase orders are accepted from approved government agencies and most well rated firms at a 10% surcharge for net 10 billing. All sales are subject to availability. All sales are final. Prices, terms and specifications are subject to change without notice. Out of stock items will be placed on backorder automatically unless **CE** is instructed differently. International orders are invited with a $20.00 surcharge for special handling in addition to shipping charges. All shipments are F.O.B. Ann Arbor, Michigan. No COD's please. Non-certified and foreign checks require bank clearance.

Mail orders to: **Communications Electronics**, Box 1002, Ann Arbor, Michigan 48106 U.S.A. Add $8.00 per case or partial-case of 100 8-inch discs or $6.00 per case of 100 5¼-inch mini-discs for U.P.S. *ground* shipping and handling in the continental U.S.A. If you have a Master Charge or Visa card, you may call anytime and place a credit card order. **Order toll-free** in the United States. Call anytime 800-521-4414. If you are outside the U.S. or in Michigan, dial 313-994-4444. **Dealer inquiries invited.** All order lines at **Communications Electronics** are staffed 24 hours.

MEMBER DIRECT MAIL MARKETING ASSOCIATION · CODE OF ETHICS · VISA · master charge

Order Toll-Free!
(800) 521-4414



For Data Reliability—Memorex Flexible Discs

™

COMMUNICATIONS ELECTRONICS™

✓263

Computer Products Division

854 Phoenix ☐ Box 1002 ☐ Ann Arbor, Michigan 48106 U.S.A.
Call TOLL-FREE (800) 521-4414 or outside U.S.A. (313) 994-4444

✓See List of Advertisers on page 354

LIST OF ADVERTISERS

For further information from our advertisers, please use the Reader Service card.

*This advertiser prefers to be contacted directly.

READER SERVICE

This card valid until October 31, 1981

Reader Service: To receive more information from any of the advertisers in this issue of *80 Microcomputing*, circle the number on the postage-paid Reader Service Card that corresponds with the Reader Service number on the ad in which you are interested. You will find numbers, preceded by a ✓, near the logo of each advertiser. Complete the entire card, drop into a mailbox and in 4-6 weeks you will hear from the advertiser directly.

Please help us to bring you a better magazine—by answering these questions.

I. What is your age?
- ☐ A. under 18
- ☐ B. 18-22
- ☐ C. 23-40
- ☐ D. 41-60
- ☐ E. over 60

II. What is your occupation?
- ☐ 1. Professional
- ☐ 2. Engineer
- ☐ 3. Data processing
- ☐ 4. Business
- ☐ 5. Education
- ☐ 6. Technician
- ☐ 7. Student
- ☐ 8. Other

III. What are your primary applications of your TRS-80 (check only two)?
- ☐ A. Business
- ☐ B. Games
- ☐ C. Home
- ☐ D. Education
- ☐ E. Scientific
- ☐ F. Control
- ☐ G. Music

IV. Your TRS-80, is it a
- ☐ 1. Level I
- ☐ 2. Level II
- ☐ 3. Model II
- ☐ 4. Don't own one yet

V. What peripherals do you have (check all that apply)?
- ☐ A. Expansion interface
- ☐ B. Disk
- ☐ C. Printer

VI. How much have you spent on hardware?
- ☐ 1. less than $500
- ☐ 2. $500-1,000
- ☐ 3. $1,000-2,000
- ☐ 4. $2,000-4,000
- ☐ 5. $4,000-6,000
- ☐ 6. more than $6,000

VII. How much have you spent on software?
- ☐ A. less than $100
- ☐ B. $100-250
- ☐ C. $250-500
- ☐ D. $500-1,000
- ☐ E. more than $1,000

VIII. What is your level of education?
- ☐ 1. Post-graduate
- ☐ 2. College
- ☐ 3. High school

IX. How many people read your copy of 80 Microcomputing?
- ☐ A. 1
- ☐ B. 2
- ☐ C. 3
- ☐ D. 4 or more

X. If you are not a subscriber please circle number 500.

1	6	11	16	21	151	156	161	166	171	301	306	311	316	321	451	456	461	466	471
2	7	12	17	22	152	157	162	167	172	302	307	312	317	322	452	457	462	467	472
3	8	13	18	23	153	158	163	168	173	303	308	313	318	323	453	458	463	468	473
4	9	14	19	24	154	159	164	169	174	304	309	314	319	324	454	459	464	469	474
5	10	15	20	25	155	160	165	170	175	305	310	315	320	325	455	460	465	470	475
26	31	36	41	46	176	181	186	191	196	326	331	336	341	346	476	481	486	491	496
27	32	37	42	47	177	182	187	192	197	327	332	337	342	347	477	482	487	492	497
28	33	38	43	48	178	183	188	193	198	328	333	338	343	348	478	483	488	493	498
29	34	39	44	49	179	184	189	194	199	329	334	339	344	349	479	484	489	494	499
30	35	40	45	50	180	185	190	195	200	330	335	340	345	350	480	485	490	495	500
51	56	61	66	71	201	206	211	216	221	351	356	361	366	371	501	506	511	516	521
52	57	62	67	72	202	207	212	217	222	352	357	362	367	372	502	507	512	517	522
53	58	63	68	73	203	208	213	218	223	353	358	363	368	373	503	508	513	518	523
54	59	64	69	74	204	209	214	219	224	354	359	364	369	374	504	509	514	519	524
55	60	65	70	75	205	210	215	220	225	355	360	365	370	375	505	510	515	520	525
76	81	86	91	96	226	231	236	241	246	376	381	386	391	396	526	531	536	541	546
77	82	87	92	97	227	232	237	242	247	377	382	387	392	397	527	532	537	542	547
78	83	88	93	98	228	233	238	243	248	378	383	388	393	398	528	533	538	543	548
79	84	89	94	99	229	234	239	244	249	379	384	389	394	399	529	534	539	544	549
80	85	90	95	100	230	235	240	245	250	380	385	390	395	400	530	535	540	545	550
101	106	111	116	121	251	256	261	266	271	401	406	411	416	421	551	556	561	566	571
102	107	112	117	122	252	257	262	267	272	402	407	412	417	422	552	557	562	567	572
103	108	113	118	123	253	258	263	268	273	403	408	413	418	423	553	558	563	568	573
104	109	114	119	124	254	259	264	269	274	404	409	414	419	424	554	559	564	569	574
105	110	115	120	125	255	260	265	270	275	405	410	415	420	425	555	560	565	570	575
126	131	136	141	146	276	281	286	291	296	426	431	436	441	446	576	581	586	591	596
127	132	137	142	147	277	282	287	292	297	427	432	437	442	447	577	582	587	592	597
128	133	138	143	148	278	283	288	293	298	428	433	438	443	448	578	583	588	593	598
129	134	139	144	149	279	284	289	294	299	429	434	439	444	449	579	584	589	594	599
130	135	140	145	150	280	285	290	295	300	430	435	440	445	450	580	585	590	595	600

Name ______________________

Address ______________________

City ______________ State ________ Zip ________

80 Microcomputing • POB 2743 • Clinton IA 52735

BOOKS

80 Microcomputing
Peterborough NH 03458

Please send me the following 80 Microcomputing products:

Qty.	Catalog #	Title	Unit Price	Total

Add $1 shipping/handling ________

Total ________

Enclosed $ ________ Check M.O.

Bill: AE MC VISA

Card # ______________________

Exp date ________ Interbank # ________

Signature ______________________

Name ______________________

Address ______________________

City ______________ State ________ Zip ________

Please allow 4–6 weeks for delivery

No C.O.D. orders accepted.

9-81

SUBSCRIPTION

80 Microcomputing POB 981
Farmingdale NY 11737

BEAT THE PRICE INCREASE

Subscribe by October 1, 1981

Please allow 4–6 weeks for delivery

New Subscription Renewal

1 year—$18

Enclosed $ ________ Check M.O.

Bill: MC VISA AE Me

Card # ______________ Exp. date ________

Signature ______________ Interbank # ________

Name ______________________

Address ______________________

City ______________ State ________ Zip ________

Canada: $20 1 year only, US funds
Other foreign: $28 one year only, US funds

319B9

All previous subscription offers are void on October 1, 1981

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 780 FARMINGDALE NY 11737

POSTAGE WILL BE PAID BY ADDRESSEE

80 microcomputing™

Subscription Dept.
POB 981
Farmingdale NY 11737

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 80 PETERBOROUGH NH 03458

POSTAGE WILL BE PAID BY ADDRESSEE

80 microcomputing™

Peterborough NH 03458

Att. Mail Order

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 217 CLINTON IA 52735

POSTAGE WILL BE PAID BY ADDRESSEE

80 microcomputing T.M.

POB 2743
Clinton IA 52735

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

Compare our prices with any in the magazine.

We win.

16 K RAM KITS

NEC 4116 200 ns	17.95
DIP SHUNTS (FOR KEYBOARD MEM UPGRADE)	4.00

TRS-80 MOD I HARDWARE

PERCOM DATA SEPARATOR	27.00
PERCOM DOUBLER	159.00
DOUBLE ZAP II/80 (CONVERTS ND/80 TO D.D.)	45.95
MPI 40 TRACK DISK DRIVE (B-51)	299.00
SHUGART 40 TRACK DISK DRIVE (400L)	299.00
MPI 80 TRACK DISK DRIVE (B-91)	429.00
TANDON 80 TRACK DISK DRIVE	429.00
TANDON 40 TRACK DISK DRIVE	299.00
LNW DOUBLER WITH DOSPLUS 3.3D	159.00
PERCOM SPEAK-2ME-2	64.95

TRS-80 SOFTWARE

LAZY WRITER MOD I	125.00
PROSOFT MOD I, III	79.00
SPECIAL DELIVERY (MAIL LIST PROG) MOD I, III	119.00
X-TRA SPECIAL DELIVERY (MAIL LIST PROG)	199.00
TRACKCESS MOD I	24.95
OMNITERM SMART TERMINAL PKG MOD I, III	89.95

PRINTERS

ANADEX DP 9000	1195.00
ANADEX DP 9001	1195.00
ANADEX DP 9500	1295.00
ANADEX DP 9501	1295.00
CENTRONICS 739	765.00
C-ITOH 25 CPS PARALLEL	1440.00
C-ITOH 25 CPS SERIAL	1495.00
C-ITOH 45 CPS PARALLEL	1770.00
C-ITOH 40 CPS SERIAL	1870.00
C-ITOH TRACTOR OPTION	175.00
EPSON MX-80	$CALL
EPSON MX-80 F/T	$CALL
EPSON MX-100 F/T	$CALL
IDS-445G PAPER TIGER	795.00
IDS-460G PAPER TIGER	1149.00
IDS-560G PAPER TIGER	1495.00
INFOSCRIBE 500 9X9, 150 CPS (TI-810 REPLACEMENT)	1495.00
MALIBU 200 DUAL MODE	2995.00
NEC SPINWRITER 5510 SERIAL RO	2595.00
NEC SPINWRITER 5530 PARALLEL RO	2595.00
NEC SPINWRITER 5500 D SELLUM OPTION	2795.00
NEC SPINWRITER 3500 SELLUM OPTION	2195.00
OKIDATA MICROLINE 80	399.00
OKIDATA MICROLINE 82	599.00
OKIDATA MICROLINE 83	795.00

MODEMS

NOVATION CAT ACOUSTIC MODEM	145.00
NOVATION D-CAT DIRECT CONNECT MODEM	155.00
NOVATION AUTO-CAT AUTO ANSWER MODEM	229.00
UDS 103LP DIRECT CONNECT MODEM	175.00
LEXICON LX-11 MODEM	115.00

DISKETTES

MD 525-01,10,16	26.50
MD 550-01,10,16	44.50
MD 577-01,10,16	34.80
MD 557-01,10,16	45.60
FD 32 OR 34 -8000	45.60

DISKETTE STORAGE

5¼" PLASTIC LIBRARY CASE	2.50
8" PLASTIC LIBRARY CASE	3.50
PLASTIC STORAGE BINDER WITH INSERTS	9.95
PROTECTOR 5¼"	24.95
PROTECTOR 8"	29.95

CP/M SOFTWARE

MICROSOFT BASIC-80	299.00
MICROSOFT BASIC COMPILER	319.00
MICROSOFT FORTRAN-80	399.00
PEACHTREE SYSTEMS	CALL
MAGIC WAND (REQUIRES CP/M)	275.00
WORD STAR (REQUIRES CP/M)	310.00
MAILMERGE (REQUIRES WORD STAR)	100.00
SPELLGUARD	239.00
CP/M PICKLES & TROUT FOR TRS-80 MOD II	175.00

CORVUS

TRS-80 MOD I, II

Controller, Case/P.S., Operating System, A&T

5 Megabytes	3095.00
10 Megabytes	4495.00
20 Megabytes	5395.00
MIRROR BACK-UP	700.00

SPECIALS

NO. 1 - TRS-80 DISK & OTHER MYSTERIES/BOX OF DISKS/PLASTIC LIBRARY CASE	44.50
NO. 3 - NEWDOS/80 2.0/BOX OF DISKS/PLASTIC LIBRARY CASE	149.00
NO. 4 - MICROSOFT BASIC COMPILER/BOX OF DISKS/PLASTIC LIBRARY CASE	190.00
NO. 5 - MICROSOFT BASIC D & OTHER MYSTERIES/BOX OF DISKS/PLASTIC LIBRARY CASE	49.95
NO. 6 - DOUBLE DENSITY SPECIAL NEWDOS/80 DOUBLE ZAP II, BOX OF DISKS, PLASTIC LIBRARY CASE	309.00
TANDON 40 TRACK WITH NEWDOS +	320.00

SUPPLIES

AVERY TABULABLES

1,000 3-1/2 x 15/16	8.49
3,000 3-1/2 x 15/16	14.95
5,000 3-1/2 x 15/16	19.95

UARCO PAPER (Prices F.O.B. S.P.)

9-1/2 x 11 18 lb white	29.00
14-7/8 x 11 18 lb white	39.00

We stock a complete line of computer covers, printer ribbons, print wheels & NEC thimbles. Call us for your needs.

Alpha Byte STORES

69

We built a reputation on our prices and your satisfaction.

We guarantee everything we sell for 30 days. If anything is wrong, just return the item and we'll make it right. And, of course, we'll pay the shipping charges.

We accept Visa and Master Card on all orders. COD accepted up to $300.00. We also accept school purchase orders.

Please add $2.00 for standard UPS shipping and handling on orders under 50 pounds, delivered in the continental U.S. Call us for shipping charges on items that weigh more than 50 pounds. Foreign, FPO and APO orders please add 15% for shipping. California residents add 6% sales tax.

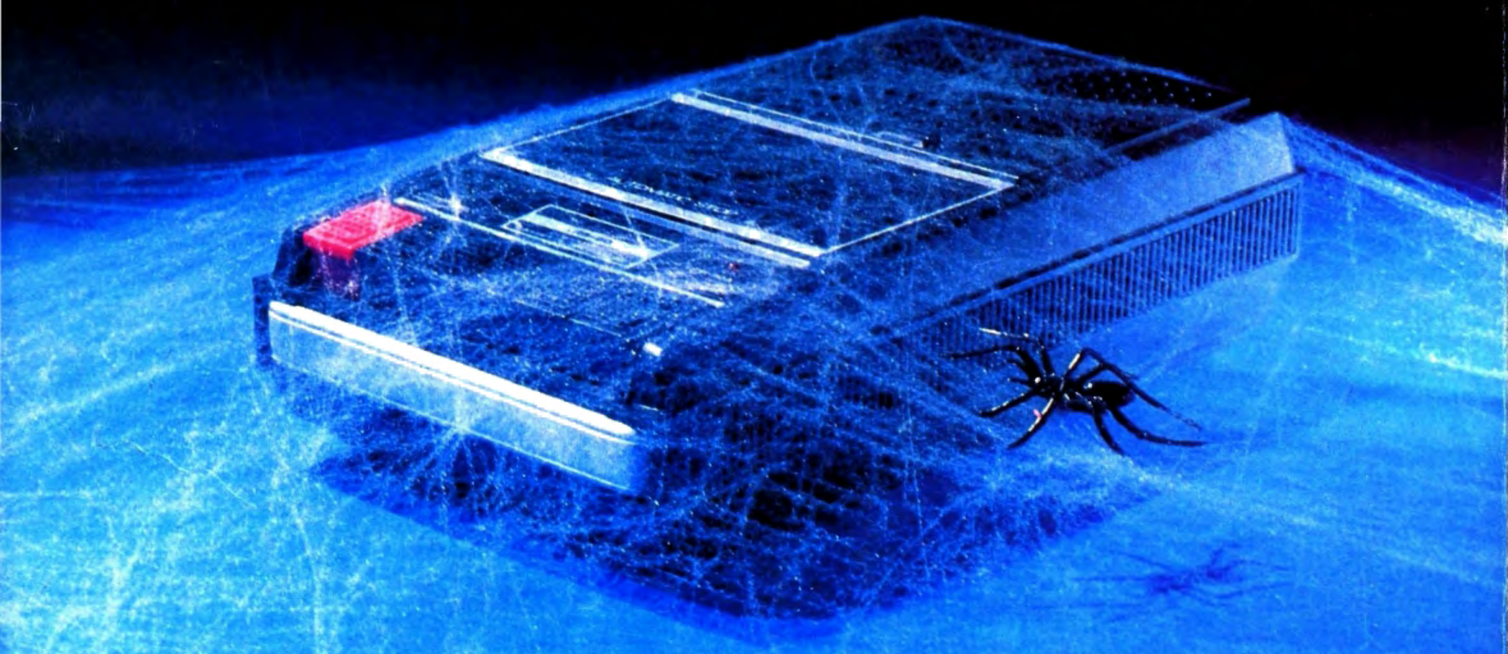
(213) 883-8594

31245 La Baya Drive, Westlake Village, California 91362

CP/M is a trademark of Digital Research.

The prices quoted are only valid for stock on hand and all prices are subject to change without notice.

TIRED OF WAITING?



Frustrating isn't it! No matter how much you speed up your program it still seems to take forever to save data onto a cassette. Wouldn't it be great if someone could design a mass storage system with the speed of a disk, but at half the cost? *Exatron* did, the *Exatron Stringy Floppy (ESF)* .

Totally self-contained, the ESF is an extremely fast, reliable, and economical alternative to cassette or disk storage of programs or data. All of the ESF's operations are under the computer's control, with no buttons, switches, knobs or levers to adjust or forget.

The ESF uses a miniature tape cartridge, about the size of a business card, called a wafer. The transport mechanism uses a direct drive motor with only one moving part. Designed to read and write digital data only, the ESF suffers from none of the drawbacks of cassettes - without the expense of disks.

Several versions of the ESF are available, for the *TRS-80, Apple, PET, OSI* and an *RS 232* unit. Even the slowest of the units is 15 times faster than a cassette, and all are as reliable as disk drives - in fact a lot of users say they are *more* reliable!

excellence in electronics

exatron

To get further information about the ESF give Exatron a call on their Hot Line *800-538 8559* (inside California 408-737 7111).

If you can't wait any longer then take advantage of their 30 day money-back guarantee, you've nothing to lose but time!

181 Commercial Street
Sunnyvale, CA 94086

✓3

TRS-80, Apple and PET are trademarks of Tandy, Apple and Commodore respectively.